Multimedia Systems
DOI 10.1007/s00530-017-0539-8

REGULAR PAPER

A content-based recommendation algorithm for learning resources

Jiangbo Shu[1] · Xiaoxuan Shen[1] · Hai Liu[1] · Baolin Yi[1] · Zhaoli Zhang[1]

Received: 6 June 2016 / Accepted: 30 January 2017

Abstract Automatic multimedia learning resources recommendation has become an increasingly relevant problem: it allows students to discover new learning resources that match their tastes, and enables the e-learning system to target the learning resources to the right students. In this paper, we propose a content-based recommendation algorithm based on convolutional neural network (CNN). The CNN can be used to predict the latent factors from the text information of the multimedia resources. To train the CNN, its input and output should first be solved. For its input, the language model is used. For its output, we propose the latent factor model, which is regularized by L_1-norm. Furthermore, the split Bregman iteration method is introduced to solve the model. The major novelty of the proposed recommendation algorithm is that the text information is used directly to make the content-based recommendation without tagging. Experimental results on public databases in terms of quantitative assessment show significant improvements over conventional methods. In addition, the split Bregman iteration method which is introduced to solve the model can greatly improve the training efficiency.

Keywords Resources recommendation · Convolutional neural network · L_1 norm · Split Bregman iteration method

Communicated by B. Prabhakaran.

✉ Xiaoxuan Shen
shenxx630@gmail.com

[1] National Engineering Research Center for E-Learning, Central China Normal University, Wuhan, Hubei 430079, China

1 Introduction

Recommendation algorithms, which have become extremely popular in recent years, are widely used in the field of movies [1–3], music [4, 5], news [6, 7], TV [8, 9], etc. The task of recommendation algorithms in e-learning systems is to give student a personalized and suitable learning service [10–13]. The learning resources are more and more diversified recent years, and it could be audio, video, pictures, text, and so on.

In the past decades, a multitude of recommendation algorithms has been developed. They can be divided into two groups: history data-based recommendation (HDBR) methods and content-based recommendation (CBR) methods. The HDBR methods have been widely researched for recommendation systems. These methods only rely on the user's history data without requiring the details of such resources. Collaborative Filtering (CF) [12, 14–16] is one of the most distinguished approaches. CF methods can be classified into two types: neighbourhood-based method [12] and model-based methods [16]. Model-based methods use L_2 norm to normalize the solution [16, 17]. HDBR methods require extensive historical data, which is difficult to obtain from the e-learning system and to achieve a decent performance, and they always suffer from the "cold start" problem. Thus, CBR methods may be a better choice for learning resources recommendation in e-learning systems.

In contrast, CBR methods [5, 18] create a profile to characterize each user and item. For example, a movie profile typically contains information about its genre (action, comedy, etc), cast, box office popularity, release date, etc. Then, the system recommends items based on their profiles. However, learning resources have become more diversified in recent years. Today, a learning resource could come in various formats: audio, video, pictures, text, etc, and there is a huge amount of implicit information in learning resources that can be difficult to obtain, such as knowledge point, complexity, and prepared knowledge. Hence, content-based recommendation algorithm for learning resources is difficult to construct.

With the great achievements on vision [19–28], and speech tasks [23, 29], and deep learning (DL) models have become a hot research field. Using deep learning algorithm, artificial intelligence has led to big breakthroughs in many areas. Inspired by these developments, we attempted to use the DL model to directly analyse the content of learning resources and then make recommendations.

To build a high-efficiency content-based recommendation algorithm in an e-learning system, we propose a content-based learning resources recommendation algorithm by convolutional neural network (CBCNN). CBCNN uses text information in learning resources (e.g., course introduction or classroom content in MOOC platform; the abstract or full content of the learning resources) to predict the latent factors of the learning resources [30–33]. In the following discussion, the latent factors are utilized to predict the rating scores between students and learning resources. The CBCNN can process the content from the learning resources directly without tagging by humans.

Although the application considered here is learning resources recommendation, the method is more generally applicable to news recommendation, video recommendation, book recommendation, and so on. The CBCNN can use all kind of text information of items to make the recommendation decision. For example, the CBCNN can utilize the brief introduction of the video to make recommendation.

The outline of the paper is organized as follows. In the next section, we introduce the proposed recommendation system based on the convolutional neural network (CBCNN). To construct CNN, the latent factor model is proposed for its output and the language model for its input. The detail processing is presented in Sect. 3. Some results and discussions are given in Sect. 4, followed by a brief "Conclusion" section.

2 Outline of CBCNN

CBCNN can be summed up as two processes, namely, the training process and the recommendation process (Fig. 1). The recommendation process can work by the CNN parameters provided by the training process.

2.1 Training process

In Fig. 1, a CNN was constructed, which is shown as Level 2. To train the CNN, its input and output should first be solved. For the input, a language model is used. For the output, we proposed the latent factor model, which is regularized by L_1-norm.

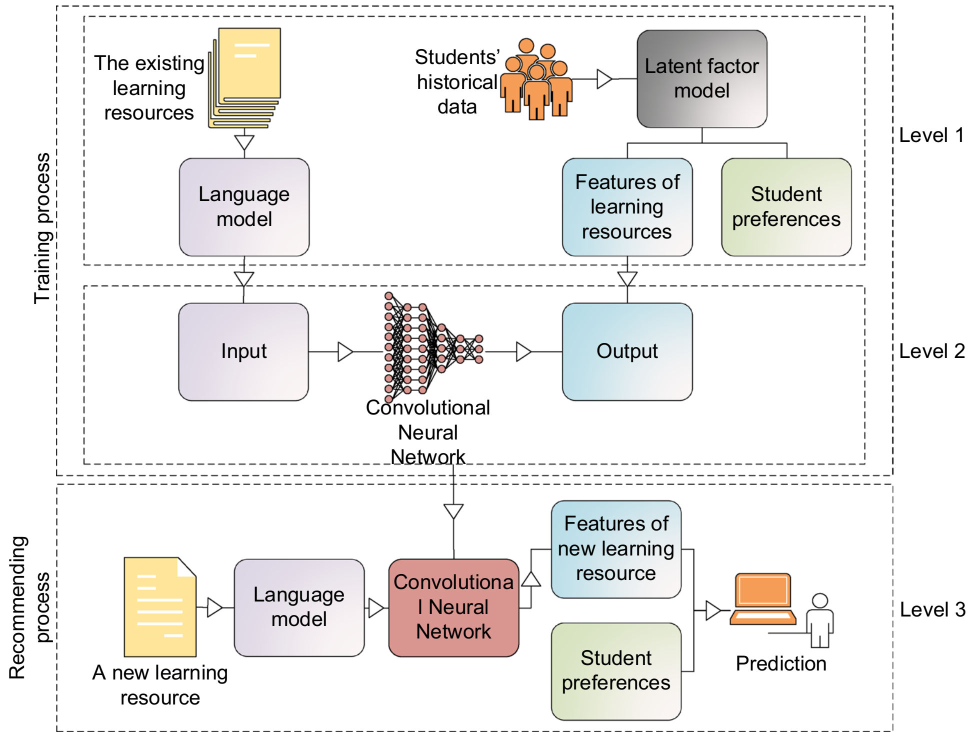

Fig. 1 Architecture of our recommendation algorithm. There are three levels in our recommendation algorithm. *Level 1* denotes how the input and output of CNN are produced from the historical data. *Level 2* shows that the CNN parameters are trained by its input and output. Both *level 1* and *level 2* are the training process. *Level 3* finished a new learning resource recommendation process

The training data for LFM are the historical rating scores between the students and the learning resources. The rating scores can be explicit in that it is based on marks made students, or implicit, and in that it is conjectured from the students' behaviours. The input data for the language model are the existing learning resources' text information.

2.2 Recommendation process

The recommendation process (Level 3) involves using the text information regarding the input learning resource—maybe the content itself or brief introduction of the content. Then, the CNN can turn the input text information into features of the learning resource; finally, in combination with the student's preferences, the rating can be predicted.

The recommendation algorithm can provide a service to predict the rating score between a student and a learning resource. This can work well for new learning resources. The predicted rating score indicates whether or not the student needs the learning resource. The proposed recommendation algorithm can be utilized as a new recommendation system, and it can also be applied to enhance an existing recommendation system.

3 Proposed method

In this section, we will describe how one can build a CBCNN. The CNN can be used to predict the latent factors from the text information. To construct the CNN model, its output and input need to be established first. The CNN output is solved by the latent factor model from the historical rating scores data. The CNN input is achieved using the language model according to the text information.

3.1 Construction of the CNN model

The CNN model, shown in Fig. 2, is a four-layer convolutional neural network with two convolutional layers, one region pooling layer, and one fully connected layer. Let $x_i \in \mathfrak{R}^k$ be the k-dimensional word representation vector corresponding to the ith word in the article. An article of length n is represented as $x = \left[x_1, x_2, ..., x_n\right]$, $x \in \mathfrak{R}^{nk}$. A convolution operation involves a filter $\mathbf{w} \in \mathfrak{R}^{sk}$, which is applied to a word representation vector to produce a new feature, as shown in Layer 1.

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_i + b). \tag{1}$$

Here, $b \in \mathfrak{R}$ is a bias term and f is a non-linear function, such as the sigmoid function. This filter is applied to each word in the sentence to produce a feature map $\mathbf{c} = \left[c_1, c_2, \ \ldots, c_{n-s+1}\right]$, $c \in \mathfrak{R}^{n-s+1}$. We then apply a mean-over-time region pooling operation over the feature map, Layer 2 achieve the pooling operation in λ regions

$$b_i = \max\left\{c_{(i-1)\times(n/\lambda)+1}, ..., c_{i\times(n/\lambda)}\right\}, i \in [1, \lambda]. \tag{2}$$

Layer 3 is a convolutional layer. A convolution operation involves a filter $\mathbf{w} \in \mathfrak{R}^{\lambda}$, which is applied on $\mathbf{b} = \left[b_1, b_2, ..., b_\lambda\right]$ to produce a feature value:

$$a = f(\mathbf{w} \cdot \mathbf{b} + b). \tag{3}$$

We have described the process that extracts one feature from one filter. The model uses multiple filters to obtain multiple features. These features are passed to a fully connected layer, whose output is the predicted latent factors, as shown in Layer 4.

Latent factor vectors are real-valued, so the most straightforward objective is to minimize the mean squared error (MSE) of the predictions. Let y_i' be the latent factor

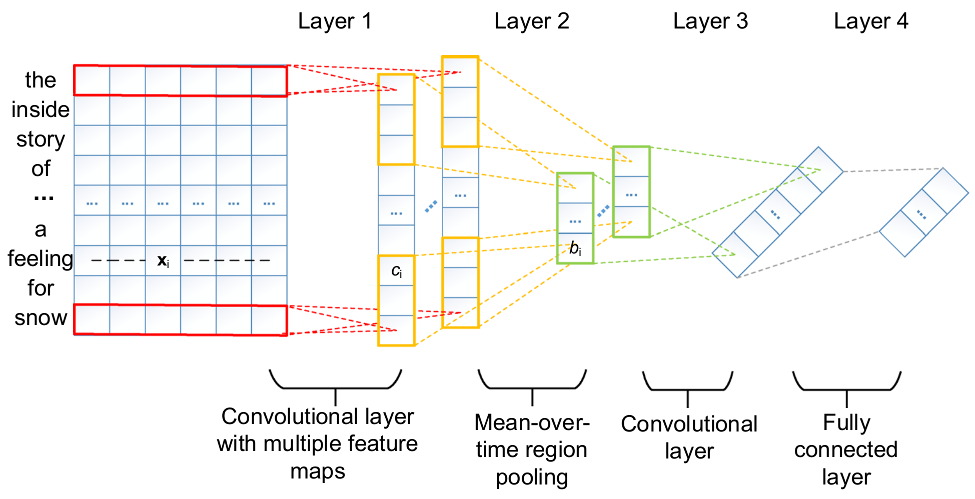

Fig. 2 Construction of convolutional neural network. *Layer 1* is a convolutional layer with multiple feature maps. *Layer 2* is a mean-over-time pooling layer. *Layer 3* is an over-time convolutional layer. *Layer 4* is a full connected layer, and is then the output

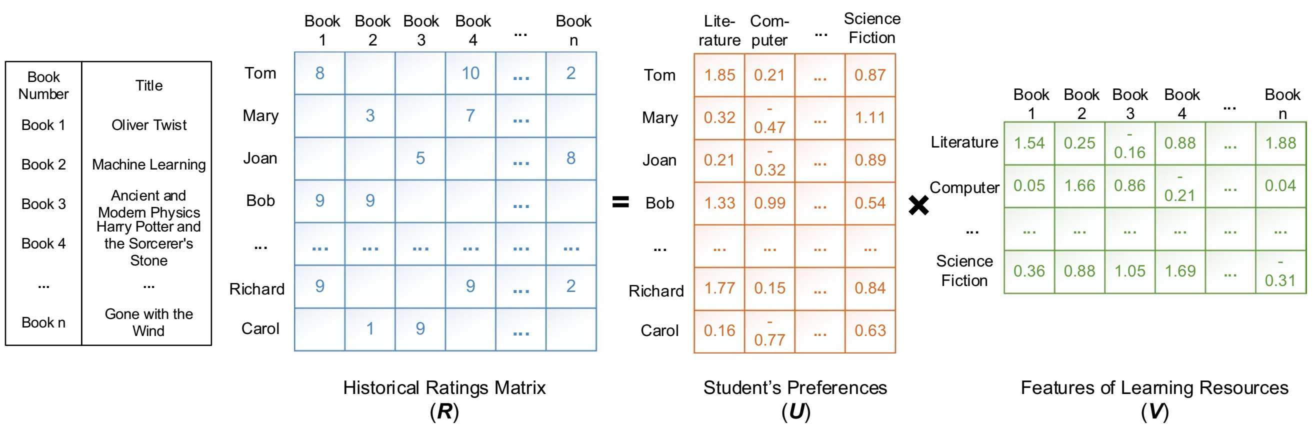
Book Number
Title
Book 1 Oliver Twist
Book 2 Machine Learning
Book 3 Ancient and Modern Physics
Book 4 Harry Potter and the Sorcerer's Stone
... ...
Book n Gone with the Wind
Book 1 Book 2 Book 3 Book 4 ... Book n
Tom 8 10 ... 2
Mary 3 7 ...
Joan 5 ... 8
Bob 9 9 ...
... ...
Richard 9 9 ... 2
Carol 1 9 ...
Historical Ratings Matrix (R)
=
Lite-rature Com-puter ... Science Fiction
Tom 1.85 0.21 ... 0.87
Mary 0.32 -0.47 ... 1.11
Joan 0.21 -0.32 ... 0.89
Bob 1.33 0.99 ... 0.54
... ...
Richard 1.77 0.15 ... 0.84
Carol 0.16 -0.77 ... 0.63
Student's Preferences (U)
×
Book 1 Book 2 Book 3 Book 4 ... Book n
Literature 1.54 0.25 -0.16 0.88 ... 1.88
Computer 0.05 1.66 0.86 -0.21 ... 0.04
... ...
Science Fiction 0.36 0.88 1.05 1.69 ... -0.31
Features of Learning Resources (V)

Fig. 3 Process of the latent factor model (LFM). Matrix R denotes the historical rating scores matrix made by all users; the vacancy means that the user did not give a corresponding rating. Matrix ***U*** denotes the relationship between the students and the latent factors. Matrix ***V*** denotes the relationship between the learning resources and the latent factors

vector for article *i* and y_i is the output of the CNN model. The objective functions are then (*w* and *b* represent all the parameters in the CNN model) (Fig. 3):

$$\arg\min_{w,b} \sum_i \left\| \mathbf{y}_i{}' - \mathbf{y}_i \right\|^2. \tag{4}$$

3.2 CNN output solved by latent factor model

To achieve the features of students and learning resources, the latent factor model (LFM) is used. In the traditional LFM L_2-norm regularization is often used. However, it usually results in the over-smoothing problem. In our algorithm, the LFM results represent the features of students and learning resources. Thus, it is more reasonable for us to use the sparse prior to regularize the result.

According to the analysis above, we proposed a modified matrix factorization method and tried to use the sparse prior (L_1-norm based regularization) to normalize the solution; the model is presented as

$$J(U,V) = \sum_{ij} \left(U_{i*} \cdot V_{*j} - r_{ij} \right)^2 + \lambda_1 \|U\|_1 + \lambda_2 \|V\|_1, \tag{5}$$

where the first term is the fidelity term, and the rest of the terms are the regularization terms. The matrix ***U*** denotes the relationship between the students and the latent factors, and matrix ***V*** means denotes the relationship between the learning resources and the latent factors. The symbol r_{ij} means the rating score made by the *i*th student to the *j*th learning resource (only calculate the nonempty part in the historical rating data). λ_1 and λ_2 are the regularization parameters, which can balance the fidelity term and regularization terms. To minimize Eq. (5), the split Bregman iteration method [34] is introduced. The split Bregman iteration method is a very efficient optimization method, which is important for the recommendation system. Next, we will describe how one can use the split Bregman iteration method to optimize the modified LFM.

This iterative method starts with an initial random matrix ***V***. Then, for a fixed ***V***, the cost function is minimized with respect to matrix ***U***. Thus, the learning resource features can be estimated. Then, we fixed ***U***, and a new estimate for ***V*** is obtained. This procedure continues repeatedly until convergence is reached, i.e., ***U*** and ***V*** are updated in a cyclic fashion.

After fix the matrix ***V***, the optimization problem becomes

$$J(U) = \sum_{ij} \left(U_{i*} \cdot V_{*j} - r_{ij} \right)^2 + \lambda_1 \|U\|_1. \tag{6}$$

After obtaining the "split Bregman" formulation, introduce the variables $\boldsymbol{d}_U$, $\boldsymbol{b}_U$, and $\boldsymbol{d}_U = \mathbf{U}$, $\mathbf{b}_U = \boldsymbol{d}_U - \boldsymbol{U}$

$$\min_{U,d_U} \left\| d_U \right\|_1 + H(U), \tag{7}$$

where

$$H(U) = \left(\frac{1}{\lambda_1} \sum_{i,j} \left(U_{i*} \cdot V_{*m} - r_{ij} \right)^2 \right) + \frac{\theta_U}{2} \left\| d_U - U - b_U \right\|^2. \tag{8}$$

Then, break this algorithm down into three easy steps:

$$\begin{cases} U^{k+1} = \arg\min\limits_U H(U) \\ d_U^{k+1} = \arg\min\limits_{d_U} \left\| d_U \right\|_1 + \frac{\theta_U}{2} \left\| d_U - U - b_U^k \right\|^2 \\ b_U^{k+1} = b_U^k + U^{k+1} - d_U^{k+1} \end{cases}. \tag{9}$$

The first equation in Eq. (9) can be optimized by gradient-descent algorithm (α is iteration step size):

$$U_{im}^{k+1} = U_{im}^k - \frac{\partial H(U)}{\partial U_{im}} \times \alpha, \tag{10}$$

$$\frac{\partial H(U)}{\partial U_{im}} = \frac{1}{\lambda_1}\sum_j 2V_{mj}\left(U_{i*}\cdot V_{*j} - r_{ij}\right) - \theta_U\left(d_{im} - U_{im} - b_{im}^k\right). \tag{11}$$

The second equation in Eq. (9) can be solved efficiently using shrinkage

$$d_U^{k+1} = shrink\left(U^{k+1} + b_U^{k+1}, \frac{1}{\theta_U}\right). \tag{12}$$

Then, fix $\boldsymbol{U}$. The split Bregman iteration method is also used to solve $\boldsymbol{J}(\boldsymbol{V})$. The solution steps are similar to that of Eqs. (7)–(12), replacing θ_U, $\boldsymbol{d}_U$, and $\boldsymbol{b}_U$ with θ_V, $\boldsymbol{d}_V$, and $\boldsymbol{b}_V$.

The regularization parameters, λ_1, λ_2, balance the data fidelity (the first term) and the regularization (the second and third terms) in Eq. 5. The two parameters control the sparsity of the $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. As λ_1 and λ_2 are increased, the feature matrix of users and items, $\boldsymbol{U}$ and $\boldsymbol{V}$, become sparser. The parameters θ_U and θ_V introduced in the split Bregman iteration method control noise ratio in solution procedure. θ_U and θ_V can be set as $\beta\cdot\lambda_1$ and $\beta\cdot\lambda_2$, respectively. Some approaches have been developed to determine these parameters automatically, such as the discrepancy principle [35], generalized cross-validation [36], and the L-curve method [37]. In this paper, the parameters are determined heuristically. We validate the large range of the parameters with the method of [36]. For the different scales of the size levels of data sets, there are small changes for the optimal regularization parameters. We find that good performance can be achieved with the regularization parameters at the narrower range λ_1, $\lambda_2 \in [10^{-3}, 10^{-2}]$, $\beta \in [1, 2]$.

The optimization procedure described can be summarized as follows.

Table 1 Introduction of data set

Items	Values
Number of books	10,393
Number of users	26,387
Numbers of rating score	407,573
Sparsity	99.85%
Numbers of book introductions	10,393
Average words of the book introduction	133.7
Minimum words of the book introduction	50
Maximum words of the book introduction	2791

3.3 CNN input computing by language model

Topic model A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is. Then, we can represent a word as the probability that belongs to the topics. In this paper, the Latent Dirichlet Allocation (LDA) method [38] is used to train the topic model.

4 Experiments and discussion

4.1 Data sets

To verify the effectiveness of the proposed recommendation algorithm, we have done several offline experiments on the Book-Crossing data set [39]. The data set (Table 1) is collected by Cai-Nicolas Ziegler from the Book-Crossing virtual book community in 2004. According to the

```
Algorithm 1. Latent factor model with $L_1$-based regularization algorithm
Input: historical rating matrix $\boldsymbol{R}$, regularization parameters $\lambda_1$ and $\lambda_2$ for $\boldsymbol{U}$ and $\boldsymbol{V}$, introducing parameters $\theta_U$ and $\theta_V$
1: Initialization $\boldsymbol{U}$ and $\boldsymbol{V}$ $\boldsymbol{d}_U$, $\boldsymbol{b}_U$, $\boldsymbol{d}_V$, $\boldsymbol{b}_V$ randomly.
2: While 'stopping criterion'$(\boldsymbol{J}(\boldsymbol{U}^{k+1},\boldsymbol{V}^{k+1})-\boldsymbol{J}(\boldsymbol{U}^k,\boldsymbol{V}^k))/\boldsymbol{J}(\boldsymbol{U}^k,\boldsymbol{V}^k)$>=0.001
      fix $\boldsymbol{V}$
      update $\boldsymbol{U}^{k+1}$ by gradient-descent algorithm in Eq. (10) Eq. (11)
      update $\boldsymbol{d}_U$ $\boldsymbol{b}_U$ by Eq. (9) Eq. (12)
     fix $\boldsymbol{U}$
      update $\boldsymbol{V}^{k+1}$ by gradient-descent algorithm in Eq. (10) Eq. (11)
      update $\boldsymbol{d}_V$, $\boldsymbol{b}_V$ by Eq. (9) Eq. (12)
   end whiles
Output: matrixes $\boldsymbol{U}$ and $\boldsymbol{V}$
```

Book-Crossing data set, we supply the brief introductions of the books from Amazon

In the data set, the rating scores marked by users are between 0 and 10. The higher the score, the higher the favourability.

4.2 Metrics

In our experiments, two popular error metrics are used to evaluate the prediction accuracy: the mean absolute error (MAE) and the root mean square error (RMSE). The smaller the MAE or RMSE value becomes, the higher the accuracy. MAE and RMSE are defined as

$$MAE = \frac{1}{|R_{test}|} \sum_{R_{ij} \in R_{test}} \left| R_{ij} - R_{ij}^{*} \right|, \tag{13}$$

$$RMSE = \sqrt{\frac{1}{|R_{test}|} \sum_{R_{ij} \in R_{test}} \left(R_{ij} - R_{ij}^{*} \right)^{2}}. \tag{14}$$

To evaluating the recommendation quality, three evaluation metrics are used: precision, recall, and F1 score. Each recommendation was set to generate a list of top-N items most likely to be interesting for a given target user:

$$\text{precision} = \frac{\sum_{n=1}^{k} \frac{|TopN_n \cap I_n|}{TopN_n}}{k}, \tag{15}$$

$$\text{recall} = \frac{\sum_{n=1}^{k} \frac{|TopN_n \cap I_n|}{I_n}}{k}, \tag{16}$$

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{17}$$

where k denotes the number of users in data sets, and ***TopNn*** is the recommended items for the *n*th user. This means that the item has been evaluated by the *n*th user. The larger precision, recall, and F1 score become, the better the performance.

4.3 Comparisons and implementation

To evaluate the effectiveness of our reconstructive method, we compared our method with some classic recommendation algorithms and the state-of-the-art methods.

4.3.1 History data-based recommendation method

The classic latent factor model with L_2 regulation named *LFM*, the number of latent factors is 40.

The classic *kNN*-based collaborative filtering method named *kNN-CF*. We use correlation and cosine similarity as the similarity measurement in *kNN*, named *kNN-CF-COR* and *kNN-CF-COS*, and $k = 30$.

The entropy-based collaborative filtering (EBCF) algorithm is proposed by Piao et al. in [40]. *EBCF* proposed an entropy-based method to measure the similarity between the users and items, by that it could get great performance in the sparse data sets. In our experiments, $k = 30$.

The Bayesian probabilistic matrix factorization (*BPMF*) is proposed by Salakhutdinov and Minh et al. in [41]. *BPMF* is the state-of-the-art HDBR method which is a fully Bayesian treatment of the Probabilistic Matrix Factorization (PMF) model in which model capacity is controlled automatically by integrating over all model parameters and hyper parameters. We use 30*D* features ($D = 30$) in *BPMF*.

When the HDBR is implemented, we do both the user-based (UB) method and the item-based (IB) method in each recommendation method.

4.3.2 Content-based recommendation method

Collaborative topic regression (*CTR*) is proposed by Wang Chong and Blei David [42]. *CTR* is the state-of-the-art CBR method which utilized a probabilistic graphical model that seamlessly integrates a topic model, latent Dirichlet allocation (*LDA*), and a model-based CF method, probabilistic matrix factorization (*PMF*). In our experiments, the number of the topics is 50.

CBCNN is the proposed recommendation algorithm. *CBCNN* has the *LDA* as input, latent factor model with L_1 regulation (LFML_1R) as output and convolutional neural network as mapping function. The topic number of *LDA* is 100. The *CNN* has 1000 filters in the first layer and 5 regions pooling in the second layer. The number of latent factors *LFML₁R* is 40, the values of λ_1 and λ_2 are 0.01 and 0.0038, respectively, and β is 1.5.

At first, we used 80% of the users' data randomly as the training set. The rest (20%) of the users' data was used as the test set. For HDBRs, we performed the five-fold cross validation in test set; 80% ratings are used to

Table 2 Results of the experiments in different UB models. The smaller MAE or RMSE value becomes, the higher the accuracy

Algorithms	MAE value	RMSE value
UB-LFM	4.4119	5.9243
UB-CF-COR	4.4219	5.9425
UB-CF-COS	4.1341	4.5545
UB-EBCF	2.9676	4.7577
UB-BPMF	2.9584	3.7373
CTR	2.6582	3.6521
CBCNN	**2.6032**	**3.3841**

Bold indicates the best result in the comparative trial

Table 3 Results of the experiments in different IB models

Algorithms	MAE value	RMSE value
IB-LFM	3.5791	4.8140
IB-CF-COR	3.5486	4.7843
IB-CF-COS	3.1455	3.9768
IB-EBCF	2.8566	3.9567
IB-BPMF	2.7263	3.5816
CTR	2.6582	3.6521
CBCNN	**2.6032**	**3.3841**

Bold indicates the best result in the comparative trial

predict the rest, 20% ratings. In addition, the CBRs need to predict all the ratings in the test set, so it is a complete "cold start" problem for CBRs. In addition, in the Top-N experiment, $N = 5$, and we only choose the ten score records as I_n.

4.4 Experiment results and discussion

The comparison results of all models are shown in Tables 2 and 3; the MAE value and RMSE value measure the distance between the real ratings and predicted ratings. Based on the results shown in Tables 2 and 3, the IB methods have a better performance than the UB methods. One possible reason was that the rating records are more plentiful for the item than for the user, and most HDBR methods will benefit from this. In a general way, HDBR methods performed better in the prediction tasks compared with CBR methods. As shown in Tables 2 and 3, CBCNN has the best performance in both MAE and RMSE indexes, even though the CBCNN method is settling a completely "cold start" problem in the experiment. The result proves that CBCNN is a highly effective recommendation algorithm even comparing to the EBCF and BPMF. Even though LFM is one of the components in CBCNN, CBCNN yielded better results than all LFM methods. This indicates that the text information about the books is really helpful to make the recommendation in this experiment. Comparing with the state-of-the-art CBR method CTR, CBCNN got better result. It indicates that the CNN model has more capability to extract the features in the text information than the topic model. In addition, we could infer that CBCNN will work well

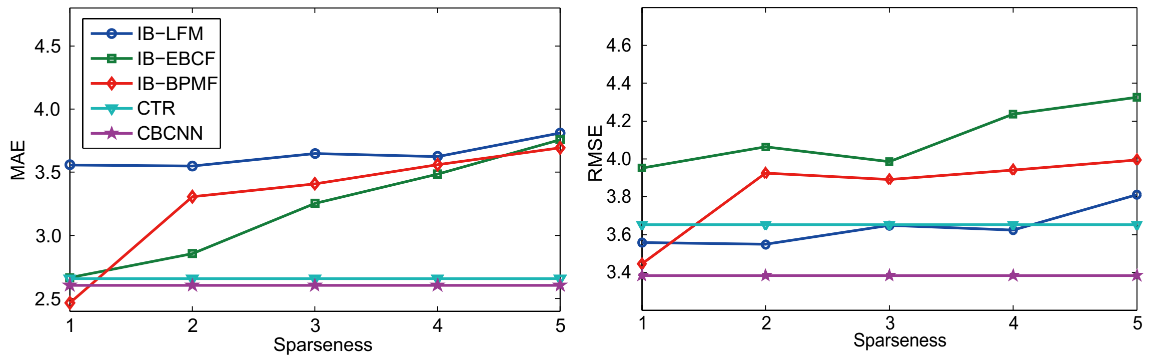

Fig. 4 MAE and RMSE with different sparseness on IB-LFM, IB-EBCF, IB-BPMF, CTR, and CBCNN

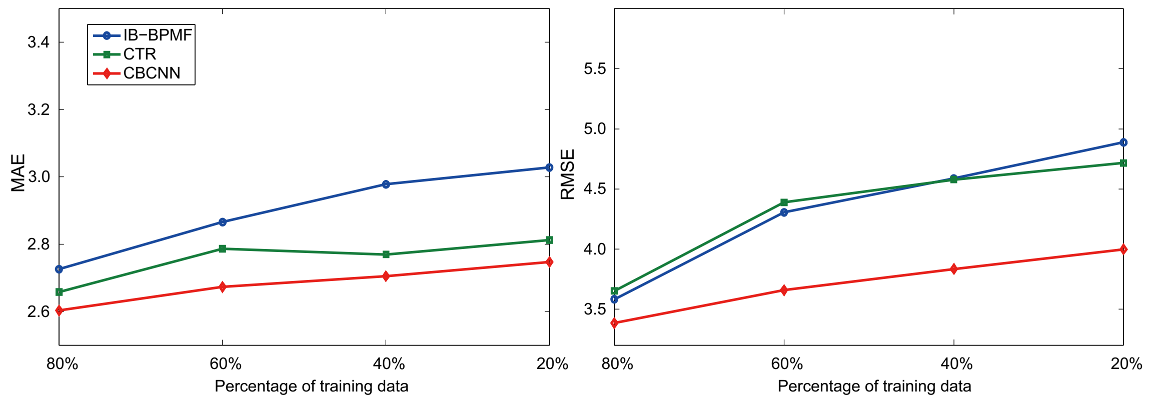

Fig. 5 MAE and RMSE with different percentages of training data on IB-BPMF, CTR, and CBCNN

for learning resources recommendation in the e-learning system.

To explore the relevance between performance of the recommendation algorithms and sparseness of the training data, the experiments are set. From the result in Figs. 4 and 5, we can observe how the sparseness of data and the percentage of training data affect the performances of different recommendation algorithms. The higher the sparseness value becomes, the less data records it represents. It is clear that IB-BPMF achieved better results when sparseness =1; however, the performance of IB-BPMF is quite unstable. The performance of comparison methods usually declines with the increase in sparseness. However, the performance of CBCNN is not related to the sparseness. CBCNN does not require the history data of items to make its prediction. In this sense, CBCNN is more robust and showed a better performance than the other methods. With the percentage of training data decreasing, the CBR methods are more robust than the HDBR methods. In addition, the CBCNN achieves the best performance among the comparison methods.

In the Top-N ($N=5$) experiments, Precision, Recall, and F1 score (the larger the precision, recall, and F1 score is, the better the performance) are used to evaluate the recommendation results. In the Top-N experiments, we only selected the ten score records as the recommendation target. Generally speaking, the kNN-based recommendation methods have an advantage over other methods because of the difference in objective function. Figure 6 shows that kNN-based models have done well in the experiments. In fact, CBCNN has the best performance among the non-kNN models. As for the experiments on new resources, the HDBRs are no longer functional, while CBCNN still managed to achieve a decent result. Compared to the CTR, the state-of-the-art CBR model, CBCNN got better result in three adoptive indexes. Therefore, CBCNN, as a CBR model, can alleviate the "cold problem" and still provide decent recommendation results.

To reveal the relevance between model performance and the structure of CNN, we add some comparison experiments. In the experiments, we have discussed how to choose filter structures, non-linear functions, and number of pooling regions in CNN. Three different non-linear functions (rectified linear units, tanh function, and sigmoid function), and two different filter structures (convolutional and non-convolutional) are chosen for the comparison

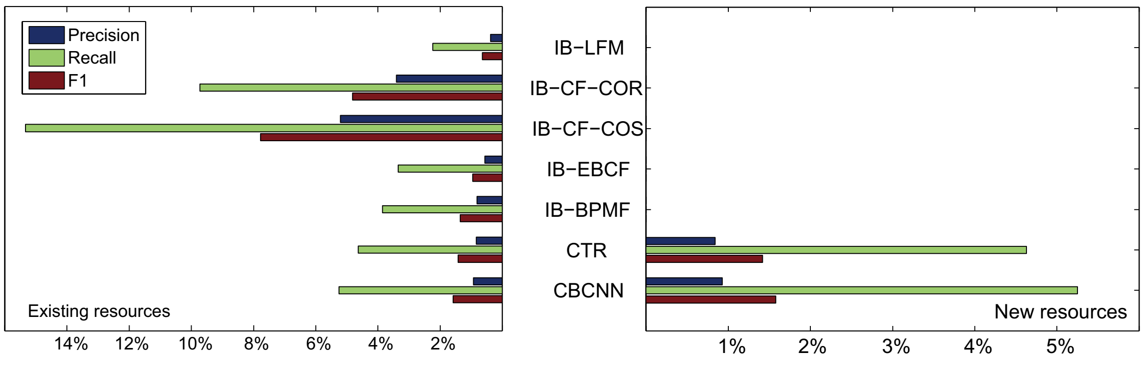

Fig. 6 Precision, recall, and F1 score versus different recommendation algorithms for the Book-Crossing data set [39]. IB means item-based algorithm. The *left* experiments deals with the Top-N recommendation on existing resources; the *right* is on the new resources

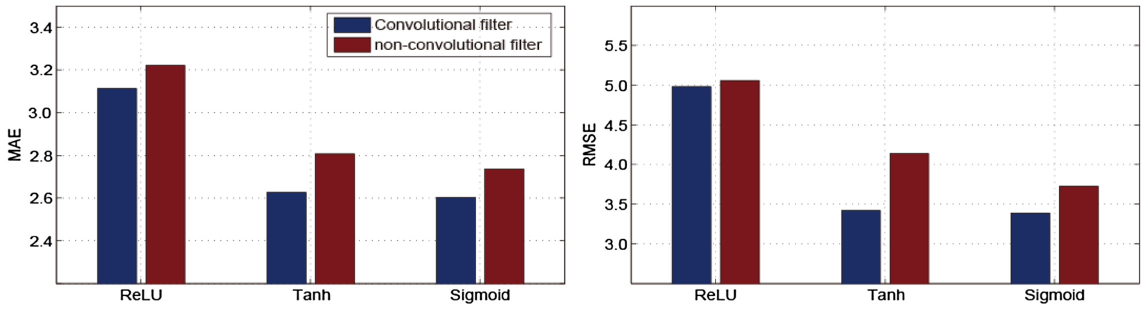

Fig. 7 MAE and RMSE versus three different filter structures (convolutional and non-convolutional) and two different non-linear functions (rectified linear units (ReLU), Tanh and Sigmoid) in CNN structure

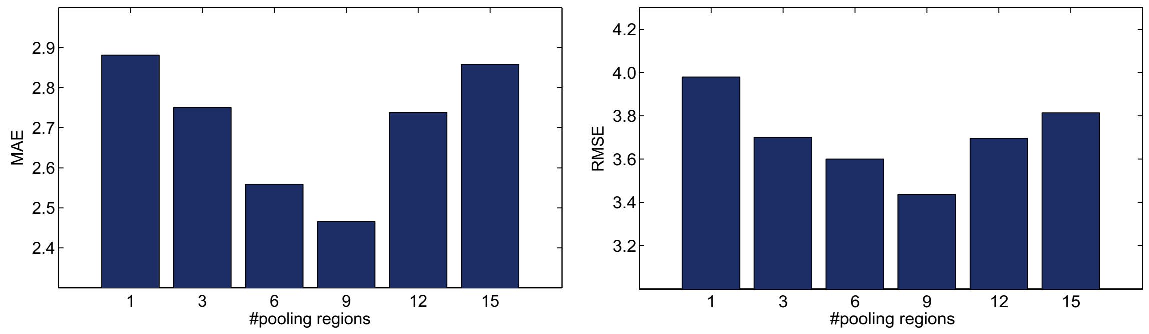

Fig. 8 MAE and RMSE versus different numbers of pooling regions in CNN structure

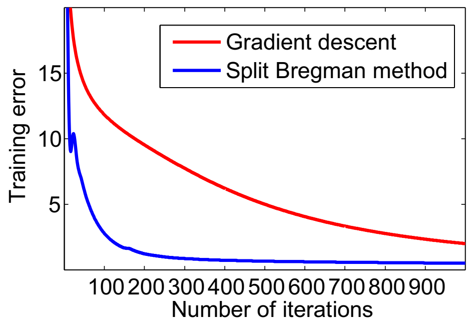

Fig. 9 Training error versus number of iterations in different optimization methods (gradient-descent and split Bregman methods)

Table 4 Complexity analyse of different optimization methods

Algorithms	Complexity of each iteration [a]
Gradient descent	$O(2\times N^R\times 2S)$
Split Bregman method	$O(2\times N^R\times 2S+2\times N^U\times S+2\times N^I\times S)$

[a] N^R denotes the number of rating scores, N^R and N^I are the size of users and items respective. S is the number of latent factors. The $2\times N^U\times S$ and $2\times N^I\times S$ are the cost of updating the $\boldsymbol{d}_U$, $\boldsymbol{b}_U$, $\boldsymbol{d}_V$, $\boldsymbol{b}_V$ in split Bregman method from Eq. (9) in each iteration. $2\times N^R\times 2S$ is the cost of gradient descent in one iteration

Table 5 Time cost of different optimization methods

Algorithms	Time cost (s)
Gradient descent	1231.435
Split Bregman method	**726.041**

Bold indicates the best result in the comparative trial

experiment. The experimental result is shown in Fig. 7. We find that sigmoid function achieves the smallest MAE and RMSE values. Thus, the sigmoid function is the best nonlinear functions in those experiments. In Fig. 7, the values of blue bar are smaller than those of red bars. It indicates that the convolutional filter structure outperforms the non-convolutional filter structure among three non-linear functions. It demonstrates that CNN will obtain a great performance in the text information extraction when the convolutional filter structure and sigmoid function are chosen simultaneously. To process the long-text information, the pooling regions is introduced. To analyse the effect of pool regions number, we set the experiment that the number of pooling regions equal to $\{1, 3, 6, 9, 12, 15\}$. The experimental results are shown in Fig. 8. We can observe that the multi-pooling regions have achieved much greater performance comparing with the general pooling operation (#pooling regions =1). We may conclude that CNN can extract features effectively with the multi-pooling regions setting in long-text information.

To optimize the L_1-norm-based model, the split Bregman iteration method is introduced. In addition, from the results shown in Fig. 9, it is clear that the split Bregman iteration method has a much faster convergence rate than the gradient-descent method. Although the complexity of each iteration in Table 4 shows that split Bregman iteration method has little higher complexity than the gradient-descent method in one iteration (N^R and N^I are much smaller than N^R). However, comprehensively considered the convergence rate and the complexity of each iteration, the split Bregman iteration method is much more efficient than the gradient-descent method which is shown in Table 5. The learning speed of the recommendation methods is a core point for the recommendation system. The split Bregman iteration method will not only save a lot of computational resources, but will also provide the chance to update the model more frequently compared with a real-time recommendation system. The CPU time is given in seconds, and the experiment was performed on a desktop PC equipped with an Intel Core i5-3470 CPU and 8 GB memory. The total computing time in MATLAB 2012b is provided in Table 5.

5 Conclusion

In this paper, we have proposed a content-based learning resource recommendation algorithm based on CNN. The CNN can be used to predict the latent factors from the text information. To train the CNN, its input and output should be solved first. For the input, language model is used. For the output, we proposed the latent factor model, which is regularized by L_1-norm. Furthermore, the split Bregman iteration method is introduced to solve the model. The major novelty of the proposed recommendation algorithm is that a CNN is constructed to make personalized recommendations and achieved a superior result. The proposed algorithm is verified on a public data set. Results indicate that our recommendation algorithm for recommending new and unpopular learning resources is feasible. Moreover, the split Bregman iteration method can greatly improve the training efficiency. We believe that the convolutional neural network-based model will play a crucial role in e-learning systems or intelligent tutoring systems in the future. Although the application considered here is focused on learning resources recommendation, the method is more generally applicable to news recommendation, etc.

Compliance with ethical standards

Funding The authors thank the editor and anonymous reviewers for their valuable suggestions. This research was partially funded by the specific funding for education science research by self-determined research funds of CCNU from the colleges' basic research and operation of MOE (CCNU16JYKX031 and CCNU16JYKX027), the National Natural Science Foundation of China under Grant (No. 61505064), the Project of the Program for National Key Technology Research and Development Program (2013BAH72B01, 2013BAH18F02, and 2015BAH33F02), and the Project of the Program for National Key Technology Research and Development Program (2014BAH22F01 and 2015BAK07B03).

References

1. Wei, S., Zheng, X., Chen, D., Chen, C.: A hybrid approach for movie recommendation via tags and ratings. Electronic Commerce Research and Applications (2016)
2. Moreno, M.N., Segrera, S., López, V.F., Muñoz, M.D., Sánchez, Á.L.: Web mining based framework for solving usual problems in recommender systems. A case study for movies′ recommendation. Neurocomputing **176**, 72-80 (2016)
3. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: proceeding International Joint Conference on Artificial Intelligence, pp 1137–1143 (1995)
4. Mao, K., Chen, G., Hu, Y., Zhang, L.: Music recommendation using graph based quality model. Signal Process. **120**, 806–813 (2016)
5. Horsburgh, B., Craw, S., Massie, S.: Learning pseudo-tags to augment sparse tagging in hybrid music recommender systems. Artif. Intell. **219**, 25–39 (2015)
6. Wang, Y., Shang, W.: Personalized news recommendation based on consumers' click behavior. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on, 2015. IEEE, pp 634–638
7. Shi, B., Ifrim, G.: Hurley N Learning-to-Rank for Real-Time High-Precision Hashtag Recommendation for Streaming News. In: Proceedings of the 25th International Conference on World Wide Web, 2016. International World Wide Web Conferences Steering Committee, pp 1191–1202
8. Zhao, X., Yuan, J., Wang, M., Li, G., Hong, R., Li, Z., Chua, T.-S.: Video recommendation over multiple information sources. Multimed. System **19**(1), 3–15 (2013)
9. Pyo, S., Kim, E., Kim, M.: Automatic and personalized recommendation of TV program contents using sequential pattern mining for smart TV user interaction. Multimed. System **19**(6), 527–542 (2013)
10. Kaššák, O., Kompan, M., Bieliková, M.: Personalized hybrid recommendation for group of users: top-N multimedia recommender. Inf. Process. Manag. **52**(3), 459–477 (2016)
11. Zhao, W., Wu, R., Liu, H.: Paper recommendation based on the knowledge gap between a researcher's background knowledge and research target. Inf. Process. Manag. **52**(5), 976–988 (2016)
12. Polatidis, N., Georgiadis, C.K.: A multi-level collaborative filtering method that improves recommendations. Expert System Appl. **48**, 100–110 (2016)
13. Alhamid, M.F., Rawashdeh, M., Dong, H., Hossain, M.A., Alelaiwi, A., El Saddik, A.: RecAm: a collaborative context-aware framework for multimedia recommendations in an ambient intelligence environment. Multimed. Syst. **22**(5), 587–601 (2016)
14. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Recommender systems handbook, pp 77–118. Springer, Heidelberg (2015)
15. Salah, A., Rogovschi, N., Nadif, M.: A dynamic collaborative filtering system via a weighted clustering approach. Neurocomputing **175**, 206–215 (2016)
16. Pan, W., Xia, S., Liu, Z., Peng, X., Ming, Z.: Mixed factorization for collaborative recommendation with heterogeneous explicit feedbacks. Inf. Sci. **332**, 84–93 (2016)
17. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, 2007. IEEE, pp 43–52
18. Ignatov, D.I., Nikolenko, S.I., Abaev, T., Poelmans, J.: Online recommender system for radio station hosting based on information fusion and adaptive tag-aware profiling. Expert System Appl. **55**, 546–558 (2016)
19. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: Learning semantic representations using convolutional neural networks for web search. In: Proceedings of the companion publication of the 23rd international conference on World wide web companion, 2014. International World Wide Web Conferences Steering Committee, pp 373–374
20. Pandey, S., Khanna, P., Yokota, H.: A semantics and image retrieval system for hierarchical image databases. Inf. Process. Manag. **52**(4), 571–591 (2016)
21. Ding, C., Tao, D.: Robust face recognition via multimodal deep face representation. Multimed. IEEE Trans. 17(11), 2049–2058 (2015)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105 (2012)

23. Graves, A., Mohamed, A.R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Acoustics, Speech and Signal Processing, IEEE International Conference on, 2013. pp 6645–6649
24. Tian, L., Fan, C., Ming, Y.: Multiple scales combined principle component analysis deep learning network for face recognition. J. Electron. Imaging. **25**(2), 023025–023025 (2016)
25. Byeon, Y.-H., Pan, S.-B., Moh, S.-M., Kwak, K.-C.: A Surveillance System Using CNN for Face Recognition with Object, Human and Face Detection. In: Information Science and Applications (ICISA) 2016. Springer, pp 975–984 (2016)
26. Ye, X., Wang, L., Xing, H., Huang, L.: Denoising hybrid noises in image with stacked autoencoder. In: Information and Automation, 2015 IEEE International Conference on, 2015. IEEE, pp 2720–2724
27. Huang, W.-B., Sun, F.-C.: Building feature space of extreme learning machine with sparse denoising stacked-autoencoder. Neurocomputing **174**, 60–71 (2016)
28. Iwata, T., Hirao, T., Ueda, N.: Probabilistic latent variable models for unsupervised many-to-many object matching. Inf. Process. Manag. **52**(4), 682–697 (2016)
29. Xue, S., Jiang, H., Dai, L., Liu, Q.: Speaker adaptation of hybrid NN/HMM model for speech recognition based on singular value decomposition. J Signal Process. Systems. **82**(2), 175–185 (2016)
30. Johnson, R, Zhang, T.: Semi-supervised convolutional neural networks for text categorization via region embedding. In: Advances in Neural Information Processing Systems, 2015. pp 919–927
31. Wikipedia Topic model. (2016)
32. Qin, P., Xu, W., Guo, J.: An empirical convolutional neural network approach for semantic relation classification. Neurocomputing **190**, 1–9 (2016)
33. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, 2013. pp 3111–3119
34. Goldstein, T., Osher, S.: The split bregman method for L1-regularized problems. SIAM J Imag Sci. **2**(2), 323–343 (2009). doi:10.1137/080725891
35. Engl, H.W., Ramlau, R.: Regularization of inverse problems. Kluwer Academic Publishers, New York (2000)
36. Devijver, P.A., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice/hall International, New Jersey (1982)
37. Hansen, P.C., O'Leary, D.P.: The use of the L-curve in the regularization of discrete ill-posed problems. Siam J. Sci. Computing. **14**(6), 1487–1503 (1993)
38. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J Mach Learn Res **3**, 993–1022 (2003)
39. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web, 2005. ACM, pp 22–32
40. Piao, C.H., Zhao, J., Zheng, L.J.: Research on entropy-based collaborative filtering algorithm and personalized recommendation in e-commerce. Service Oriented Comput Appl **3**(2), 147–157 (2009)
41. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. Icml'08 Proceedings of International Conference on Machine Learning pp 880–887
42. Wang C, Blei DM Collaborative topic modeling for recommending scientific articles. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, Ca, Usa, August, 2011. pp 448–456 (2012)