Expert Systems With Applications 186 (2021) 115752

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Shared-view and specific-view information extraction for recommendation

Huiting Liu [a,b,*], Jindou Zhao [a,b], Peipei Li [c], Peng Zhao [a,b], Xindong Wu [d,e]

[a] Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University, Hefei 230601, Anhui, China
[b] School of Computer Science and Technology, Anhui University, Hefei 230601, Anhui, China
[c] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, Anhui, China
[d] Research Institute of Big Knowledge, Hefei University of Technology, Hefei 230601, Anhui, China
[e] Mininglamp Academy of Sciences, Mininglamp Technology, Beijing 100084, China

ARTICLE INFO

Keywords:
Recommender system
Dual-view learning
Review analysis

ABSTRACT

In various recommender systems, ratings and reviews are the main information to show user preferences. However, recommendation models that only use ratings, such as collaborative filtering, are vulnerable to data sparsity. And models only using review information will also suffer from the sparsity of reviews. On one hand, most ratings and reviews are interrelated and complementary, reviews may explain why a user gives a high or low rating to an item. On the other hand, ratings and reviews are numerical and textual information, respectively, and they reflect the preference of the user from a coarse-grained level and a fine-grained level A user may comment positively about some aspects of an item, even he gives a very low score to this item. There are specific information among each of them because of their heterogeneity. Therefore, it is possible to learn more accurate representation of users and items by effectively integrating ratings and text reviews from different views, that is, shared-view and specific-view. In this paper, we propose a Shared-view and Specific-view Information extraction model for Recommendation (SSIR), which integrates the information from reviews and interaction matrix to predict ratings Our model has two key components, including shared-view information extraction and specific-view exploitation. From the perspective of shared-view, SSIR jointly minimizes the loss of confusion adversarial and rating prediction loss to extract the shared information from reviews and user–item interaction matrix. For the specific-view part, SSIR applies orthogonal constraints on shared-view and specific-view modules to extract the discriminative features from reviews and interaction data. We fuse the features extracted from these two views to predict the final ratings. In addition, we use auxiliary reviews to deal with the sparsity problem of reviews. Experimental results on eight datasets show the effectiveness and robustness of our method, which could adapt to the recommendation scenarios with fewer reviews and ratings.

1. Introduction

With the explosive growth of information, recommendation systems are becoming more and more important in alleviating information overload, and have been widely used in many websites and applications. Recommendation systems help customers make decisions by displaying candidate products that they may be interested in, according to their preferences and past purchasing behaviors. Many important methods widely used in recommendation systems are collaborative filtering (CF) based techniques (Berg, Kipf, & Welling, 2017; Panagiotakis, Papadakis, & Fragopoulou, 2021), which use historical records (such as ratings and clicks) to properly model user preferences and product features. These methods assume that users who have similar choices in the past will be interested in similar products in the future. Many CF-based methods decompose the rating matrix into two latent feature matrices of users and items according to Matrix Factorization (MF), and then predict the missing rating of the user to the item with the inner product of user and item latent feature vectors. Those methods can find some common factors, which may be the potential reasons why users like the item. For example, when recommending a book to users, these factors can be the author or genre of the book that users express concern about. Besides finding these hidden factors, MF technology can also show how each item meets each factor and how important they are to each user. But the performance of those CF-based recommendation systems decreases significantly, when there is little user–item interaction information in the recommendation system. In this case, the neighbor-based methods are introduced to tackle the problem of data sparsity, which use various

* Corresponding author at: School of Computer Science and Technology, Anhui University, Hefei 230601, Anhui, China.
E-mail addresses: htliu@ahu.edu.cn (H. Liu), jindou.zhao@foxmail.com (J. Zhao), peipeili@hfut.edu.cn (P. Li), zhaopeng_ad@163.com (P. Zhao), xwu@hfut.edu.cn (X. Wu).

https://doi.org/10.1016/j.eswa.2021.115752
Received 6 April 2021; Received in revised form 31 July 2021; Accepted 8 August 2021
Available online 5 September 2021

relationships between entities, such as items or users, to make the recommendation more accurate.

In social media sites and e-commerce systems, some users further express their preferences by writing reviews after they have consumed the items. These reviews contain rich information, which can be exploited to alleviate the problem of data sparsity. Some studies (Dong, Ni, Cheng, Chen, & Melo, 2020; Jakob, Muller, Weber, & Gurevych, 2009; Zheng, Noroozi, & Yu, 2017) show that the methods taking reviews into consideration usually outperform the CF-based methods only considering interaction records. The user and item representations of these review-based methods are extracted from reviews. For example, Deep Cooperative Neural Networks (DeepCoNN) (Zheng et al., 2017) obtains the embeddings of user preferences and item properties from all reviews of this user and item by using two parallel neural networks. However, some reviews contain noise and some contain too few words to extract effective information. The performance of those recommendation systems will be seriously damaged when these useless reviews are introduced into the model.

Ratings and reviews are two different channels of information to represent the preferences of users to items. For example, in movie recommendation, the preferences of users can often be represented by ratings and comments to movies. The rating may reflect how much the user is interested in one movie. While the reviews will explain the reasons from several aspects, such as the performance of actors, the style of the movie, and the plot. In this paper, we seek to seamlessly unify ratings and reviews to make recommendation.

There are two examples in Fig. 1. The first user gave the highest rating, and the reviews also showed that he was satisfied with the CD to some extent. Since reviews can express the opinions of the user from several aspects, we can find the user also pointed out some drawbacks of the CD even though he gave the highest score. The second user gave a very low rating, and he talked about the shortcomings of the disc in his reviews. But it can be seen from the reviews that he did not completely deny the disc. From Fig. 1, we can see that most ratings and the reviews are similar with each other from the emotion view, and may complement each other. We can extract shared information from the interaction data and reviews. However, ratings are numerical information while reviews are textual information, and they reflect the preference of the user from a coarse-grained level and a fine-grained level, respectively. That is to say, each of them contains their own specific contribution to the final rating prediction, and the heterogeneity between them cannot be ignored. The main challenge is how to integrate the two types of heterogeneities in an efficient and general way.

In fact, some works have combined reviews and the interaction data to extract more effective representations of users and items, and have proved that incorporating both of them into recommendation systems can further improve the recommendation performance. Wu, Quan, Li, Wang, and Zheng (2019) propose a context-aware user–item representation learning model (CARL), which integrates interaction information and text reviews to make rating prediction. However, CARL encounters a problem. When trying to combine interaction data and reviews information, CARL firstly maps each kind of information into its own subspace, and then concatenates the representation of each subspace to predict ratings. This process is always carried out in an independent way, and the complementary and inherent relation between ratings and reviews has been ignored. Hybrid Recommendation model to learn Deep Representation (HRDR) (Liu, et al., 2020) takes their relationship into account, and the representation learned from ratings is used as an attention query vector to select shared information from reviews. But the heterogeneity between reviews and ratings makes HRDR unable to obtain some useful reviews via rating-based selection, such as the sentences marked in blue in the first example and those marked in red in the second example in Fig. 1, and those reviews also make specific contributions to enhance rating prediction.

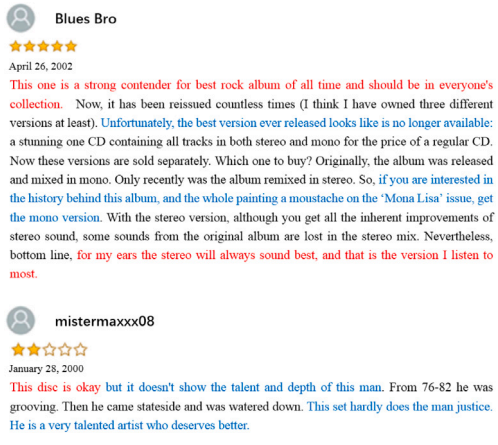


Fig. 1. Examples of reviews written by two users.

In order to address the above issues, we propose a Dual-View Learning model with Shared-view and Specific-view Information extraction for Recommendation (SSIR), which integrates the information from reviews and interaction matrix to predict ratings. The model SSIR has two key components, they are shared subspace information extraction and specific-view exploitation parts, separately. From the perspective of shared-view, SSIR jointly minimizes the loss of confusion adversarial and rating prediction loss to extract the shared information from reviews and user–item interaction matrix. The distributions of shared features extracted from ratings and reviews will become as similar as possible when we minimize the loss of confusion adversarial. For the specific-view part, the useful specific information is hard to define. However, we can extract it by eliminating the shared information from original information. This can be achieved by applying orthogonal constraints on shared-view and specific-view modules, to guarantee there is no correlation between these two modules. Then, we concatenate the shared-view and the specific-view representations to gain the final embedding for the user and item. And the final embedding is fed into a Factorization Machine (FM) for rating prediction. We have carried out comprehensive experiments on eight real-world datasets. The experimental results show that SSIR is superior to several state-of-the-art algorithms. The results also validate that the Shared-view and Specific-view Information can complement each other, and fusion of them will make a better prediction performance. The major contributions of our work are summarized as follows:

- We put forward a Dual-View recommendation model (SSIR), which extracts information from the shared view and their specific views, to fully mine the information in the reviews and the interaction matrix.
- In the shared view module, we utilize the confusion adversarial loss to extract the shared features from different kinds of information, such as reviews and the interaction data. As far as we know, we are the first to apply this idea in the recommendation scenario. We also enforce orthogonal constraints on shared and specific subspaces to extract the information from specific views. Finally, shared-view and specific-view information are synergized to learn the final embeddings of users or items for rating prediction.
- We empirically conduct extensive experiments on eight real-world datasets. Experimental results demonstrate that our model SSIR outperforms several state-of-the-art alternatives. Moreover, some ablation studies verify the effectiveness of our model for recommendation and the utility of our model in alleviating data sparsity.

2. Related work

In this section, we briefly review the related work from two different perspectives. The first category discusses the relevant literatures on collaborative filtering. The second category covers advanced studies about reviews-based recommendation.

2.1. Collaborative filtering

Collaborative filtering has achieved tremendous success for personalized recommendations in academic as well as industrial research areas. Traditional CF-based methods mainly exploit the rating matrix, which aggregates information from previous user–item interactions, to make recommendations. Most CF-based methods use MF for recommendation (Hammou, Lahcen, & Mouline, 2019). MF maps users and items into separate latent vectors with the same dimension and then estimates missing values of the user–item rating matrix using the inner product of their factors. But CF-based methods are faced with the problem of data sparsity, when there is little user–item interaction data in the recommendation system. A tremendous amount of work has been done to address the issue of data sparsity by using relations between users or items, and the neighbor-based method is one of them. SVD++ (Koren, 2008) was a well-known neighbor-based model. It considered not only the interaction between users and items, but also the user's preference of other items, when predicting the user's rating of a certain item. It was superior to the traditional MF-based methods. He, He, Song et al. (2018) found that optimizing matrix factorization with Bayesian Personalized Ranking (BPR) made a recommender model not robust enough. To improve the robustness of a recommender model, they proposed a new optimization framework, namely Adversarial Personalized Ranking (APR). APR enhanced the pairwise ranking method BPR by performing adversarial training. Through the application of this adversarial matrix factorization, APR achieved good performance for item recommendation. Zhang, Liu, Wang, and Li (2021) proposed a Multi-criteria recommender system, which could effectively extend the application scope of social recommendation model by an implicit social relationship inference technique. However, all the above models used the dot product(DP) operation to make predictions. The DP operation only allows linear combination of latent features, but never considers their higher-order interaction. This strong limitation hinders the performance of traditional MF-based methods.

With the great success of neural network, some researchers applied various neural networks to recommendation systems, such as autoencoders (Liang, Krishnan, Hoffman, & Jebara, 2018; Wu, Dubois, Zheng, & Ester, 2016), multilayer perceptron (MLP) (He, Liao, Zhang, Nie, & Chua, 2017; Xue, Dai, Zhang, Huang, & Chen, 2017), memory network (Chen, Zhang, Liu, & Ma, 2018; Tay, Tuan, & Hui, 2018), CNN (Tang & Wang, 2018), attention architectures (Chen, et al., 2017; He, He, Du, & Chua, 2018) and sequential neural network (Zou, Xia, Gu, Zhao, & Yin, 2020). Wu et al. (2016) used an automatic encoder framework to learn from noisy inputs and made the top-n recommendation. He et al. (2017) utilized a MLP to learn the nonlinear interaction of latent factors of users and items. Chen, Xu, Zhang, Tang, and Zha (2018) leveraged an external memory matrix to explicitly store and update the users' historical interaction data, and improved the performance of the model. Tang and Wang (2018) captured the point-level and union-level sequential patterns via horizontal and vertical convolution filters for sequential recommendation. Since different historical items would have different importance to profile the preference of users, Chen, et al. (2017) and He, He, Song et al. (2018) introduced attention mechanisms to learn the varying importance of the historical items. Zou et al. (2020) put forward the exploration policy, which was represented with a sequential neural network, to balance between the recommendation and the learning of preference of users. In order to improve the recommendation accuracy, Panagiotakis, Papadakis, Papagrigoriou, and Fragopoulou (2021) proposed a parameter free, second stage that could be executed after any model-based recommendation system. Although user preferences and item features can be learned from user–item interaction data, the performance of these CF-based recommendation systems decreases significantly, when there is little user–item interaction information in the recommendation system.

2.2. Rating prediction from reviews

Reviews is a kind of auxiliary information that can be easily obtained in many recommendation systems. Therefore, using reviews to address the limitations of the CF-based methods has been widely studied in the past few years (Chen et al., 2020; Guan et al., 2019; Li, Zhou, Xu, Liu, & Xiong, 2020; Zhang et al., 2020).

Jakob et al. (2009) was the pioneer work that improved the accuracy of rating prediction by extracting opinions from reviews. They found that reviews are closely related to product information, such as price and service quality, which can be used for rating prediction. After that, some researchers obtained latent topic factors from reviews by using topic modeling techniques. Wang and Blei (2011) and McAuley and Leskovec (2013) used a technique similar to LDA to mine latent topics from reviews. Diao et al. (2014) employed CF and a topic based probabilistic model to reveal aspects and emotions of users and items. These methods are superior to models that only exploit user item interaction data. However, the above methods use bag-of-words (BOW) model to represent reviews, ignoring local context information and the orders of words. In this case, a great quantity of useful information is lost.

In order to overcome this limitation, many deep neural networks were used to deal with reviews (Sun, Wu, Zhang, Fu, & Wang, 2020; Wang, Xia, Du, Chen, & Gang, 2021; Xia, et al., 2019). CNN based model was a common one of them. Kim, Park, Oh, Lee, and Yu (2016) considered both word orders and local context, and utilized CNN to learn more comprehensive latent semantic representation from reviews. DeepCoNN transformed all user and item reviews into two long documents respectively, and used CNN to jointly model user preferences and item features. Then, DeepCoNN concatenated the latent vectors of users and items, and input the vectors into FM to make the rating prediction. Catherine and Cohen (2017) added a transformation layer to extend DeepCoNN. It can simulate the latent representation of target reviews, which is not available during testing, and make DeepCoNN predict more accurate ratings. Seo, Huang, Yang, and Liu (2017) made use of global and local attention to focus CNN on more important reviews rather than all reviews, and the prediction accuracy of their model was superior to that of DeepCoNN. Later, Chen, Zhang, et al. (2018) introduced a review-level attention to learn the useful representation of reviews. Wu, Quan, Li, and Ji (2018) found that reviews also had the problem of data sparsity, and exploited reviews written by a user's like-minded neighbors to solve this problem. Although these existing review-based recommendation systems have improved the performance of recommendation, they ignore the complementarities between the interaction data and reviews, which can improve the quality of recommendation.

In order to combine reviews and the interaction data to extract more effective representations of users and items, HRDR designed a network based on review-level attention, which selected more useful reviews by integrating rating-based features with this attention network, to learn more effective users and items representations. Liu, Wang, Xu, Peng, and Jiao (2020) extended HRDR by adding an additional reviews extraction module, but their attention mechanism was still similar to HRDR. Based on the latent features and the interaction data of a user–item pair, CARL learned their context-aware representations. These methods mapped each information into its own subspace when they tried to combine interaction information with reviews information, and then concatenated the representation of each subspace to predict ratings. This process was always carried out in an independent way or with an attention network. They could not fully take advantage of

Table 1
Notations and descriptions.

Symbols	Definitions and Descriptions
N	the number of users
M	the number of items
T	the number of convolution filters
t	the window size of convolution filters
x_u	the one-hot encoding of user u
p_u	the representation of user u extracted from ratings
$\mathbf{D}_u$	word matrix of user u
h_u	the representation of user u extracted from reviews
p_{us}	the shared representation of user u extracted from ratings
h_{usaux}	the shared representation of user u extracted from auxiliary reviews
h_{us}	the shared representation of user u extracted from reviews
s_u	the shared representation of user u extracted from reviews and ratings
h_{uspe}	the specific representation of user u extracted from reviews
p_{uspe}	the specific representation of user u extracted from ratings
y_i	the one-hot encoding of item i
q_i	the representation of item i extracted from ratings
h_i	the representation of item i extracted from reviews
q_{is}	the shared representation of item i extracted from ratings
h_{isaux}	the shared representation of item i extracted from auxiliary reviews
h_{is}	The shared representation of item i extracted from reviews
s_i	the shared representation of item i extracted from reviews and ratings
h_{ispe}	the specific representation of item i extracted from reviews
p_{ispe}	the specific representation of item i extracted from ratings

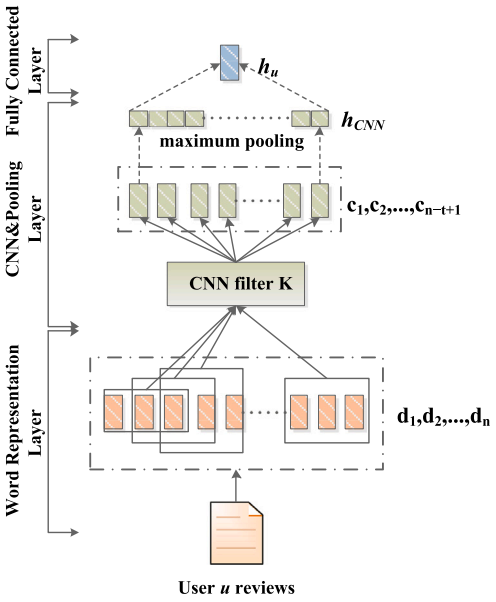


Fig. 2. The CNN Text Processor architecture.

the discriminative information between text reviews and the numerical ratings, and were unable to extract the discriminative information from them.

In sum, text reviews and the numerical ratings are the two types of heterogeneities, to integrate them in an efficient and general way, we need take advantage of both the shared information and the discriminative information between them. Here, we propose a Dual-View recommendation system, which applies shared-view and specific-view information extraction modules to learn shared and specific embeddings from user–item rating data and text reviews, to make recommendation.

3. Preliminaries

We first give the formulation of the problem in Section 3.1. Then we introduce how to initialize ID embeddings of users and items in Section 3.2. In Section 3.3, we shortly recapitulate the CNN text processor.

3.1. Problem formulation

In this paper, we use $\chi = \left\{U, I, R, \widehat{D}\right\}$ to denote the training corpus, where $U \in R^N$ and $I \in R^M$ are the sets of users and items respectively. $R \in \mathbb{R}^{N\times M}$ is the rating matrix, and $\widehat{D}$ is the set of review documents. Each $r_{u,i} \in \{1, r\}$ in matrix $\mathbb{R}$ indicates how much user u prefers to item i.

$$r_{ui} = \begin{cases} r, & \text{the rating given by user } u \text{ to item } i \text{ and } 1 \le r \le 5 \\ 0, & \text{there is no interaction between user } u \text{ and item } i \end{cases}$$

The goal of our recommendation model is to predict the u's rating for item i, that is, $\widehat{r}_{u,i}$. The mathematical symbols used in this paper are listed in Table 1.

3.2. Rating-based encoder for User/Item

The rating-based encoder will map each user and item into a dense vector by using MF-based CF. Each user–item interaction contains a user ID u and an item ID i. We adopt the one-hot encoding to encode the user ID u:

$$x_u = 1 - hot\,(u) \tag{1}$$

where $x_u \in \{0, 1\}^N$ is the one-hot encoding of user u. Then we obtain the latent representation of user u with an embedding layer below:

$$p_u = Px_u \tag{2}$$

where P is a randomly initialized embedding matrix of users. We can use a similar method to get the rating-based latent representation q_i for an item i.

3.3. CNN text processor

We use the same method as the model DeepCoNN to process texts. Fig. 2 illustrates the detailed network architecture of our CNN text processor. It processes texts as follows:

Word Representation Layer: We combine all reviews written by user u into a single document D_u, which contains n words. By utilizing a word embedding matrix $E \in R^n$, D_u can be expressed as a embedding matrix $[d_1, d_2, \ldots, d_n]$, where d_j is the jth word embedding of D_u. Word embedding matrix E can be initialized by word2vec. In this way, the word embedding matrix D_u will preserve the word order, which is superior to BOW representation.

CNN&Pooling Layer: Then we put the word embedding matrix D_u into the CNN&Pooling layer, which consists of a convolutional neural network with pooling operation. In convolutional neural network, we apply T neurons over embedding matrix D_u to extract contextual features of user u. Each neuron j uses a filter $K_j \in \mathbb{R}^{c\times t}$ with a sliding window of size t to perform convolution operation as follows:

$$c_j = \sigma\left(D_u * K_j + b_j\right) \tag{3}$$

where σ is a nonlinear activation function such as $Relu$, symbol * is the convolution operator, and b_j is the bias.

After gaining the embedding c_j produced by filter K_j, we employ a pooling operation, such as maximum pooling, over this embedding. Maximum pooling uses the maximum value of a specific kernel to represent the corresponding characteristics. And the maximum pooling operation will get the feature with the maximum value of each feature map. This aggregation scheme can handle texts of different lengths. After the pooling operation, the feature mapping results are reduced to a vector.

$$o_j = max\left\{c_1, c_2, \ldots, c_{n-t+1}\right\} \tag{4}$$

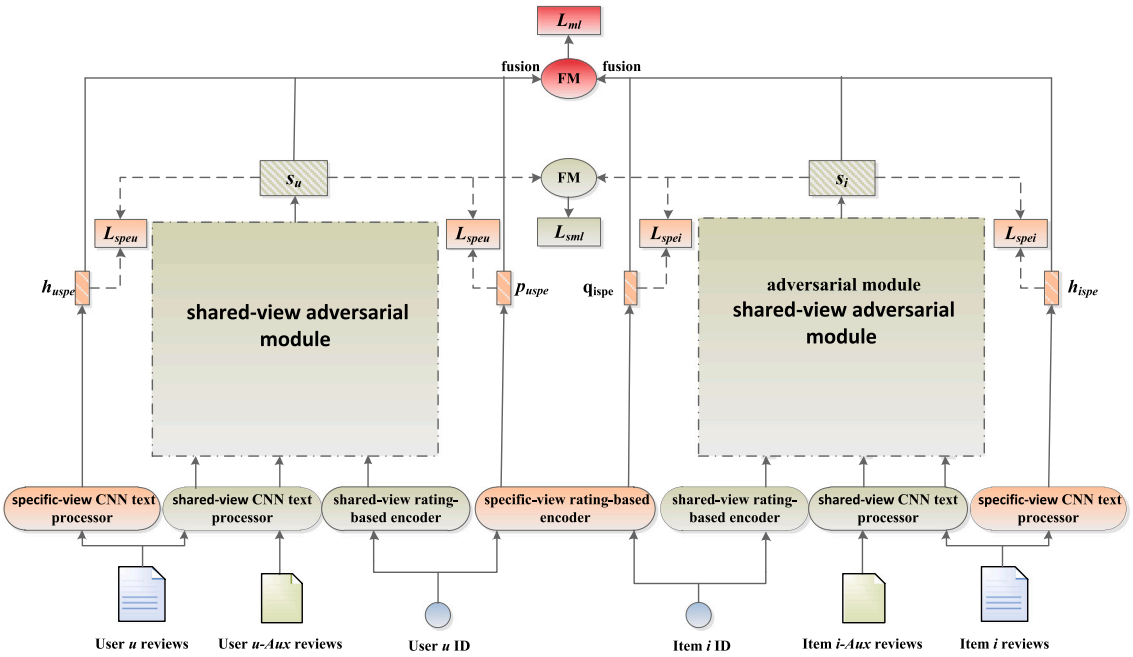


Fig. 3. The overall framework of SSIR. The part marked with orange is the specific-view information extraction module, the part marked with green is the shared-view information extraction module, and the part marked with red is the rating prediction module.

The above steps have illustrated the process of extracting a feature vector from a kernel. The convolutional neural network model has T convolution kernels to extract different feature embeddings, and the outputs of the maximum pooling operation are concatenated as Eq. (5).

$$h_{CNN} = \{o_1, o_2, \ldots, o_T\} \tag{5}$$

Fully Connected Layer: Finally, a fully connected layer is used to process the pooled embeddings, and the latent representation of user u is obtained.

$$h_u = \sigma\left(W_u h_{CNN} + b_u\right) \tag{6}$$

where W_u is the weight matrix and b_u is the bias. We can use a similar method to get the review-based latent representation h_i for an item i.

4. The proposed model

Fig. 3 illustrates the overall framework of SSIR, which consists of three components: (1) shared-view information extraction module, which gets the shared features from reviews and interaction data; (2) specific-view information extraction module, which captures the discriminative features from reviews and interaction data; and (3) rating prediction module, where the final comprehensive representations are used to estimate the a user's rating to an item. The framework of SSIR is symmetric. The left is the user network, which is used to get the final representations of users. The right is the item network. We now illustrate how to implement the user network in the following.

4.1. Shared-view information extraction

Inspired by the neighbor-based method, we extract useful related features of one user from the reviews of his neighbors, referred as auxiliary reviews, to alleviate the sparseness of his reviews. The neighbors of one user may be the most similar ones, and the similarity of their historical purchase records can be calculated with user-based CF. But the time and space complexity of this method is very high. We use the method put forward by Wu et al. (2018), which regards users have the same rating to an item with the target user as neighbors and randomly selects their reviews as auxiliary reviews. Then, we will get

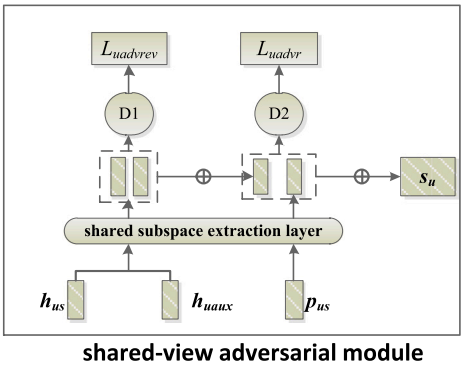


Fig. 4. The network architecture of shared-view adversarial module.

the latent vectors h_{us} and h_{usaux} from the reviews and auxiliary reviews of user u with the formulas (3)–(6) in Section 3.2. In order to map the representations of user reviews and auxiliary reviews into the same dimension, we use one shared-view CNN text processor to extract both h_{us} and h_{usaux} as illustrated in Fig. 3.

Because the same rating for only one item does not mean that the reviews of two users are totally the same. For example, one user gives a highest rating 5 because of the melody of the music A. While the other user also gives a highest rating 5 because of the singer of the music A. In this case, their reviews are similar with each other on the coarse-grained level, because they both like the music A. But their reviews may be discriminative on a fine-grained level, because they are interested in different aspects of this music. Meanwhile, reviews and ratings are two different channels of information to represent the preferences of users to items. In order to integrate the multiple types of heterogeneities in an efficient and general way, we use an adversarial module to extract shared-view information in model SSIR. Fig. 4 illustrates the detailed network architecture of this adversarial module. In particularly, the adversarial module has two parts. The first part aggregates all the embeddings from user reviews and auxiliary reviews. Let c_{revu} and c_{uaux} be the embeddings of user u obtained from user reviews and

auxiliary reviews, respectively. We get c_{revu} and c_{uaux} with the following formulas:

$$\begin{aligned} c_{revu} &= F\left(h_{us}\right) \\ c_{uaux} &= F\left(h_{uaux}\right) \\ F(x) &= \sigma(Wx+b) \end{aligned} \tag{7}$$

where $F(\cdot)$ is the shared subspace extraction layer and it is implemented with a fully connected network, W is the weight matrix and b is the bias. For the sake of brevity, we denote the embedding of user u obtained from his reviews with c_u^1, that is, $c_u^1 = c_{revu}$, and denote the latent representation of user u obtained from his auxiliary reviews with c_u^2, that is, $c_u^2 = c_{uaux}$. Then we label c_u^1 with $y_u^1 = 0$, and c_u^2 with $y_u^2 = 1$. In this way, we can construct a training set $D_{uadvrev} = \left\{\left(c_u^v, y_u^v\right) | 1 \le v \le 2, 1 \le u \le N\right\}$ for N users. Let D_1 be the discriminator. SSIR wants to confuse the discriminator that it cannot distinguish which kind of reviews the coming embeddings come from. That is, SSIR extracts the shared review-based embedding by minimizing the adversarial loss $L_{uadvrev}$. $L_{uadvrev}$ can be formed as:

$$L_{uadvrev} = F_{adv}\left(-\sum_{u=1}^{N}\sum_{v=1}^{2}\left(y_u^v log\hat{y}_u^v + \left(1-y_u^v\right)\left(1-\hat{y}_u^v\right)\right)\right) \tag{8}$$

where $F_{adv} = e^{-x}$ is a monotonically decreasing function, $\hat{y}_u^v$ is the prediction output of the discriminator D_1, i.e. $\hat{y}_u^v = D_1\left(c_u^v\right)$. In this way, there is no specific-view information between the embeddings of user u obtained from user reviews and auxiliary reviews. That is to say, the merged representation c_{urev}, i.e. $c_{urev} = c_{revu} + c_{uaux}$, only contains the shared representation of reviews.

The second part merges all the embeddings from user reviews and ratings in a similar way. Let p_{us} be the rating-based representation of user u and it is obtained by the shared-view rating-based encoder as illustrated in Fig. 3. We pass p_{us} into the shared subspace extraction layer in Fig. 4 and get the latent representation c_{usr}, i.e. $c_{usr} = F\left(p_{us}\right)$. Then we denote c_{urev} and c_{usr} with c_{ur}^1 and c_{ur}^2 respectively, that is $c_{ur}^1 = c_{urev}$ and $c_{ur}^2 = c_{usr}$. We also label them with $y_{ur}^1 = 0$ and $y_{ur}^2 = 1$. After constructing a training set $D_{uadur} = \left\{\left(c_{ur}^v, y_{ur}^v\right) | 1 \le v \le 2, 1 \le u \le N\right\}$ for the discriminator D_2, we can learn the shared-view representation of the user by minimizing the confusion adversarial loss L_{uadvr} between c_{urev} and c_{usr}. L_{uadvr} can be formed as:

$$L_{uadvr} = F_{advr}\left(-\sum_{ur=1}^{N}\sum_{v=1}^{2}\left(y_{ur}^v log\hat{y}_{ur}^v + \left(1-y_{ur}^v\right)\left(1-\hat{y}_{ur}^v\right)\right)\right) \tag{9}$$

where $F_{advr} = e^{-x}$ is a monotonically decreasing function, y_{ur}^v is the prediction output of the discriminator D_2, i.e. $\hat{y}_{ur}^v = D_2\left(c_{ur}^v\right)$. Then the shared representation of the user s_u, i.e. $s_u = c_{urev} + c_{usr}$ only contains the shared-view information of reviews and ratings. Therefore, the confusion adversarial loss of users can be formed as:

$$L_{advu} = L_{uadvrev} + L_{uadvr} \tag{10}$$

Because the item network has a similar network architecture, it is easy to capture the shared representation of item i, i.e. s_i by minimizing the confusion adversarial loss of items L_{advi}.

However, if only L_{advu} and L_{advi} are used, it may be invalid, because noise can easily confuse the discriminator. We further use the rating prediction loss to ensure that s_u and s_i are effective. In particular, s_u and s_i can be input into a FM to get a preliminary prediction rating $\hat{r}_{u,i}^s$ as follows:

$$\begin{aligned} z_{u,i}^s &= \left[s_u, s_i\right] \\ \hat{r}_{u,i}^s &= m_{s_0} + \mathbf{m_s^T} z_{u,i}^s + \frac{1}{2} z_{u,i}^s M_s z_{u,i}^s \\ M_s &= v_{sj}^T v_{sk} \end{aligned} \tag{11}$$

where $[\cdot]$ is the concatenating operation, m_{s_0} is the global bias, $\mathbf{m}_s$ is the coefficient vector for $z_{u,i}^s$, M_s is the weight matrix of the second order interaction, and the diagonal elements of M_s are 0. v_{sj} and v_{sk} are the latent vectors, which are v-dimensional and related to the jth and kth dimension of $z_{u,i}^s$ respectively. Therefore, the preliminary rating prediction loss can be formed as:

$$L_{sml} = \sum_{u=1}^{N}\sum_{i=1}^{M}\left(r_{u,i} - \hat{r}_{u,i}^s\right)^2 \tag{12}$$

4.2. Specific-view information extraction

On one hand, we should take advantage of the shared-view information between reviews and ratings. On the other hand, the discriminative information between reviews and the ratings cannot be ignored. In SSIR, we introduce a specific-view module for further rating prediction. However, what is the useful specific-view information is hard to define. We can extract it by eliminating the shared-view information from original information, and this can be achieved by applying orthogonal constraints on shared-view module and specific-view module. For the user u, s_u is the shared representation, h_{uspe} is the specific-view representation of his reviews and p_{uspe} is the specific-view representation of his ratings. h_{uspe} is obtained with a specific-view CNN text processor, which has the same architecture with the shared-view CNN text processor but different parameters. p_{uspe} is got with a specific-view rating-based encoder, which has different parameters with the shared-view rating-based encoder used to obtain p_{us}. The loss L_{speu} forces the orthogonality between s_u and h_{uspe}, p_{uspe}:

$$L_{speu} = \sum_{u=1}^{N}\left(\left\|s_u^T h_{uspe}\right\|_2^2 + \left\|s_u^T p_{uspe}\right\|_2^2\right) \tag{13}$$

where $\|\cdot\|_2^2$ is the squared L2-norm. L_{speu} encourages the shared-view representation be different from the specific-view representation. We can use a similar method to get s_i and h_{ispe}, q_{ispe} for item i.

4.3. Rating prediction

After receiving the shared-view and specific-view representations for users and items, we process all of them with dropout technology to prevent our model from over fitting. For modeling the high-order feature interaction between user representations and item representations, we cascade all the vectors into $z_{u,i}$, and then send them into FM to predict the final rating $\hat{r}_{u,i}$:

$$\begin{aligned} z_{u,i} &= \left[s_u, h_{uspe}, p_{uspe}, s_i, h_{ispe}, q_{ispe}\right] \\ \hat{r}_{u,i} &= m_0 + \mathbf{m}^{\mathbf{T}} z_{u,i} + \frac{1}{2} z_{u,i} M z_{u,i} \\ M &= v_j^T v_k \end{aligned} \tag{14}$$

where m_0 is the global bias, $\mathbf{m}$ is the coefficient vector for $z_{u,i}$, M is the weight matrix of second order interaction, the diagonal elements of M are 0. v_j and v_k are the latent vectors, which are v-dimensional and related to the jth and kth dimension of $z_{u,i}$ respectively. For the convenience of calculation, we limit the sizes of $s_u, h_{uspe}, p_{uspe}, s_i, h_{ispe}, q_{ispe}$ to be the same. Therefore, the final rating prediction loss can be formed as:

$$L_{ml} = \sum_{u=1}^{N}\sum_{i=1}^{M}\left(r_{u,i} - \hat{r}_{u,i}\right)^2 \tag{15}$$

The total loss function is as follows:

$$LOSS = L_{ml} + L_{sml} + L_{advu} + L_{advi} + L_{speu} + L_{spei} \tag{16}$$

We optimize parameters over mini-batches by utilizing RMSprop optimizer for training.

5. Experiments

In this section, we use eight real-world datasets from two different websites to conduct extensive experiments for performance evaluation. Then, we analyze the influence of different parameter settings of SSIR on the experimental results and the contribution of different components to the performance of SSIR.

Table 2
Statistics of the datasets.

Datasets	DG	MI	OF	VD	TO	AUTO	KD	YELP
Users	5541	1429	4905	5130	16638	2928	63769	444
Items	3568	900	2420	1685	10217	1835	71031	1366
Ratings	64705	10254	53237	37126	134413	20467	1003060	3055
Words per-user	1917	528	1290	545	721	485	954	808
Words per-item	2977	838	2615	1658	1175	774	1063	262
Density	0.327%	0.797%	0.448%	0.430%	0.079%	0.381%	0. 221%	0.504%

Table 3
When the training set is 80%, the overall performance comparison on eight datasets. The best and the second-best results are highlighted in boldface and underlined respectively. * and ** denote the statistical significance for $p < 0.05$ and $p < 0.01$, respectively, between the best and the second-best results.

Method (MSE)	DG	MI	OF	VD	TO	AUTO	KD	YELP
User based-CF	1.619	1.111	1.112	1.682	1.432	1.123	-	1.229
Item based-CF	1.380	1.045	1.133	1.689	1.389	1.161	-	1.533
SVD++	1.456	1.082	1.107	1.569	1.355	1.101	1.193	1.205
DeepCoNN	1.013	0.835	0.805	1.087	1.036	0.873	0.814	0.997
PARL	0.849	0.817	0.749	0.958	0.997	0.823	0.652	0.965
CARL	0.860	0.816	0.773	0.967	1.013	0.807	0.830	0.978
SSIR	**0.842***	**0.778****	**0.733****	**0.925****	**0.979****	**0.772****	**0.645***	**0.916****
Method (MAE)	DG	MI	OF	VD	TO	AUTO	KD	YELP
User based-CF	0.933	0.736	0.780	0.952	0.856	0.736	-	0.896
Item based-CF	0.877	0.733	0.799	0.978	0.862	0.759	-	0.946
SVD++	0.902	0.756	0.767	0.954	0.859	0.759	0.701	0.881
DeepCoNN	0.741	0.624	0.638	0.752	0.709	0.717	0.690	0.789
PARL	**0.681****	0.632	0.635	0.708	0.714	0.635	0.550	0.810
CARL	0.785	0.613	0.668	0.726	0.767	0.663	0.735	0.809
SSIR	0.688	**0.603****	**0.629***	**0.704***	**0.710***	**0.605****	**0.545***	**0.770****

5.1. Datasets

Seven of our datasets are from Amazon Public 5-core datasets[1]: Digital Music (DG), Musical Instruments(MI), Automotive (AUTO), Instant Video (VD), Tools Improvement (TO), Office Products (OF), Kindle Store and Automotive(KD) The other dataset is collected from the Yelp challenge website.[2] Note that for yelp, we extract 5-core from it, and only the records in January 2017 are selected as the final dataset, which is represented as YELP. Each user and item selected by the 5-core operation has at least 5 reviews. We preprocess all the datasets as follows: (1) delete stop words and remove the words with document frequency greater than 0.5; (2) calculate the TF-IDF score of all the words and only select the top 20000 different words to make a vocabulary; (3) remove the words of the original document which are not in the vocabulary; (4) enforce the length of all review documents to be 300 words.

Table 2 shows the statistics of all datasets. As we can see, these eight datasets have different sparsity degrees in the user–item interaction data and user–item reviews. To be specific, the size of MI is the smallest in seven Amazon datasets, but it has the highest density among all eight datasets. On the contrary, the average number of words in each review in YELP and AUTO datasets is the least.

We randomly split the dataset into three parts: 80% of the dataset is the training set, 10% of it is the validation set, and the other is the test set.

In our experiments, mean square error (MSE) and mean absolute error (MAE), which are the most common evaluation metrics in related works, are used to evaluate the performance of all the comparing methods:

$$MSE = \frac{1}{|\mho_t|}\sum_{u,i\in\mho_t}\left(y_{u,i}-\hat{y}_{u,i}\right)^2$$
$$MAE = \frac{1}{|\mho_t|}\sum_{u,i\in\mho_t}\left|y_{u,i}-\hat{y}_{u,i}\right| \quad (17)$$

where $\mho_t$ is the test set of the user–item pairs.

5.2. Baselines

We compare our model SSIR with the following methods:

User based-CF (Konstan, Miller, Maltz, Herlocker, & Riedl, 1997): This algorithm adopts the nearest neighbor method to find the neighbor

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] https://www.yelp.com/dataset/challenge

Table 4
When the training set is 60%, the overall performance comparison on eight datasets. The best and the second-best results are highlighted in boldface and underlined respectively. * and ** denote the statistical significance for $p < 0.05$ and $p < 0.01$, respectively, between the best and the second-best results.

Method (MSE)	DG	MI	OF	VD	TO	AUTO	KD	YELP
User based-CF	1.661	1.063	1.184	1.761	1.469	1.250	-	1.401
Item based-CF	1.531	1.053	1.221	1.607	1.481	1.286	-	1.792
SVD++	1.459	1.082	1.146	1.569	1.377	1.191	1.206	1.352
DeepCoNN	1.212	0.819	0.890	1.243	1.083	0.886	0.854	1.092
PARL	0.908	0.779	0.780	1.148	1.014	0.862	0.667	1.337
CARL	0.993	0.798	0.770	1.001	1.004	0.842	0.832	1.325
SSIR	**0.895****	**0.770****	**0.762****	**0.965****	**0.998****	**0.830***	**0.656***	**1.040****
Method (MAE)	DG	MI	OF	VD	TO	AUTO	KD	YELP
User based-CF	0.941	0.721	0.801	0.964	0.862	0.761	-	1.401
Item based-CF	0.912	0.724	0.814	0.958	0.879	0.790	-	1.792
SVD++	0.906	0.756	0.796	0.954	0.863	0.777	0.709	1.352
DeepCoNN	0.812	0.615	0.761	0.863	0.757	0.638	0.662	1.092
PARL	**0.701****	0.612	0.648	0.805	0.743	0.653	0.561	1.337
CARL	0.751	0.656	0.670	0.759	0.764	0.678	0.736	1.325
SSIR	0.706	**0.604****	**0.642****	**0.726****	**0.720****	**0.646****	**0.558***	**1.040****

set of one user. Then the algorithm predicts users' preferences based on the preferences of their neighbors.

Item based-CF (Sarwar, Karypis, Konstan, & Riedl, 2001): This algorithm calculates the similarity between items according to all users' historical interaction data, and then recommends items which are similar with the items preferred by users.

SVD++ (Koren, 2008): This algorithm not only considers the interaction between users and items, but also takes the user's preference of other items into account, when predicting the user's rating of a certain item.

DeepCoNN (Zheng et al., 2017): This algorithm transforms all user and item reviews into two long documents respectively, and uses CNN to model preferences of users and features of items. Then, it concatenates the latent vectors of users and items, and inputs them into FM to predict ratings.

PARL (Wu et al., 2018): It exploits auxiliary reviews written by users' like-minded neighbors, and then extracts useful features from auxiliary reviews to alleviate the sparsity problem of reviews and make recommendation.

CARL (Wu et al., 2019): This algorithm learns each user–item pair's context-aware representation based on the individual latent features and their interaction data to make rating prediction.

5.3. Experimental settings

In this paper, we optimize the Hyper-parameters of all methods according to grid search. We run all comparing approaches more than five times on the test set and report the mean metrics results. We optimize the number of convolution filters from $\{25, 50, 75, 100, 125\}$. The latent dimension of Amazon's seven datasets is optimized from $\{15, 30, 45, 50, 75\}$, and the dimension of YELP is optimized from $\{2, 4, 8, 16, 32\}$. The word embeddings are initialized randomly and then processed by fine-tuning technique. Let the dimension of word embeddings be 300. The mini-batch of all datasets is 200.

For SSIR, the latent dimension of Amazon's seven datasets is 30, and YELP's latent dimension is 4. The number of convolution filters is 50 and the window size is 3. v is set to 100 for FM to predict ratings. The learning rate is set to 0.002. And the retention probability of dropout is set to 0.6.

5.4. Performance evaluation

The performance of SSIR and baselines are reported in Table 3. Here, we make the following observations:

First, we can see that the oldest methods (User based-CF and Item based-CF), perform worst on all eight datasets. It is more obvious on sparse datasets, such as TO and DG. Moreover, because of the high space complexity of these two methods, the memory required on the largest dataset KD exceeds the memory of our computer. SVD++ integrates MF technology and neighbor-based model. It performs better than the oldest methods. But the performance of SVD++ is not the best, and all review-based methods achieve far better performance than SVD++, especially for those datasets with sparser interaction data, such as TO and DG. This verifies that we can extract more semantic information from reviews than only from rating matrix when predicting user rating behavior.

Second, among the review-based methods, DeepCoNN gets the worst overall performance. Compared with DeepCoNN, PARL selects auxiliary reviews from like-minded neighbors, and CARL exploits both rating interaction information and reviews. They both get better performance than DeepCoNN. This reveals that the performance of the review-based methods can be improved by exploiting auxiliary reviews and rating interaction information.

Third, the overall performance of SSIR is the best on eight datasets. Compared with PARL, the performance of SSIR achieves the maximum improvement on MI dataset. This is reasonable because MI provides the most abundant interaction information, which is not taken into consideration in PARL. Compared with CARL, the performance of SSIR gains the maximum improvement on YELP dataset. Since YELP has the least number of words, auxiliary reviews play a greater role in solving the sparsity problem of reviews.

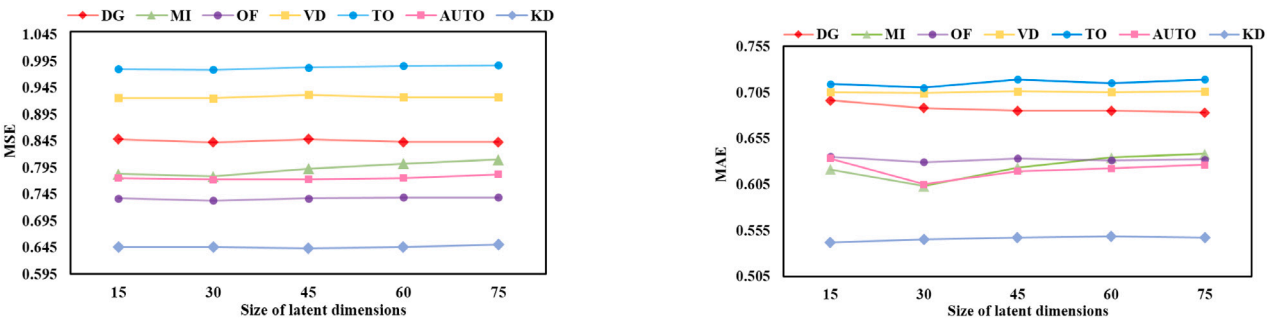


Fig. 5. Impact of size of latent dimensions on Amazon's seven datasets.

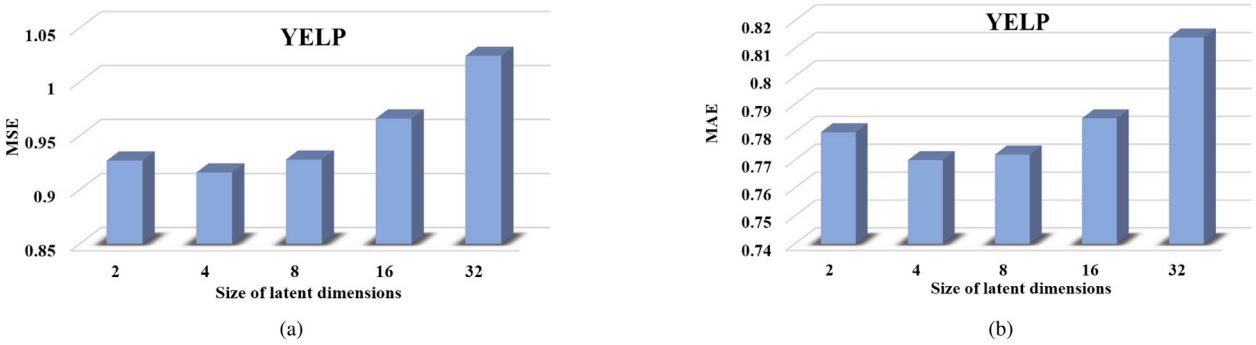


Fig. 6. Impact of size of latent dimensions on YELP.

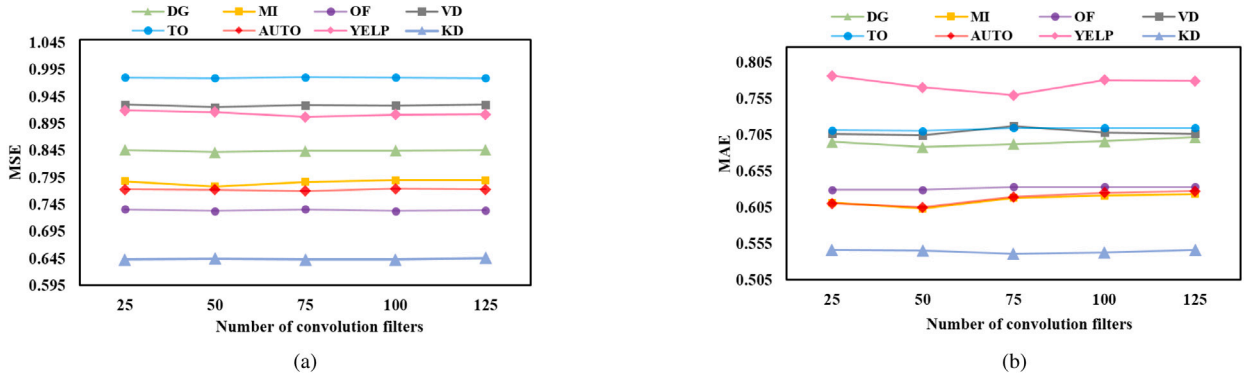


Fig. 7. Impact of number of convolution filters.

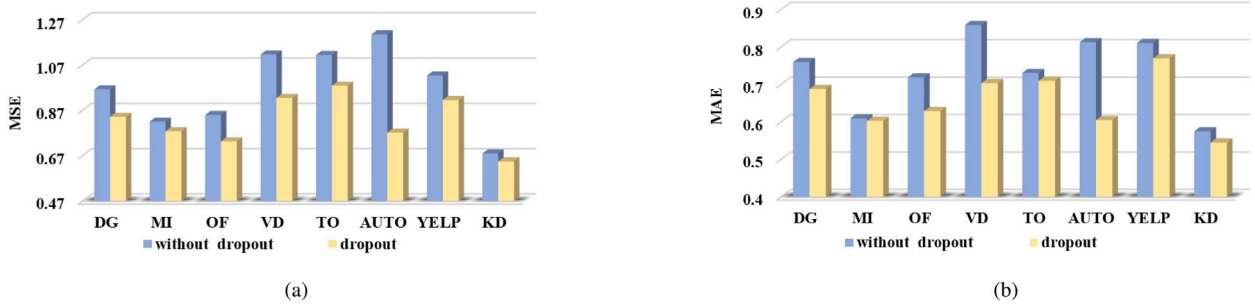
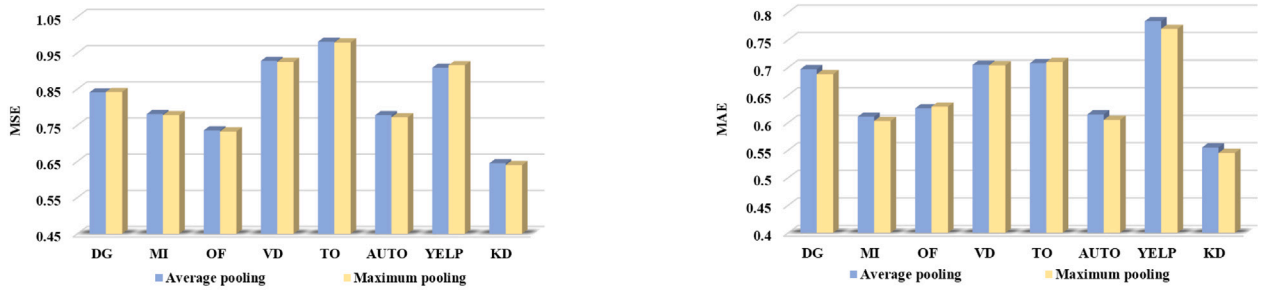


Fig. 8. Impact of dropout.

In order to demonstrate the superiority of SSIR, we reduce the proportion of the training set to 60% for a group of comparative experiments. From Table 4, we can see that the overall performance of all models will deteriorate when the training sets of the datasets become smaller. But the performance of our model is still better than the baselines. This shows that SSIR can adapt to the recommendation scenarios with fewer reviews and ratings.

Table 5
Average time cost (in seconds) of training per epoch.

Method	DG	MI	OF	VD	TO	AUTO	KD	YELP
SSIR	14.06	2.29	11.68	8.38	29.31	4.46	82.45	0.68
PARL	30.05	4.61	23.39	15.87	59.60	9.14	269.04	1.47
CARL	7.66	1.05	6.14	4.29	18.40	2.15	165.63	0.34



(a) (b)

Fig. 9. Impact of different pooling operations.

5.5. Parameter analysis of SSIR

We now carry out extensive research into the influence of different parameter settings on SSIR.

Size of latent dimension. Figs. 5 and 6 plot the performance of SSIR by changing the size of latent dimensions. As we can see from these two figures, with the increasing of the latent feature dimension size, both MSE and MAE of SSIR first decrease and then increase. When the size of latent dimensions is set to 30, the performance of SSIR reaches the best on Amazon's six datasets. And the performance of SSIR reaches the best when the size of latent dimensions is specified as 4 on YELP dataset. We think that SSIR cannot capture enough latent information about users and items when the dimension of latent features is too small. And the model may have the problem of overfitting, when the size of latent dimensions is too large. Therefore, in our experiment, the size of latent dimensions is set to 30 on Amazon's six datasets and 4 on YELP dataset.

Number of convolution filters. Fig. 7 plots the performance of SSIR varying with number of convolution filters. As can be seen from Fig. 7, with the increasing of the number of convolution filters, both MSE and MAE of SSIR first decrease and then increase in our experiments. If the number of convolution filters is too small, fitting ability of SSIR is not enough, and the model will be underfitting. On the contrary, if the number of convolution filters is too large, the model is easy to be overfitting. Based on the above observation, we set the number of convolution filters as 50.

Dropout. Fig. 8 shows the effect of dropout on SSIR. When dropout is executed, we set the keep probability of each neuron to 60%. From Fig. 8 we can see that the performance of SSIR will be improved, when dropout strategy is used, especially on AUTO dataset. This is because dropout strategy can prevent SSIR from overfitting, and make the model have a better generalization ability. Among all the datasets, AUTO has both sparser interaction data and sparser reviews. Therefore, the performance of SSIR is improved significantly on AUTO.

Maximum pooling and Average pooling. Fig. 9 shows the effect of two pooling operations: maximum pooling and average pooling. As can be seen from Fig. 9, the selection of different pooling operation has little effect on the performance of SSIR. Both these two pooling operations can extract the semantic information from reviews very well. In this paper, we use maximum pooling as an example.

Efficiency Comparison. We conduct the experiments of all the datasets except KD on the machine with GPU Tesla V100 16 GB. For KD dataset, we conduct the experiments on the machine with GPU NVIDIA RTX 3090 24 GB. The mini-batch of all datasets is 200. We count the average time cost for training one epoch of each model. Since the traditional CF-based methods run much faster than the reviews-based methods, we do not compare them with our model. SSIR, PARL, and CARL are all extended on the basis of DeepCoNN, so DeepCoNN is faster than these three models. Therefore, we only show the efficiency comparison results of these three models in Table 5. CARL is faster than SSIR, because it does not use auxiliary reviews, and it does not extract information from ratings and reviews from dual-views. However, SSIR is faster than PARL, because our model does not perform additional feature extraction and gating operations for auxiliary reviews, which may reduce the time complexity of the model.

5.6. Ablation study

In order to validate the effectiveness of each component in SSIR, we conduct ablation experiments. By removing different components, we design four variants of our model as follows:

- **SSIR-SHA**: We remove the shared-view module, and only use the information of the specific-view to predict the ratings.
- **SSIR-SPE**: We delete the specific-view module, and only use the information of the shared-view to predict the ratings.
- **SSIR-AUX**: We abandon the neighbor-based component, and do not use auxiliary reviews for rating prediction.
- **SSIR-ADV**: In the shared-view module, we do not apply the adversarial loss L_{advu} or L_{advi}, but directly fuse the latent features of ratings and reviews.

The performance of SSIR and its variants can be seen in Fig. 10. Whether the shared-view module or the specific-view module is removed, the performance of the variant models, i.e. SSIR-SHA and SSIR-SPE, will be extremely poor. This validates that in our model, shared-view and specific-view modules are important and can complement each other. The performance of SSIR-AUX is much lower than SSIR. This reveals that auxiliary reviews can alleviate the sparsity of reviews. Compared with SSIR-AUX, the performance of SSIR gains the maximum improvement on YELP dataset. This is reasonable, because YELP has the least average number of words. Both MSE and MAE of SSIR-ADV are higher than that of SSIR. This validates that the idea of confusion adversarial loss can help to extract shared-view information and improve the performance of recommendation models.

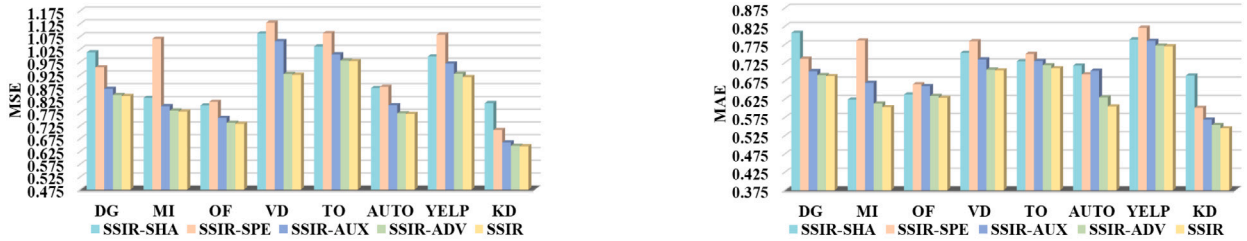
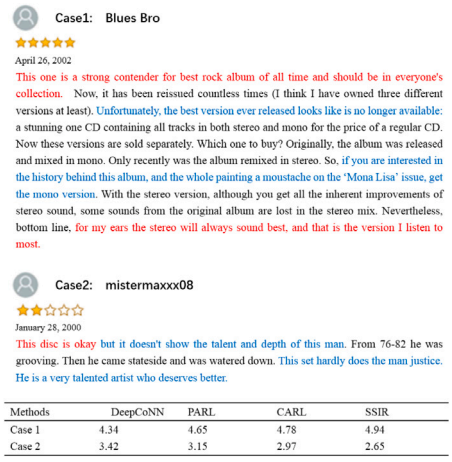


Fig. 10. Comparison of SSIR and its variants.



Methods	DeepCoNN	PARL	CARL	SSIR
Case 1	4.34	4.65	4.78	4.94
Case 2	3.42	3.15	2.97	2.65

Fig. 11. Case study of recommendation for two users of Digital Music dataset.

5.7. Case study

One contribution of this paper is to learn more accurate representations of users and items by extracting both shared-view and specific-view information from numerical ratings and text reviews while recommending preferable items. To demonstrate this, we show the two examples drawn from Digital Music(DG) dataset.

We randomly select two users, whose IDs are u960 and u4500, and track their reviews on items i3 and i18, respectively. Fig. 11 demonstrates the effectiveness of our method.

- From Fig. 11 we can see, u960 gave the highest score to the CD i3. He also expressed his interest in his reviews with the sentences marked with red. Even though he pointed out the shortcomings of the CD with the words marked with blue, we can estimate he was indeed satisfied with the CD. The predicted rating of our method SSIR is 4.94, and it is closer to the ground truth than DeepCoNN, PARL and CARL.
- From the sentences marked with blue in the second reviews, we can infer u4500 did not like the disc i18 because he thought the disc did not show the talent and depth of the artist, and he gave a very low score. Combined with the user's rating and review, we estimate that the more reasonable rating u4500 may give should vary from 2.4 to 2.6. The predicted rating of our method SSIR is 2.65, and it is also closer to the ground truth than the baselines.
- Our method minimizes the confusion adversarial loss to make the distributions of features extracted from ratings and reviews become as similar as possible, and it applies the orthogonal constraints on the shared-view and specific-view modules to guarantee there is no correlation between these two modules, thus the specific information is obtained by eliminating the shared information from the original information. Our method may integrate the two types of heterogeneities, that is ratings and reviews, in an efficient and general way, and performs better.

6. Conclusion and future work

In this paper, we propose a Dual-view recommendation model SSIR, which can extract effective representations from ratings and reviews. The key of our model is to extract shared view and specific-view information from ratings and text reviews. Specifically, we jointly minimize the loss of confusion adversarial and rating prediction loss to extract the shared information in the shared-view module. Then, we apply orthogonal constraints on the specific-view module and shared-view module to extract the discriminative features from reviews and interaction data. In addition, we use neighbor-based approach to complement reviews with auxiliary reviews for better performance. Experiments show that SSIR outperforms the comparing methods on eight real-world datasets. However, the neighbor sampling method used in this paper needs to be improved, and there are other useful information in the recommendation system, such as the age of users and the genre of items. In the future, we will study how to select neighbors and how to extract the similar information from the reviews of neighbors of the target user, and plan to incorporate other useful information into our model for more efficient recommendation.

CRediT authorship contribution statement

Huiting Liu: Conceptualization, Methodology, Writing – review & editing. **Jindou Zhao:** Software, Validation, Formal analysis, Data curation, Writing – original draft, Visualization. **Peipei Li:** Investigation. **Peng Zhao:** Resources. **Xindong Wu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been supported by the National Key Research and Development Program of China No. 2016YFB1000901, the National Natural Science Foundation of China Nos. 61202227 and 61602004, Natural Science Foundation of Anhui Province, No. 2008085MF219 and Provincial Natural Science Foundation of Anhui Higher Education Institution of China, No. KJ2018A0013.

References

Berg, R., Kipf, T., & Welling, M. (2017). Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263.

Catherine, R., & Cohen, W. (2017). TransNets: Learning to transform for recommendation. In *Proceedings of the eleventh ACM conference on recommender systems* (pp. 288–296). ACM.

Chen, X., Xu, H., Zhang, Y., Tang, J., & Zha, H. (2018). Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 108–116). ACM.

Chen, T., Yin, H., Ye, G., Huang, Z., Wang, Y., & Wang, M. (2020). Try this instead: Personalized and interpretable substitute recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 891–900). ACM.

Chen, J., Zhang, H., He, X., Nie, L., Liu, W., & Chua, T. (2017). Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 335–344). ACM.

Chen, C., Zhang, M., Liu, Y., & Ma, S. (2018). Neural attentional rating regression with review-level explanations. In *Proceedings of the international conference on world wide web* (pp. 1583–1592). ACM.

Diao, Q., Qiu, M., Wu, C., Smola, A., Jiang, J., & Wang, C. (2014). Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 193–202). ACM.

Dong, X., Ni, J., Cheng, W., Chen, Z., & Melo, G. (2020). Asymmetrical hierarchical networks with attentive interactions for interpretable review-based recommendation. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7667–7664).

Guan, X., Cheng, Z., He, X., Zhang, Y., Zhu, Z., Peng, Q., et al. (2019). Attentive aspect modeling for review-aware recommendation. *ACM Transactions on Information Systems (TOIS)*, *37*(3), 28:1–28:27.

Hammou, B., Lahcen, A., & Mouline, S. (2019). An effective distributed predictive model with matrix factorization and random forest for big data recommendation systems. *Expert Systems with Application*, *137*(DEC.), 253–265.

He, X., He, Z., Du, X., & Chua, T. (2018). Adversarial personalized ranking for recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 355–364). ACM.

He, X., He, Z., Song, J., Liu, Z., Jiang, Y., & Chua, T. (2018). NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *30*(12), 2354–2366.

He, X., Liao, L., Zhang, H., Nie, L., & Chua, T. (2017). Neural collaborative filtering. In *proceedings of the international conference on world wide web* (pp. 173–182). ACM.

Jakob, N., Muller, M., Weber, H., & Gurevych, I. (2009). Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on topic-sentiment analysis for mass opinion* (pp. 57–64). ACM.

Kim, D., Park, C., Oh, J., Lee, S., & Yu, H. (2016). Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 233–240). ACM.

Konstan, J., Miller, B., Maltz, D., Herlocker, J., & Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, *40*(3), 77–87.

Koren, Y. (2008). 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. 426–433). ACM.

Li, S., Zhou, J., Xu, T., Liu, H., & Xiong, H. (2020). Competitive analysis for points of interest. In *Proceedings of the 26th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 1265–1274). ACM.

Liang, D., Krishnan, R., Hoffman, M., & Jebara, T. (2018). Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference on world wide web* (pp. 689–698). ACM.

Liu, H., Wang, Y., Peng, Q., Wu, F., Gan, L., Pan, L., et al. (2020). Hybrid neural recommendation with joint deep representation learning of ratings and reviews. *Neurocomputing*, 77–85.

Liu, H., Wang, W., Xu, H., Peng, Q., & Jiao, P. (2020). Neural unified review recommendation with cross attention. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1789–1792). ACM.

McAuley, J., & Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the seventh ACM conference on recommender systems* (pp. 165–172). ACM.

Panagiotakis, C., Papadakis, H., & Fragopoulou, P. (2021). A dual hybrid recommender system based on SCoR and the random forest. *Computer Science and Information Systems*, *18*(1), 115–128.

Panagiotakis, C., Papadakis, H., Papagrigoriou, A., & Fragopoulou, P. (2021). Improving recommender systems via a Dual Training Error based Correction approach. *Expert Systems with Applications*, *183*(5), Article 115386.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the international conference on world wide web* (pp. 285–295). ACM.

Seo, S., Huang, J., Yang, H., & Liu, Y. (2017). Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the eleventh ACM international conference* (pp. 297–305). ACM.

Sun, P., Wu, L., Zhang, K., Fu, Y., & Wang, M. (2020). Dual learning for explainable recommendation: Towards unifying user preference prediction and review generation. In *Proceedings of the international conference on world wide web* (pp. 837–847). ACM.

Tang, J., & Wang, K. (2018). Personalized top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 565–573). ACM.

Tay, Y., Tuan, L., & Hui, S. (2018). Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 world wide web conference on world wide web* (pp. 729–739). ACM.

Wang, C., & Blei, D. (2011). 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.

Wang, Z., Xia, H., Du, B., Chen, S., & Gang, C. (2021). Joint representation learning with ratings and reviews for recommendation. *Neurocomputing*, *425*, 181–190.

Wu, Y., Dubois, C., Zheng, A., & Ester, M. (2016). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining* (pp. 153–162). ACM.

Wu, L., Quan, C., Li, C., & Ji, D. (2018). PARL: Let strangers speak out what you like. In *Proceedings of the 27th ACM international on conference on information and knowledge management* (pp. 677–686). ACM.

Wu, L., Quan, C., Li, C., Wang, Q., & Zheng, B. (2019). A context-aware user-item representation learning for item recommendation. *ACM Transactions on Information Systems*, *37*(2), 22:1–22:29.

Xia, H., Wang, Z., Du, B., Zhang, L., Chen, S., & Chun, G. (2019). Leveraging ratings and reviews with gating mechanism for recommendation. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 1573–1582). ACM.

Xue, H., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the twenty-sixth international joint conference on artificial intelligence* (pp. 3203–3209).

Zhang, K., Liu, X., Wang, W., & Li, J. (2021). Multi-criteria recommender system based on social relationships and criteria preferences. *Expert Systems with Applications*, *176*, Article 114868.

Zhang, S., Yin, H., Chen, T., Nguyen, Q., Huang, Z., & Cui, L. (2020). GCN-based user representation learning for unifying robust recommendation and fraudster detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 689–698). ACM.

Zheng, L., Noroozi, V., & Yu, S. (2017). Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 425–434). ACM.

Zou, L., Xia, L., Gu, Y., Zhao, X., & Yin, D. (2020). Neural interactive collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 749–758). ACM.