Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6120068, 12 pages
https://doi.org/10.1155/2021/6120068

Research Article

Hybrid Recommendation Scheme Based on Deep Learning

Fangpeng Ming,[1] **Liang Tan**,[1,2] **and Xiaofan Cheng**[1]

[1]*School of Computer Science, Sichuan Normal University, Chengdu 610000, SiChuan, China*
[2]*Institute of Computer Science, Chinese Academy of Sciences, Beijing 100000, China*

Correspondence should be addressed to Fangpeng Ming; frank.ming@foxmail.com

Received 2 July 2021; Revised 2 November 2021; Accepted 16 November 2021; Published 22 December 2021

Academic Editor: Xiao-dong Feng

Big data has been developed for nearly a decade, and the information data on the network is exploding. Facing the complex and massive data, it is difficult for people to get the demanded information quickly, and the recommendation algorithm with its characteristics becomes one of the important methods to solve the massive data overload problem at this stage. In particular, the rise of the e-commerce industry has promoted the development of recommendation algorithms. Traditional, single recommendation algorithms often have problems such as cold start, data sparsity, and long-tail items. The hybrid recommendation algorithms at this stage can effectively avoid some of the drawbacks caused by a single algorithm. To address the current problems, this paper makes up for the shortcomings of a single collaborative model by proposing a hybrid recommendation algorithm based on deep learning IA-CN. The algorithm first uses an integrated strategy to fuse user-based and item-based collaborative filtering algorithms to generalize and classify the output results. Then deeper and more abstract nonlinear interactions between users and items are captured by improved deep learning techniques. Finally, we designed experiments to validate the algorithm. The experiments are compared with the benchmark algorithm on (Amazon item rating dataset), and the results show that the IA-CN algorithm proposed in this paper has better performance in rating prediction on the test dataset.

1. Introduction

With the continuous development and maturity of big data, cloud computing, and other technologies, this has greatly enriched our production and life. At the same time, the information in the network has also shown an explosive growth. The information generated by human beings in the past ten years or so is more than the total amount of information generated in the previous thousands of years [1]. Generally, when faced with massive amounts of information, it is often difficult for people to find what they want and what they are interested in. Even if they can find it, it will take a long time. How to effectively deliver useful information to users is a very important issue. In order to solve this problem, Google has introduced search engine technology. Users can enter keywords to search according to the purpose of quickly and accurately finding the content they want. However, this method is only suitable for occasions where users actively seek information and need a clear purpose. In actual daily life, many users' needs for seeking information are vague, latent, and passive. Users may not know how to describe what they are interested in and may not even actively seek out the content they are interested in. In this case, search engines cannot accurately convey effective information to users.

Recommendation algorithm is one of the effective means to solve the above problems. It can be modeled by analyzing users' browsing history, user-item rating information, and other information such as users' preferences, to discover some potential personalized needs of users. Currently, this type of recommendation algorithm is the most widely used on e-commerce websites and social networks, so recommendation algorithms have important research significance and application value.

In traditional recommendation algorithms, both content-based recommendation [2] and collaborative filtering recommendation [3] have achieved certain success in discovering the relevance of users and items. However, with the

rapid increase of network data in recent years and taking the characteristics of "big data" into consideration, these traditional recommendation algorithms are not completely suitable for the current network environment. First of all, the data is sparse. In current e-commerce websites and social networks, the user's evaluation record for a certain item is incomplete, and there are a lot of null values in the evaluation-related index of the user and the item. Secondly, for newly listed items or newly registered users in the system, the algorithm does not properly handle this type of data. Finally, the frequency of a user consuming a valuable item in the system is extremely low, but, for the investors, the profit that can be made is considerable, and the algorithm does not specifically deal with this type of problem. Therefore, the effect of traditional recommendation algorithms is limited.

To address the above issues, this paper designs a hybrid recommendation algorithm based on deep learning. The algorithm uses integrated strategies to fuse multiple recommendation algorithms and generalize and classify the output results of multiple models, thus compensating for the disadvantages carried out between different algorithms. Then, based on the fusion model, the deeper and abstract nonlinear interaction relationship between users and items is captured through deep learning technology, which can effectively solve the cold start and long tail items problems that exist in traditional algorithms. At the same time, the accuracy of the algorithm has been further improved. The main contributions of this article are as follows:

(1) An integrated strategy model of multiple recommendation algorithms is designed, and the output result of the model is preprocessed and preclassified on the dataset.

(2) Related comparative experiments are designed, an improved CNN deep learning model IA-CN is trained, and a comparative analysis and comparison of advantages and disadvantages with the benchmark model are made.

The remainder of the paper is organized as follows: the second section introduces related work, the third section introduces the design and algorithm description related to the hybrid recommendation scheme, the fourth section is devoted to the experiment, and the final section gives the summary.

2. Related Work

The recommendation algorithm originated from a branch of data mining science and became an independent research field in the 1990s. Early GroupLens was proposed based on the idea of collaborative filtering [4] to complete the recommendation task, and the research of recommendation algorithm has been in continuous development.

Collaborative filtering algorithms are starting to gain popularity. They can be broadly classified into two types: nearest-neighbor-based and model-based. The former calculates the similarity of users or items in the rating matrix to make recommendations for users. Sarwar et al. [5] proposed the item-based algorithm on this basis, and it was successfully applied in Amazon's e-commerce system. In addition to calculating the similarity, the latter also requires designing a model, but it can be easily extended to large-scale data systems to solve the recommended real-time problem. However, massive data can bring data sparsity and cold start problems. Yu [6] proposed a collaborative filtering algorithm that utilizes the potential factor space of the auxiliary domain to expand user and item features, which overcomes the shortcoming of using only a single auxiliary domain recommendation. Simon Funk proposed a collaborative filtering recommendation algorithm based on matrix decomposition [7], which gained a large improvement in recommendation accuracy. Subsequently, scholars have continuously improved it and successively proposed probabilistic matrix decomposition model (PMF) [8], SVD++ model [9], MCFSAE [10], and so forth, which have gained a certain improvement in overcoming data sparsity.

However, for the highly sparse rating data in the big data environment, the performance of model-based recommendation algorithms is relatively low, and they cannot cope with the "cold start" problem of new items or new users. With the development of the Internet and social networks, a large amount of auxiliary information can be obtained, such as social relationships and user information. The application of machine learning models has given rise to hybrid recommendation algorithm that combines content-based filtering and collaborative filtering [11].

In fact, the shallow layers of machine learning models cannot learn the deeper features implied by users and items. Due to the excellent results of deep learning in hidden feature extraction, many scholars have introduced deep learning into the hidden feature learning of recommendation systems [12]. Salakhutdinov et al. [13] first introduced deep learning to learn the implicit factors of users and items in recommender systems and proposed a restricted Boltzmann machine- (RBM-) based collaborative filtering algorithm. Yu [14] used SVM to improve the traditional CF method and made progress for the study of cross-domain collaboration, but he did not take the underlying information into account. Wang et al. [15] directly used convolutional neural networks (CNN) and DBN to obtain the implicit factors from content information, which is only applicable to music data, since it only considers the implicit factors of items. Kim [16] used convolutional neural networks to extract potential features of relevant auxiliary information and then combined them with a probability matrix model to make recommendations, but many of the hyperparameters need to be adjusted manually. Chen Da also applied Deep Belief Network (DBN) [17] to recommender systems earlier and proposed a new deep hybrid recommendation model. This approach intrinsically suffered from the data sparseness problem. To address this issue, Wang [18] introduced collaborative deep learning (CDL) model by integrating the probabilistic topic model with the collaborative filtering technique. Zheng [19] proposed DeepCoNN to exploit two CNN networks to jointly model both the user and item reviews for improving rating prediction. Covington et al. [20] highly summarized the great impact of deep learning techniques on YouTube video

recommendation system. The YouTube system consists of two neural networks, one for generating candidate sets and one for ranking, and the two neural networks can work independently without affecting each other, and the ranking layer can not only use candidate sets but also add other datasets. Experiments have proved that this two-stage information retrieval method can effectively improve the recommendation effect and bring better user experience to users while handling large amount of data very well.

Although deep learning techniques have made certain achievements in the field of recommendation algorithms, there is still a great deal of improvement in the application of deep learning techniques in the field of personalized recommendation algorithms, which also presents new challenges and opportunities for related research work.

3. Comprehensive Recommendation Model IA-CN

In recommendation systems, multiple recommendation algorithms are often combined in different stages of the recommendation system to meet the diversity of recommendation results. In order for the system to make recommendations to users under low-latency requirements based on items that are of interest to users, the four following steps are generally required. The details are shown in Figure 1. The recall phase is mainly responsible for triggering as many correct results as possible from the full information collection and returning the results to the "sort." The sorting stage is mainly responsible for selecting the feature set that matches the consumption at this stage from the screening set as complete as possible. The fine ranking stage is mainly responsible for scoring and sorting the current set through model training from the feature set, so as to obtain the recommendation result. In the mechanism strategy stage, there will be deviations in estimates for items that have not been shown, and there may be problems. Therefore, at this stage, it is necessary to rely on rerank to filter and screen items through various recall methods.

3.1. Recall Processing-Description of Hybrid Collaborative Filtering Algorithm. In the recall phase, we will improve the traditional collaborative filtering algorithm. Collaborative filtering is a widely used technology in the category of recommender systems, and the approach of using neighbors is more explanatory than the approach based on models. In the nearest-neighbor collaborative filtering algorithm, the most important task is to find the k users/items closest to the target user/item in the entire space. The core is to calculate the similarity. The existing basic methods are all based on similarity calculation under vectors, that is, calculating the distance between vectors, the closer the distance, the greater their similarity. In recommendation, we can calculate the similarity between users by taking the user's preference for all items as a vector or use all users' preferences for an item as a vector to calculate the similarity between items. Among them, the most common method is cosine similarity formula.

$$\operatorname{sim}(u,v) = \frac{\sum_{i\in I} R_{u,i}\cdot R_{v,i}}{\sqrt{\sum_{i\in I} R_{u,i}^2}\cdot\sqrt{\sum_{i\in I} R_{v,i}^2}}. \tag{1}$$

In the above formula, what is sought is the similarity between user u and user v, I is an item that both users have rated, $R_{u,i}$ represents the score of user u on item i, and $R_{v,i}$ represents user v's score on item i.

Pearson's correlation calculation formula is as follows:

$$\operatorname{sim}(u,v) = \frac{\sum_{u\in U}\left(R_{u,i}-\dot{R}_i\right)\cdot\left(R_{u,v}-\dot{R}_v\right)}{\sqrt{\sum_{u\in U}\left(R_{u,i}-\dot{R}_i\right)^2}\cdot\sqrt{\sum_{u\in U}\left(R_{u,v}-\dot{R}_v\right)^2}}. \tag{2}$$

In the above formula, U represents the set of users who have scored, $\dot{R}_i$ represents the average score for item i, and $\dot{R}_j$ represents the average score for item j. The Pearson correlation coefficient is used to calculate the closeness of the connection between two users, and the value is [−1,1].

The collaborative filtering prediction scoring formula is as follows:

$$R_{v,i} = \overline{r_v} + \frac{\sum_{u\in U}\left(r_{u,i}-\overline{r_u}\right)\cdot\operatorname{sim}(u,v)}{\sum_{u\in U}|\operatorname{sim}(u,v)|}. \tag{3}$$

In the above formula, $R_{v,j}$ represents the predicted value of user v for the unrated item i, and v represents the set of users who have item i to be scored in the nearest neighbor.

This paper uses hybrid collaborative filtering to recall the dataset. The purpose is to obtain a dataset that is easy to classify. The detailed algorithm description is as in Algorithm 1.

3.2. Rough Sorting Model Based on XGboost Classification Method. As a boosting method [21], XGBoost has a good performance in improving the accuracy of the classification algorithm. The reason is that its internal decision tree uses regression trees, and the use of parallelization technology can improve the efficiency of learning. It has not only the advantages of fast speed and good results but also a series of advantages such as large-scale data and custom loss functions. It is an additive model composed of a series of base models. Assume that we are training the tree model for the t-th iteration, and it is expressed by $f_t(x)$; then we have the following formula:

$$Y_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = y_i^{(t-1)} + f_t(x_i). \tag{4}$$

In the above formula, $Y_i^{(t)}$ represents the prediction result of sample i after the t-th iteration, $y_i^{(t-1)}$ represents the prediction result of the $n-1$-th tree, and $f_t(x_i)$ represents the model of the t-th tree.

The definition of XGBoost is as follows.

Define the complexity of the tree:

$$f_t(x) = w_{q(x)}, \quad w\in R^T, q\in R^d \longrightarrow \{1,2,3\ldots,T\}. \tag{5}$$

In the above formula, q represents the structure of the tree, and w represents the weight of the leaves.

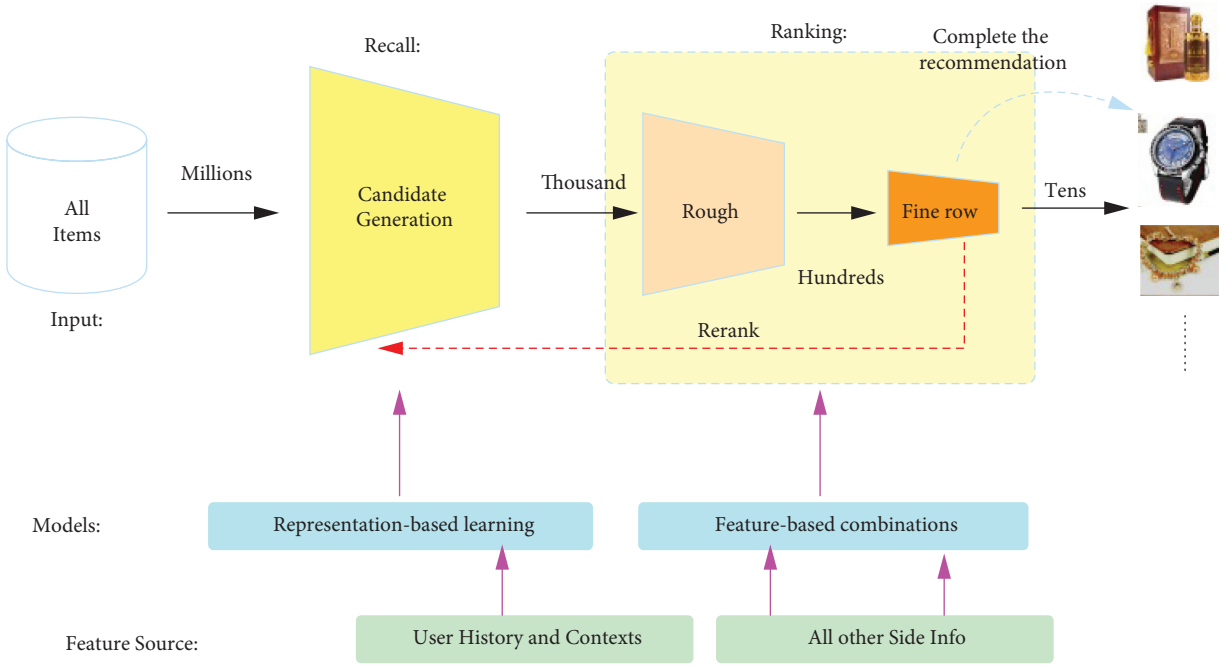

Figure 1: Recommended model process.

Improve the complexity function:

$$\varnothing(f_t) = \gamma^T + \frac{1}{2}\tau\sum_{j=1}^{T} w_j^2. \tag{6}$$

In the above formula, T is the number of leaf nodes, w_j^2 represents the L2 modulus square of the output score on each leaf node, and γ and τ control the proportion of this part in the final model formula. In this way, the complexity of the model can be measured, thereby effectively controlling overfitting.

Perform the objective function:

$$\text{obj}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}^{(t-1)} + f_t(x_i)\right) + \varnothing(f_t) + \text{constant}. \tag{7}$$

Taylor rewrite the objective function:

$$\text{obj}^{(t)} = \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \varnothing(f_t) + \text{constant}. \tag{8}$$

In the above formula, l is the loss function, and $\varnothing(f_t)$ is the regularization term. *constant* is a constant item. Derivative and rewrite the above formula, taking the square loss function (formula (9)) as an example:

$$l\left(y_i, \hat{y}^{(t-1)}\right) = \left(y_i - \hat{y}^{(t-1)}\right)^2. \tag{9}$$

Since it is actually a known value $\hat{y}^{(t-1)}$ at the t-th step, $l(y_i, \hat{y}^{(t-1)})$ is a constant, which will not affect the optimization of the function. Therefore, removing all the constant terms, the objective function is obtained as follows:

$$\text{obj}^{(t)} \approx \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \varnothing(f_t). \tag{10}$$

So we only need to find the value of the first and second derivatives of the loss function at each step (because $\hat{y}^{(t-1)}$ in the previous step is known, so these two values are constants) and then optimize the goal function; you can get $f(x)$ of each step and finally get an overall model according to the additive model.

For the set preprocessed by Algorithm 1, we train the XGBoost model to add descriptive tags to the original dataset to achieve the purpose of preparing for classification. The tags are for user and product datasets. The purpose of this is to reduce the complexity of scoring model training and improve the recommendation accuracy of the algorithm. The detailed algorithm description is given in Algorithm 2.

3.3. Classification Algorithm Description. Recall processing and classification are to improve the efficiency of the improved deep learning model recommendation algorithm. In actual model deployment, recall and classification do not necessarily have to be processed in real time. The classification algorithm is based on the user's past behavior, score, features, and user interaction, interuser relations, and interuser project relations, and calculate the similarity separately, build a similarity matrix, calculate the similarity of each user to other users and items, cluster the calculation results, form the nearest neighbor set, and then use the XGBoost algorithm for classification, find out the samples with incorrect classification, calculate the classification error rate, update the weights and learning rate, and reset the weights to achieve the effect of improving the accuracy of the algorithm classification.

3.4. Refinement: An Improved CNN Model Based on the Attention Mechanism. For the processing of fine ranking, we propose a new deep network-CNN improved network based on the attention mechanism (IA-CN), which is used for user's real-time recommendation. Figure 2 shows the model architecture of IA-CN. It first includes an embedding layer, which embeds the hotspot vector of the original action feature into the dense vector, and then a fully connected layer composed of two different feature spaces. Then, the convolutional layer is applied to the user word vector and the item word vector convolution to generate new features to model and capture the deeper and abstract nonlinear interaction relationship between the user and the item. Then, the attention pooling layer can fuse the output from the convolutional layer. Finally, the prediction layer will give the ratio of recommended predictions, which is used to rank the recommended list. Next, we will introduce the various components of IA-CN in detail.

3.4.1. Embedding Layer. The embedding layer is to extract a set of sequence words from the user's comment information and item description information and map them to an n-dimensional matrix. Word vector embedding can be used to construct the word vector matrix, so as to learn its semantic information. Specifically, the recent comments of user u are divided into a single document d, which contains a total of n words. Then, a word vector matrix V is constructed for user u, and the rules are shown as follows.

$$V^u_{1:n} = \complement(d^u_1) \oplus \complement(d^u_2) \oplus \complement(d^u_3) \oplus \ldots \oplus (d^u_n), \tag{11}$$

where d^u_n represents the n-th word in document d, the preprocessing function $\complement(d^u_1)$ returns the c-dimensional word vector of the corresponding word d, and $\oplus$ is the concatenation operation. Through matrix V, the order of words can be maintained. This article uses Word2vec [22] to initialize the embedding layer on a large Google news corpus.

3.4.2. Convolutional Layer. After being processed by the embedding layer, user comment data and item description data will enter the convolutional layer part. The convolutional layer contains m neurons, which generate new features by performing convolution operations on the word vector V of user u and the word vector of item v. Taking the user as an example, the size of filter K_i of each neuron i in the convolutional layer is set to t; that is, the filter operates on t words and then performs a convolution operation with V. The specific operation is shown in the following formula:

$$Z_i = f\left(V^u_{1:n} * K_i + b_i\right). \tag{12}$$

In the above formula, f is the activation function, $*$ is the convolution operation, and b_i is the bias term. As for the loss function used in this article, the Relus function is used as the loss function, because it performs better in training speed [23].

3.4.3. Attention Layer. The traditional convolutional neural network has the characteristics of translation invariant weight sharing. It also has certain disadvantages. For example, although the dimensionality reduction output can be easily achieved during the experiment, it will also cause a lot of valuable information to be lost, and the connection between the part and the whole will be ignored. Therefore, this paper tries to combine the attention mechanism with the CNN network and at the same time pay attention to the comment text and item description information and jointly model the user's behavior and item attributes. The pooling technology of the attention mechanism is used to model the convolved word vectors, which can extract more prominent features. The word vector in the comment or description text is mapped to the corresponding real-valued vector by the attention network, and then the normalization operation is performed to obtain the weight of the vector. As the final pooling result, attention should be paid to the connection between the whole and the parts.

The specific pooling strategy uses a weighted average strategy to replace the average and maximum and minimum strategies. The weighted average result of all word vectors in the auxiliary text is regarded as the vector representing the auxiliary text. The weighted weight corresponds to each auxiliary text. The importance of words in corresponding comments and descriptions is the attention factor. For a given auxiliary text vector sequence $V = (v_1, v_2, v_3, \ldots, z)$, the process of describing the attention-based pooling technique is shown in the following formula:

$$v = \sum_{l=1}^{L} a_l v_l. \tag{13}$$

In the above formula, a_l is the weight, that is, attention, which is obtained by calculating each word in the sequence through the normalized exponential function. It is obtained by the following formula:

$$a_l = \frac{\exp(e_l)}{\sum_{s=1}^{L} \exp(e_s)}. \tag{14}$$

In the above formula, exp is an exponential function with e as the base, and e_l is the input of normalization processing, which is formed by an attention network α. Take the user as an example; the entire pooling algorithm is as follows.

In the algorithm description,

Step 1 initializes the attention weight matrix A to facilitate the subsequent splicing of the matrix,

Step 2 is start the loop from user U. Take out the word vectors one by one from all the word vectors in it,

Step 3 uses the modified cosine similarity as the attention network to calculate the weight of each word in all words,

Step 4 performs normalization processing to obtain the final attention weight, and

Step 5 is where the attention weights processed in step 4 are joined together, and finally the pooled word vector matrix o is obtained by weighted average, which is used as the input of the next layer.

Input: User feature mount *U feature.* product feature matrix *I feature,* user rating matrix *Rp,* user list *uL* and product list *uI*
Output: Residual data set Data = {data1, data2, . . . ,} (data = $\{x1, x2, R_{iu}\}$)
(1) Init Empty similarity matrix sim *RU* and sim *RI* //REviewed users and products similarity matrix;
(2) Search the user list List*U* and the product List List*I* in *Rp*;
(3) Construct. List *U* and ListI into pairwise tuples set $U = \{(u1, u2), (u1, u2), \ldots,\}$ and set$I = \{(i1, i2), (i1, i3), \ldots,\}$;
(4) **for** *u* in set*U* and *i* in set *I* **do**
(5) Compute sim (ut, ux) and sim (it, ix);
(6) sim add in sim *RU*, sim *RI*;
(7) Calculate all users similarity matrix Sim RU_* and all items similarity matrix sim *RI* ∗
(8) **for** *tt* in R_p **do**
(9) Define polarity score: $Y_{uk}^{it}(u \in Ul, i \in uI)$;
(10) Search the K-clustering U $uS = \{u1, \ldots, uk, uk + n\}$ and T-clustering-I $iS = \{in, \ldots, it, it + n\}$ in SimRU SimRI
(11) Compute the rating $Y * (i - > us)$ of all users of uS on the product It//user collaboration;
(12) User Uk's rating $Y * (u - > is)$ on the products in the iS collection of all fields//item collaboration;
(13) Query *Rp*, construct $Y_{uk}^{it}(i - > us) *$ and $Y_{uk}^{it}(u - > is)$ respectively;
(14) Select relevant users and products are selected to construct a real score matrix $Rp_{uk}^{it} *$
(15) repeact//after multiple iterations, find the Wrong samplesby updating the learning *r* ate and weight coefficients
(16) Package data = $\{Y_{uk}^{it}(i - > us) *, Y_{uk}^{it}(u - > is), Rp_{uk}^{it} *\}$;
(17) Add data in Data;
(18) **until** *Get all data*
(19) **final**;
(20) **return** *Data*;

Algorithm 1: Hybrid collaborative filtering, pretreatment.

Input: user list *uL* and product list *iL*, Residual data collection Data
Output: User description category *US*, product description category *IS*
(1) Init sample same training weight *w*
(2) **for** data in Data **do**
(3) Use XGBOOST to iterate *m* times to calculate the error rate through $\text{err}_m = (\sum wiI(y_i \neq G_m x_i)/\sum w_i)$
(4) Update the learning rate according to $a_m = (1/2)\log((1 - \text{err}_m)/\text{err}_m)$
(5) Reset the weight of the nth sample as $w_0 = w_i * e^{a_{m*1}}$
(6) **repeat**//after multiples iterations find the wrong samples by updating the learning rate and weight coefficients
(7) Constructs a regression tree and input the data set as the root node in the form of a label;
(8) Build the XGBoost model according to the objective function of formulas (3)–(6)
(9) Defined the logistics loss function $l(yi, \widehat{yi}) = yiIn(1 + e^{-\widehat{yi}}) + (1 - yi)IN1 + e^{\widehat{yi}}$
(10) Obtain the second-order partial deivative of the loss function update the learning rate
(11) **until** N;
(12) Add label description in US and IS
(13) **Final**;
(14) **Return** US, IS

Algorithm 2**:** XGBoost classification and labeling.

3.4.4. Prediction Layer. The rating prediction layer is where the actual rating prediction task of the recommendation process occurs. The prediction layer usually accepts user/item representation $(\sigma u, \varphi i)$ and aspect representation (Qu, Qi) as input and passes them to the decomposition machine. The factorization machine accepts real-valued eigenvectors and processes pair interactions. Therefore, the overall rating can be inferred as follows:

$$\check{r}_{u,i} = \sum_{a \in A}\left(Qu \cdot Qi \cdot \sigma u\left(\varphi i^T\right)\right) + b_u + b_i + \mu, \tag{15}$$

where b_u, b_i, and μ are user, project, and global deviations, respectively. The back-propagation method can be used to learn model parameters, and the mean square error function is used as the loss function formula:

$$\text{RMSE}(X, r) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\check{r}_{u,i} - r_{u,i}\right)^2}. \tag{16}$$

In the above formula, $\check{r}_{u,i}$ represents the predicted score of the convolutional neural network and the actual score of the $r_{u,i}$ user on the item. Of course, the result of prediction layer training only represents the behavior probability of a certain user for the item. The specific recommendation process also needs to rank all behavior probabilities and output the first *n*

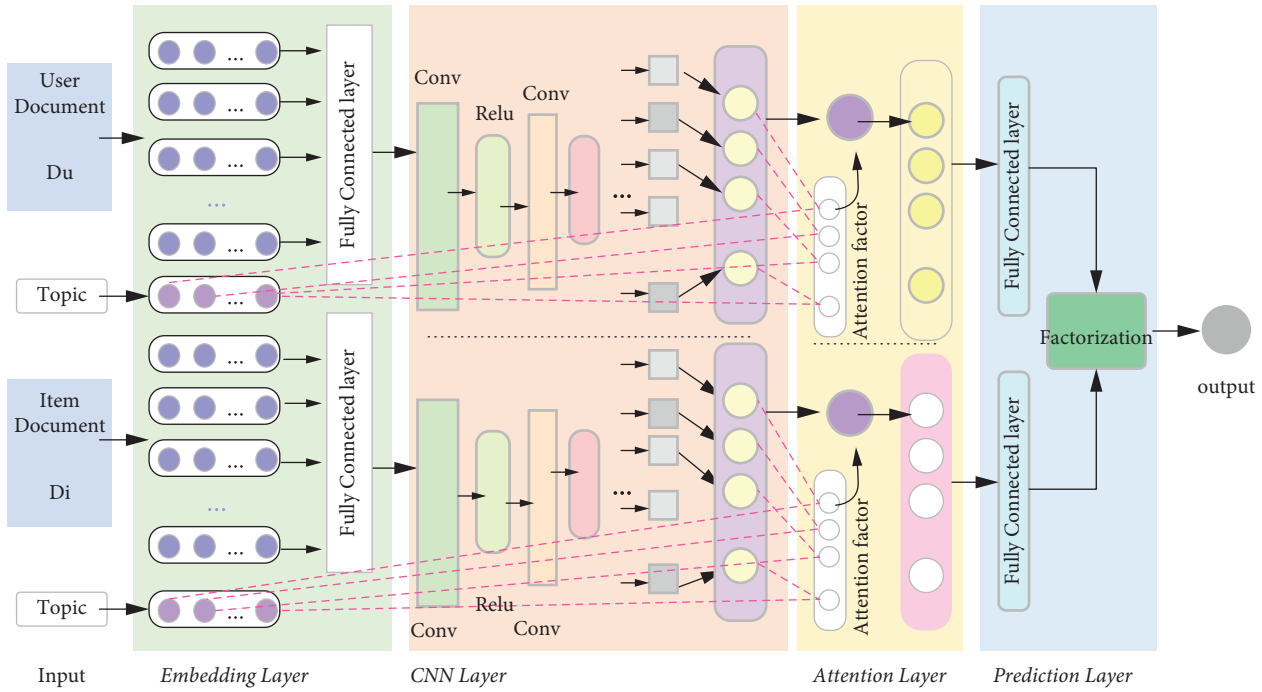

Figure 2: IA-CN architecture diagram.

results as a display. The ranking algorithm is not the focus of this article, so there is no systematic discussion.

4. Experiment

This section provides different experiments to evaluate the scientificity of the proposed IA-CN model. This experimental study aims to answer the following questions:

(i) RQ1: Can the IA-CN model perform better than the benchmark model method?

(ii) RQ2: Is the IA-CN model sensitive to hyperparameters such as degeneration, latent factors, and embedding size?

4.1. Dataset. In order to evaluate the performance of IA-CN, this article uses the current relatively complete Amazon product dataset 2018 [12], which consists of two parts: product metadata and user comment data. This dataset is the information collected by Amazon.com from 1996 to October 2018. It currently contains 29 categories of items and is the largest source dataset available for recommendation algorithm experiments.

Metadata of Amazon products includes mainly the description of the product, including information such as product number, price, feature description, product category, and related products.

As regards Amazon review dataset, in previous works [1, 8, 9, 15, 17, 19], researchers have used this data for rating prediction tasks. The main information contained in the review data includes user number, product number, user's vote on the product, rating, and review information.

Due to the long time span of the collection, the original dataset is too large, and the original data contains large numbers of unrated users and unrated products. So, in our experiment, we especially used the 5-core version of the Amazon-mini dataset (each user or item has at least 5 classified interactions). The data volume of this version is about 0.3% of the original data, including only 10-item classification; even so, the amount of data is still very large, so the experimental error caused by insufficient samples can be ignored. In the specific experiment process, before the word vector mapping, in order to reduce the time and space required for the program to repeatedly read the data, this article deletes all uncommon terms, duplicates, and irrelevant attributes. Only the data necessary for this experiment is kept, and finally 5 classification versions are got. All stop words and nonvocabulary words in the comment dataset are filtered. The dataset is divided into training set, test set, and validation set with a ratio of 8 : 1 : 1. In the parameter sensitivity experiments, we constructed 5 categorical versions of the data in categorical form and constructed 4 different sets of data (single-class item, 3-class item, 5-class item, and full).

4.2. Benchmark Model and Configuration. In order to try to restore the fairness of the experimental comparison, for the benchmark model, this paper uses the same set of software and hardware platforms for experiments. Four different benchmark models were used for comparison, namely, matrix factorization (MF) model, probabilistic matrix factorization (PMF) model, collaborative topic regression (CTR) model, and deep cooperative neural network (DeepCoNN). Among them, MF [2] and PMF [3, 8] are the most widely used and well-known standard benchmarks for collaborative filtering methods. In contrast, the effect of the

description information on the recommendation algorithm can be verified. CTR [24] is a more successful topic-based modeling method, which also uses potential topics to model ratings and reviews. DeepCoNN [19] is currently a relatively advanced deep-learning-based rating prediction model. It also uses two parallel CNN models to learn user and item representations. It uses matrix factorization in the sharing layer for rating prediction, and the IA-CN, design ideas in this article are also derived from this. For MF and PMF, we use the open recommendation system library MyMedialite to evaluate the value of the model. For CTR and DeepCoNN, we refer to the source code on GitHub and fine-tune and restore the experiment according to the relevant parameters set by the authors of the paper.

In this experiment, we first designed a parameter sensitivity experiment to preliminarily set the parameters of the model and then continuously optimized the parameters during the experiment, especially the characteristic dimensions of users and objects and the number of nuclei in the convolutional neural network. The user and item feature dimensions are set to 10 to 100 and the number of cores is set to 10 to 500. Finally, for optimal performance, the feature dimension of users and objects is set to 50, the number of cores is set to 100, the dimension of word embedding is 300, the learning rate is 0.001, the sliding window parameter size is set to 3, the batch size is set to 100, and the dropout parameter is set to 0.6.

4.3. Experimental Environment and Measurement Labelling. The hardware configuration of the deep learning model experimental platform in this article is Intel i7-10700 CPU, the graphics card is GTX 2070, the memory is 32 G, the operating system is Windows 10, and the model is implemented based on the Python programming language and the TensorFlow framework. The optimizer uses AdamOptimizer, the default learning rate is 0.001, and all models are trained until convergence.

The metric experiment uses the root mean square error (RMSE as formula (16)) to measure the accuracy of the prediction score, because the root mean square error can measure the deviation between the predicted value and the true value, which is defined as follows: the smaller the calculation result, the better the prediction accuracy and the better the recommendation effect.

The main goal of the recommendation system is to generate top n recommendations for end users, so we use Recall@N (formula (17)) and Precision@N (formula (18)) indicators to evaluate our proposed IA-CN model. They are expressed as follows:

$$\text{Recall@}N = \frac{A \cap B}{B}, \tag{17}$$

$$\text{Precision@}N = \frac{A \cap B}{N}, \tag{18}$$

where A represents the number of items the user likes in top-N and B represents the number of various items used by the user. In order to evaluate the proposed hybrid model, we arranged the predicted scores of all products for each user and recommended the top n recommendation lists to each user.

4.4. Model Training. Because the model parameters are an important factor that affects the experiment, we tune and compare the parameters in the convolutional neural network in the four following aspects. Finally, the final model was constructed using experimental data.

4.4.1. Impact of Dropout. Dropout has been proven to be an effective method to solve the problem of neural network model overfitting [25]. Therefore, in order to check the impact of dropout, we change dropout with different values.

Figure 3 shows that, after setting dropout, the impact on reducing prediction errors is very obvious. It can be observed from Figure 3 that there is a significant improvement of the model across all the datasets. However, the gain of RMSE varies on different datasets. The model records the best results between 0.55 and 0.65 in most of the datasets, while it records the worst performance when the dropout is not used. It can also be observed that, compared with the small-scale dataset, the effect of dropout in a relatively large dataset is less obvious. This is consistent with previous observations [18] and reiterates that dropout is more important for small-scale datasets. Therefore, as shown in Figure 3, the replacement value of the model is between 0.55 and 0.65.

4.4.2. Embedding Dimension. We set different word embedding dimensions {50, 100, 200, 300, 400, 500, 600} to verify the sensitivity of the model to word embedding dimensions. The result is as shown in Figure 4.

In the figure, it can be easily observed that the model has greater sensitivity within 500 word embedding dimensions. In the single-category data, the results of sensitivity are more obvious. This is not difficult to analyze.The reason is that the frequency of words appearing in the single class item data is not rich, which is why the full data set model is more accurate. The results also show that, in all datasets, the performance is the highest around 300 dimensions, and it remains relatively stable above 400. This is enough for the model's sensitivity to the word embedding dimension. Therefore, further use of larger values did not show a significant improvement in the model. In the end, we choose 300 as the word embedding dimension in the experiment.

4.4.3. Impact of Length of the Document. In order to evaluate the impact of the length of the document information on the model, we set up a verification experiment on 5-class data. The accuracy and training time of the model are recorded by adjusting the length of the document. It is not difficult to see from Figure 5 that when the maximum length of the document is 200, the model has better performance, and the time spent at this time is also more reasonable; that is, the model is more stable.

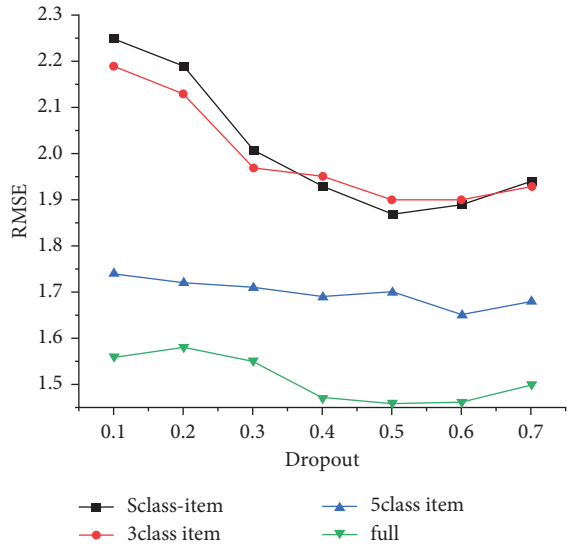

Figure 3: Impact of dropout.

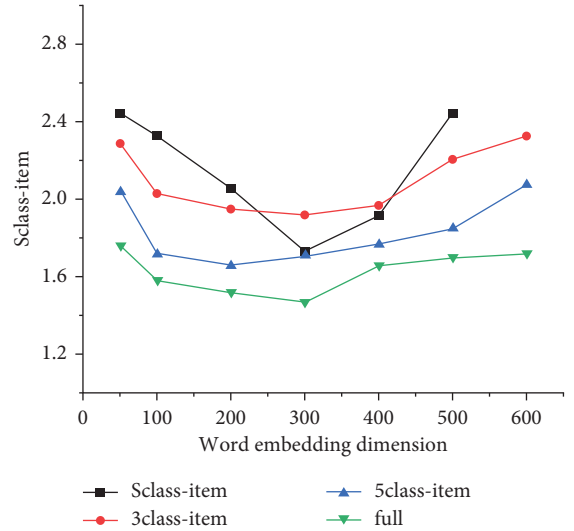

Figure 4: Impact of embedding dimension.

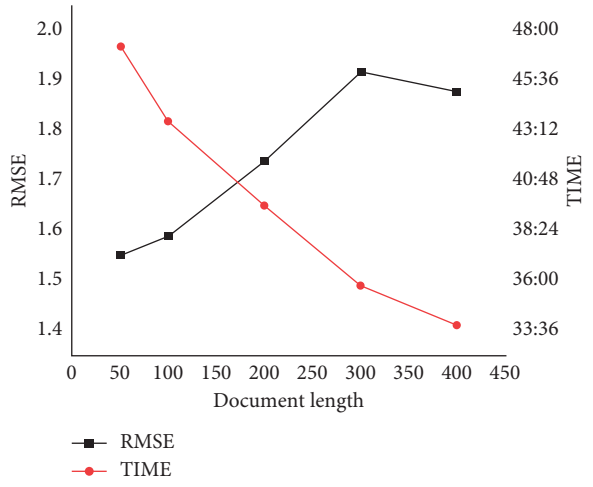

Figure 5: Impact of document length.

Table 1: RMSE compared with different test sets

Methods	Mean square error under different datasets				Total RMSE
	10 W	20 W	30 W	40 W	
Our method	**2.044**	**1.551**	**1.471**	**1.372**	**1.294**
MF	2.741	2.346	2.212	1.784	1.597
HMF	2.544	2.147	2.030	1.710	1.681
CTR	2.410	2.134	1.904	1.611	1.511
DeepCoNN	2.142	1.644	1.579	1.507	1.471

It can also indicate that the additional document information can be accurately used for the document latent vector until the maximum length of 400 is reached, and no further information can be obtained even if the document length is greater than 400. That is because while increasing the length of the document, the time it takes to process the document information will increase, and the model needs more time for effective training. Therefore, when processing document information, it is important to consider the trade-off between training time and document length.

4.5. Experimental Comparison and Analysis. In order to verify the effectiveness of the proposed algorithm IA-CN, this paper selects a total of 4 models for comparison experiments under the condition of the above super-parameters, namely, the matrix factorization (MF) model, the hybrid matrix factorization (HMF) model, the collaborative topic regression (CTR) model, and the deep cooperative neural network (DeepCoNN); they are all classic models in recommendation algorithms. Among them, the matrix factorization model and the hybrid matrix factorization model belong to model-based recommendation algorithms. By comparing them, the effect of review text and description information on the recommendation algorithm can be verified, especially when the score matrix is relatively sparse; the collaborative topic model belongs to topic-based model recommendation algorithms. Most recommendation algorithms that consider review text are based on topic models. Compared with them, the advantages of the proposed algorithm can be verified. In particular, the word embedding vector technology is used to replace the topic model in the preprocessing step, focusing on the performance improvement brought by the order of words in the text; DeepCoNN is a typical representative of collaborative filtering algorithms based on deep learning, and the algorithm proposed in this chapter is improved on the basis of the deep cooperative neural network (CNN), and it is also necessary to make a vertical comparison with it. This experiment was repeated three times, and the average of the three results was taken as the final result. Ultimately, experiments on Amazon product evaluation datasets of different data sizes yielded the experimental results as shown in Table 1.

Of course, we also chose the Amazon product evaluation dataset to compare the more convincing Recall@N and Precision@N to compare our method with the benchmark

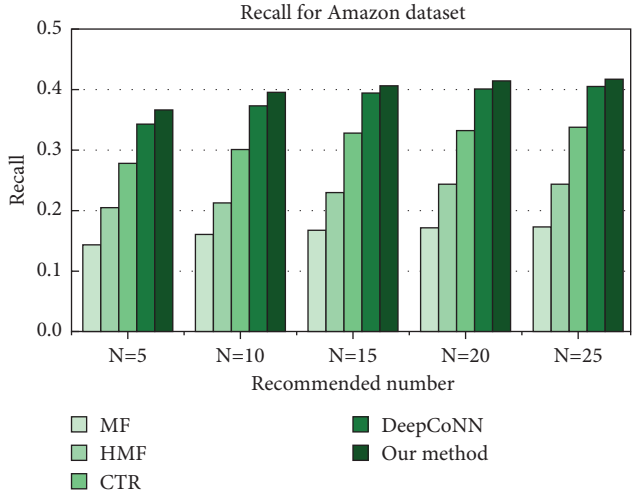

Figure 6: Recall@N comparison.

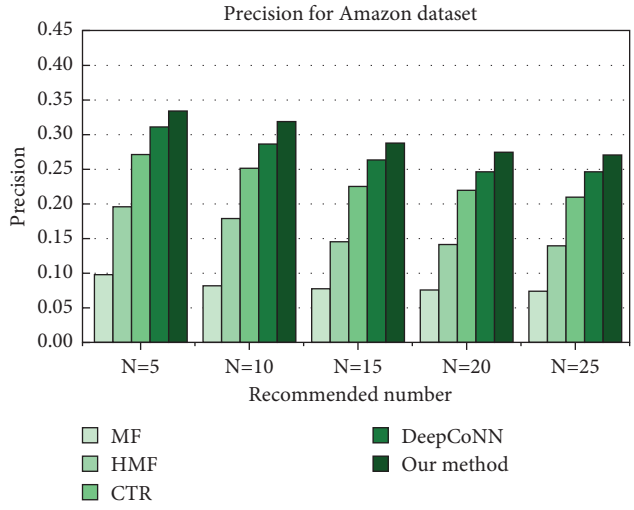

Figure 7: Precision@N comparison.

experiments MF, HMF, CTR, and DeepCoNN. The results of Figures 6 and 7 show that the value of Recall@N gradually increases with the increase of n. Compared with our method in the benchmark experiments of Recall@N and Precision@N, our method obtains relatively higher results.

The running time is considered as a nonessential evaluation criterion; we designed the following experiments: the 10w-full dataset was trained with different models, and then 20 different samples were randomly selected from the test machine for testing, and the time comparison graph as

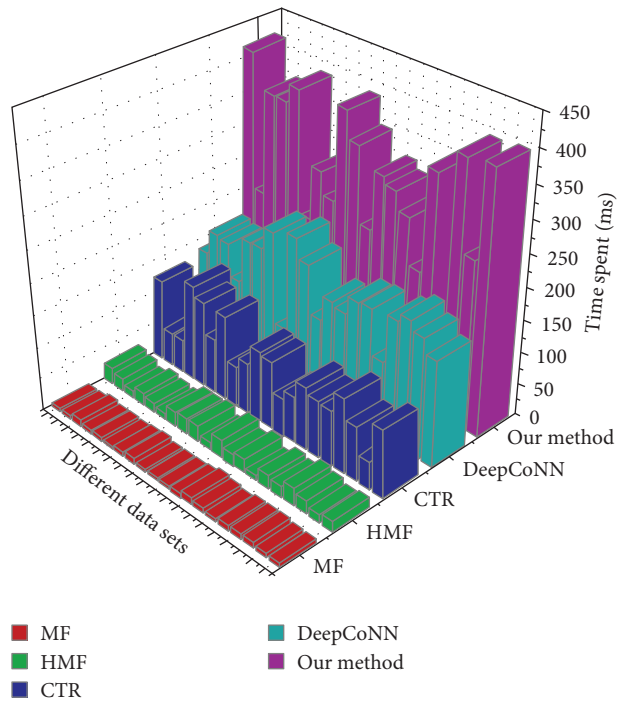

Figure 8: 10w-full running time comparison chart.

shown in Figure 8 was obtained; it can be seen that MF and HMF took less time in the same test environment; the reason is that the machine learning method learns fewer layers. Meanwhile DeepCoNN and our method in deep learning took more time, and because the algorithm in this paper added attention mechanism and a factorization, the time it takes is higher than the former However, even so, the recommendation can be completed within 0.4 s, which is obviously an acceptable recommendation time range, so our algorithm can be considered correct and reasonable.

5. Conclusion

Aiming at the data sparseness, cold start of items, long tail items, and other related issues in existing personalized item recommendation algorithms, the paper studies the use of existing recommendation techniques and deep learning methods that have powerful fitting capabilities and can perform complex tasks. The advantages of nonlinear mapping and its powerful representation ability can alleviate the impact of data sparseness and item cold start in these existing item personalized recommendation systems. The work done in the paper designs an integrated strategy model of recommendation algorithms and generalizes and classifies the output of the model and then trains an improved CNN deep learning model and finally verifies the feasibility of the scheme through experiments. Although the experimental results show some progress, the research and application of deep learning methods and technologies in recommendation systems are still in the development stage. The hybrid recommendation scheme based on deep learning proposed in this article still has phased and partial problems, and it remains to be used in actual recommendation systems.

Data Availability

The dataset used in this paper can be found at https://nijianmo.github.io/amazon/index.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] X. Zeng, Y. Yang, S. Wang, T. He, and J. Chen, "A hybrid recommendation algorithm based on deep learning," *Computer Science*, vol. 46, no. 01, pp. 126–130, 2019.

[2] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix," *ACM Computing Surveys*, vol. 47, no. 1, pp. 1–45, 2014.

[3] S. Ruslan and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 880–887, Bled, Slovenia, 2008.

[4] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pp. 175–186, ACM, October 1994.

[5] B. Sarwar, G. Karypis, and J. Konstan, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295, Hong Kong, China, 2001.

[6] X. Yu, F. Jiang, J. Du, and D. Gong, "A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains," *Pattern Recognition*, vol. 94, pp. 96–109, 2019.

[7] S. Funk, "Netflix update: try this at home," 2006, https://llsifter.org/~simon/journal/20061211.html.

[8] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2008.

[9] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the ACMSIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 426–434, ACM, August 2008.

[10] M. Yu, T. Quan, Q. Peng, and X. Yu, "A model-based collaborate filtering algorithm based on stacked Autoencoder," *Neural Computing and Applications*, pp. 1–9, 2021.

[11] S. Ahmadian, P. Moradi, and F. Akhlaghian, "An improved model of trust-aware recommender systems using reliability measurements," in *Proceedings of the 2014 6th Conference on Information and Knowledge Technology (IKT)*, pp. 98–103, IEEE, Shahrood, Iran, May 2014.

[12] Z. J. Sun, L. Xue, and Y. M. Xu, "Overview of deep learning," *Jisuanji Yingyong Yanjiu*, vol. 29, no. 8, pp. 2806–2810, 2012.

[13] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the International Conference on Machine Learning*, pp. 791–798, ACM, June 2007.

[14] X. Yu, Y. Chu, F. Jiang, Y. Guo, and D. Gong, "SVMs classification based two-side cross domain collaborative filtering by inferring intrinsic user and item features," *Knowledge-Based Systems*, vol. 141, pp. 80–91, 2018.

[15] P. Chiliguano and G. Fazekas, "Hybrid music recommender using content-based and social information," in *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech*

and Signal Processing (ICASSP), pp. 2618–2622, IEEE, Shanghai, China, March 2016.

[16] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 233–240, Boston, MA, USA, September 2016.

[17] D. Chen, *Research on Recommendation System Based on Deep Learning*, Beijing University of Posts and Telecommunications, Beijing, China, 2014.

[18] H. Wang, N. Wang, and D. Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1235–1244, 2015.

[19] L. Zheng, V. Noroozi, and P. S. Yu, "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, 2017.

[20] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the ACM Conference on Recommender Systems*, pp. 191–198, ACM, 2016.

[21] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, San Francisco, CA, USA, 2016.

[22] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 977–984, ACM, June 2006.

[23] F. Tahmasebi, M. Meghdadi, S. Ahmadian, and K. Valiallahi, "A hybrid recommendation system based on profile expansion technique to alleviate cold start problem," *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 2339–2354, 2021.

[24] C. Wang and D. M. Blei, "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, San Diego, CA, USA, August 2011.

[25] M. Kirienko, "Convolutional neural networks promising in lung cancer T-parameter assessment on baseline FDG-PET/CT," *Contrast Media & Molecular Imaging 2018*, 2018.