In-depth Recommendation Model Based on Self-Attention Factorization

Hongshuang Ma[1], and Qicheng Liu[1*]
[1] School of Computer and Control Engineering, Yantai University,
Yantai 264000, Shandong, China
[e-mail: 15666750583@163.com, ytliuqc@163.com]
[*]Corresponding author: Qicheng Liu

Received March 10, 2022; revised November 9, 2022; accepted February 20, 2023; published March 31, 2023

Abstract

Rating prediction is an important issue in recommender systems, and its accuracy affects the experience of the user and the revenue of the company. Traditional recommender systems use Factorization Machines for rating predictions and each feature is selected with the same weight. Thus, there are problems with inaccurate ratings and limited data representation. This study proposes a deep recommendation model based on self-attention Factorization (SAFMR) to solve these problems. This model uses Convolutional Neural Networks to extract features from user and item reviews. The obtained features are fed into self-attention mechanism Factorization Machines, where the self-attention network automatically learns the dependencies of the features and distinguishes the weights of the different features, thereby reducing the prediction error. The model was experimentally evaluated using six classes of dataset. We compared MSE, NDCG and time for several real datasets. The experiment demonstrated that the SAFMR model achieved excellent rating prediction results and recommendation correlations, thereby verifying the effectiveness of the model.

Keywords: Self-attention network, deep learning, recommendation model, review text.

This work was supported by the National Natural Science Foundation of China under Grant 62172351.

http://doi.org/10.3837/tiis.2023.03.003 **ISSN : 1976-7277**

1. Introduction

The rapid development of the Internet has satisfied users' demand for information, but has also brought about information overload [1]. Faced with massive amounts of data, it is difficult for users to find the information that they need. Therefore, the Recommender System [2] merges the required times. A Recommender System can select the most valuable information from a large amount of information and provide users with personalized services to improve their knowledge acquisition efficiency [3]. Although the traditional content-based recommendation and Collaborative Filtering (CF) recommendation algorithm is simple and effective [4][5], it has the problems of a cold start and missing rating matrix [6], which degrade the recommendation performance.

With the application of Deep Learning in Natural Language Processing (NLP), many researchers and industry professionals have begun applying Deep Learning to Recommender Systems [7][8][9]. Deep Learning methods can effectively learn the basic features of datasets. Some researchers drew inspiration from neurons and constructed a method for learning tasks using neural networks. Convolution neural networks, recursive neural networks, multi-layer perceptron and other technologies were used to analyze texts, greatly improving the efficiency of feature extraction [10][11][12]. Some scholars have used clustering algorithms to study document similarity, which can be widely applied to information retrieval and recommendation models [13][14].

Besides, the attention mechanism in Deep Learning can be used to distinguish the importance of the different features. By introducing an attention mechanism, the weights of irrelevant parts can be reduced [15]. From the perspective of the interpretability of the attention mechanism, it allows the direct inspection of the inner workings of the Deep Learning system. It achieves the effect of enhancing the interpretability of deep models by visualizing the attention weights of inputs and outputs [16]. Therefore, introducing an attention mechanism into the recommendation model can distinguish the importance of each potential factor or feature and improve its performance of the recommendation model.

In recent years, the use of review texts to enhance the interpretability of recommendation models has become a research hotspot. Review texts can reflect user preferences and explain why a high or low rating is assigned. Simultaneously, review text can compensate for the deficiency of adequate information, provide rich information for user and item modeling, and improve the recommendation effect. However, review texts have a complex structure and cannot be handled directly using recommendation models [17]. Therefore, it is essential to study recommendation models that can parse review text.

This study presents a depth recommendation model, SAFMR, based on the self-attention mechanism technology. This model uses a convolutional neural network to extract the comment features, a self-attention mechanism to automatically learn the importance of different features, and to distinguish the importance of features by assigning them different weights. The self-attention mechanism automatically assigns weights according to the relationships within the features, which can strengthen the relationships within the features and capture their correlations. Experiments were conducted using real datasets to evaluate the performance of the proposed model. Experiments show that, compared with the traditional recommendation model, it improves the accuracy of the recommendation model.

2. Related work

In the CF algorithm, the interaction between users and items is used to find similar users and items, and the similarity is then used to make recommendations. With the development of neural networks, some recommendation algorithms have merged the CF algorithms and neural networks. Some of the latest research has focused on capturing better similarities, including designing more complex network structures. For example, He et al. [18] used deep neural networks to learn interaction functions from data, in which user preferences were indirectly reflected by interaction functions. To improve the cold start problem existing in CF, the SCF model [19] uses spectral convolution operations to discover deep interaction information between the item and user to solve the cold start problem. The NGCF recommendation framework [20] can effectively integrate interactions between users and items into the embedding equation. He et al. [21] thought that feature transformation and nonlinear activation have no effect on CF and proposed a light graph neural network applied to Recommender Systems. Xia et al. [22] proposed a combination of Graph Convolution Networks (GCN) and incremental Temporal Convolutional Networks with CF to initialize user and item embeddings using the MAML model [23]. Therefore, the recommendation model accelerates its adaptation. This alleviates the cold start problem of the CF algorithm. The basic concept of the CF algorithm is that similar users have similar properties. The IMP-GCN model [24] uses user features and graph structures to identify users with similar interests and recommends products to users with similar interests. Zhang et al. [25] proposed model considers the new factor between active users and the nearest neighbor, introduces the trust network into the recommendation model, and selects the best trust path between users through algorithm integration, which improves recommendation performance.

These studies were improved based on CF algorithms. Although neural networks are used in CF, no roundup text covers the rich user preferences and item information, and the recommendation model is insufficient in terms of interpretability.

Compared with CF, using review text to predict ratings can improve the interpretability and accuracy of the recommendation model. The continuous development of Deep Learning makes it possible for recommender systems to use neural networks to process review texts. The DeepCoNN model [26] proposes the use of two Convolutional Neural Networks (CNN) to process the review text of users and items and help predict ratings. Referring to the DeepCoNN method, Chen et al. [27] used two CNNs to learn the features of user and item reviews, which were fed into an extended Latent Factor for rating predictions.

In the case of incomplete or sparse target user reviews, Wu et al. [28] proposed a PARL model that integrates the and-play model based on DeepCoNN, and uses reviews from similar users to enrich the preferences of the target users. Catherine et al. [29] indicated that the DeepCoNN model can obtain the best performance only when the sample contains the target user's reviews of the target item during the test. During the training process, reviews written by users on items were used by the DeepCoNN model to predict ratings, which was not reasonable. Therefore, a TransNet model was proposed based on the DeepCoNN model, which extends the Transform layer. The Transform layer is a fully connected layer of the L layer, as part of the network structure. The Transform layer can transform the potential features of user and item reviews into an approximate representation of target the reviews. Finally, the model uses the Factorization Machines (FM) [30] to predict ratings. The CARL model [31]was proposed to learn potential features from reviews using convolutional operations and attention mechanisms and then integrate the potential features and possible ratings into the FM model to obtain the missing ratings. However, recommendation models such as DeepCoNN and TransNet use Factorization Machines to process cross-features with the same weight for each feature, and useless features can introduce noise that affects the effectiveness of the model's recommendations.

The introduction of an attention mechanism into the recommendation model can distinguish the importance of each potential factor or feature and improve its performance of the recommendation model. Zhang et al. [32] proposed a new sequence-aware recommendation model that uses the self-attention mechanism to infer the relationship between items from the user's historical interaction, and used the self-attention mechanism to estimate the relative weight of each item in the user interaction trajectory to learn the expression of the user's short-term interest. Zhou et al. [33] proposed a TAFA model that uses attention to select comments related to a recommendation task to make recommendations.

In Deep Learning, the self-attention mechanism automatically learns the importance of different features and distinguish them by assigning different weights [34]. By combining the self-attention mechanism with the Factorization Machine to process the features, the self-attention neural networks automatically learn the dependencies within the features to improve the data representation capability and solve the deficiency of the data representation capability. Considering the advantages of the self-attention mechanism, an improved SAFMR model was proposed in this study by combining the self-attention technology.

3. Preparatory theory

3.1 Convolutional neural networks

The review text contained complex user and item features. Extracting features for rating prediction from review text requires constructing feature extraction networks, and CNN is generally used to remove essential keywords from reviews as features. Compared with manual extraction and traditional machine learning, rich semantic features can be extracted from reviews, and complex high-dimensional data can be handled in CNNs. The problem of insufficient accuracy in the manual feature extraction was avoided. At the same time, the amount of calculation is reduced in the CNN through the parameter sharing of each network layer [35]. The following section introduces various parts of the CNN in detail.

3.1.1 Embedding layer

The embedding layer of the CNN maps the review text into an $n \times k$ matrix D, where n is the number of words in the review, and k is the word vector corresponding to each word. The review text vectorization of the model is realized using the word embedding tool, which maps the linguistic information to the semantic space.

3.1.2 Convolutional layer

The convolutional layer is the primary building block used in CNN, which extracts features from the input data. The convolutional layer was composed of several convolution kernels. Through multiple convolutions, useful features are enhanced and useless features are reduced. Multiple regular convolutions can enhance compelling features and reduce useless features [36]. Feature extraction is performed using the convolution kernel law. The convolution operation of the matrix obtained by the input layer can be expressed by Eq. (1):

$$a_j = \phi(\varpi \cdot D[j : j + v - 1] + g) \tag{1}$$

where a_j is the jth feature, j=1, …, n-v+1, D [j: j+v-1] represents the window of size $v \times k$ formed by the j to j+v-1 rows of matrix D, and the width of the convolution kernel is the dimension k of the word vector. The height is v, ϖ is the $h \times k$ dimension weight matrix, g is the

bias, ϕ is the activation function, expressed as $\phi(z)=\max\{0,z\}$. According to Eq. (1), matrix *D* can be convolved to obtain *n*-*v*+1 features.

3.1.3 Pooling layer

After the features of the review text are obtained in the convolutional layer, if the classifier is trained directly using the features, it faces the challenge of huge computational effort and is prone to overfitting. To further reduce the training parameters and overfitting of the model, the features of the convolutional layers must be pooled. The pooling layer was used to reduce the dimensionality of the features, compress the data and parameters, reduce overfitting, and improve the fault tolerance of the model. The pooling layer most commonly uses Max-Pooling and Mean-Pooling, where Max-Pooling is used.

Max-Pooling filters out one of the largest features generated by each sliding window and then stitches these features together to form a vector representation, which is defined as:

$$m_j=\max\{a_j^1,a_j^2,...,a_j^{(n-v+1)}\} \tag{2}$$

where, m_j is the *jth* maximum feature generated by the sliding of the convolutional kernel. The final output is the connection of the results from the *d* convolution kernels, which is expressed by the following equation:

$$M=[m_1,m_2,...,m_d] \tag{3}$$

As can be seen from Eq. (3), Max-Pooling represents sentences of different lengths as a fixed-length vector representation. Max-Pooling was performed separately on different channels, and the pooling operation did not change the number of channels. Therefore, Max-Pooling ensures that the features are position and rotation-invariant. It also reduces the number of model parameters and the fit of the model [37].

3.1.4 The fully connected layer

The fully connected layer is the final part of the CNN. All units in each layer were fully connected to the previous layer. The primary function of the fully connected layer is to reduce the loss of the feature information.

The fully connected layer is composed of weight matrices and bias matrices, which computes the final representation of *M* of the input pooling layer as:

$$X=f(WM+b) \tag{4}$$

where *W* is the weight matrix $W\in \mathrm{R}^{d\times n}$, and *b* is the bias matrix $b\in \mathrm{R}^{n}$. The feature matrix *X* of the review sentence is obtained through the convolution processing of the CNN.

3.2 TransNet recommendation model

The TransNet recommendation model is a neural network recommendation model that utilizes reviews to improve recommendation performance [29]. The model converts the potential features of users and items into approximate representations of the target reviews and uses Factorization Machines to predict ratings.

In real-life scenarios, a recommendation means that the product is recommended to the target user before purchasing, and the user can only evaluate the product after purchasing it. Therefore, these nonexistent reviews cannot be used as inputs to the model to predict the user ratings of the product. The TransNet recommendation model proves that user reviews of the target item have high predictive value. These reviews should be effective during training and not testing. Therefore, two different neural networks were designed in the TransNet model, the target neural

network and the source neural network. The target network processes the review rev_{AB} written by the target review $user_A$ on $item_B$. The source network processes the reviews of $user_A$ on $item_B$ that do not contain rev_{AB}. The target network uses a CNN to process the target reviews and FM to predict ratings. The source network was composed of two CNNs and a Transform layer. Without rev_{AB}, the user and item reviews were processed using two CNNs. The Transform layer is an L-layer nonlinear fully connected network that converts user and item reviews into an approximate representation of the target item review [29] used in the later FM rating prediction.

3.3 Factorization Machines

Factorization Machines (FM) are supervised learning algorithms proposed by Google Researcher Steffen Rendle [31]. The FM is an ideal choice for processing tasks involving high-dimensional sparse datasets (such as click prediction and item recommendation). FM plays a role in predicting ratings in the recommendation model [38]. The FM model enhances the ability of the linear model by modeling the second-order crossover features. The equation for FM prediction rating is as follows:

$$y = \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \hat{w}_{ij} x_i x_j + w_0 \tag{5}$$

where y is the prediction rating, n is the number of features, x_i is the ith feature, w_0 is the global bias, w_i is the weight vector of the feature vector x_i, $\hat{w}_{ij}$ is the weight matrix, $\hat{w}_{ij} = v_i^{\mathrm{T}} v_j$, v_i is the implicit vector of the ith dimension feature, $w_0 \in \mathrm{R}$, $w_i \in \mathrm{R}^n$, $V \in \mathrm{R}^{n \times k}$. w_0, w_i, V are the parameters learned by the FM.

FM can handle high-dimensional sparse datasets and have applications in recommender systems and NLP fields. However, because the FM assigns the same weight to each cross-feature, useless features may introduce noise during feature selection. This affects the final prediction performance of the model [39].

4. The SAFMR Model

When traditional neural network recommendation models use FM to process features, each feature has the same weight. In the reality, different features often have different effects. For a fixed feature, not all features are useful for feature selection, and these useless features introduce noise and cause interference. The self-attention mechanism can automatically increase the weight of important features and reduce the weight of features with a low impact. Therefore, we can introduce the self-attention mechanism into the real recommendation model and learn the weights of the different features from the self-attention network.

The self-attention mechanism maps the feature matrix to the query, the key and the value matrices from the same input. When the self-attention mechanism processes a feature, it first calculates the correlation between the feature's query matrix and each key matrix, obtains the weight coefficient of the value matrix corresponding to each key matrix, and then weighs the value matrix to obtain the attention weight. The attention weight determines the features that require attention. It can be seen that the self-attention mechanism can effectively learn the internal dependencies of features and capture the internal dependencies of features by processing the feature matrix [40].

The process of using the self-attention mechanism to process the feature matrix can be divided into two steps: calculating the attention distribution of the input feature matrix and calculating the weighted average of the feature matrix according to the attention distribution.

4.1 Calculate the distribution of attention

For N input vectors $[x_1, \ldots, x_N]$, to reflect the importance of the features, it is necessary to calculate the weight of each input vector. The feature matrix $X = [x_1, \ldots, x_N]$ obtained by the CNN was linearly mapped to three different spaces by the self-attention mechanism. The query matrix R, key matrix H, and value matrix S were obtained. The matrix operations are as follows:

$$R = W_r X, H = W_h X, S = W_s X \tag{6}$$

where, W_r, W_h, and W_s are the trainable weight matrices, representing the different weight selections of feature matrix X. The parameter W of the linear transformation of R, H, and S used in Eq. (6) is different. The weight parameter of the attention mechanism is a globally learnable parameter fixed to the model. The weight parameter of the self-attention mechanism is determined by the input, such that different information in the same model has other weight parameters.

By introducing the feature-related query vector R, the correlation between each query vector and input vector can be calculated.

For each query vector $R = [r_1, r_2, ..., r_N]$, the key-value pair attention mechanism was used. After normalization, the attention distribution $\hat{X}$ was obtained, from which the weight of each feature vector was calculated. The attention distribution is expressed by Eq. (7):

$$\hat{x}_i = \frac{\exp(\frac{h_i^{\mathrm{T}} r_i}{\sqrt{k}})}{\sum_{j=1}^{N} \exp(\frac{h_j^{\mathrm{T}} r_j}{\sqrt{k}})} \tag{7}$$

In Eq (7), $\hat{X} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\}$, $H = [h_1, h_2, ..., h_N]$, k is the dimension of the word vector. After the HR matrix was multiplied, the Scaled Dot-Product model was used for scaling. The normalization process is highly susceptible to larger or smaller inputs, and it is easy to map to 0 and 1. After the normalization transformation, when the full probability is assigned to the label corresponding to the maximum value, the model is trained with factor k for scaling to prevent the gradient from disappearing during backpropagation [41].

4.2 The weighted average of the Eigenmatrix

The weighted sum provides the output based on the attention distribution $\hat{X}$. In this section, the similarity between the query vector and the known key vector is calculated separately and then assigned to the value vector as a weight, and their weighted sum is returned. Therefore, the prediction rating equation is:

$$\hat{y} = \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} s_i \hat{x}_i + w_0 \tag{8}$$

Because the FM assigns identical weights to feature interactions, it can only express the relationship between pairwise combinations of features. When the SAFMR model processes features, the self-attention mechanism provides an effective modeling method to capture global context features through triples of key, query, and value [42]. In the SAFMR model, the internal correlation of features is better learned by introducing the self-attention network, the dependence on external information is reduced, and the data are accurately expressed, thereby improving recommendation model accuracy.

The SAFMR model includes the target network and the source network, respectively. The model framework is illustrated in **Fig. 1**:

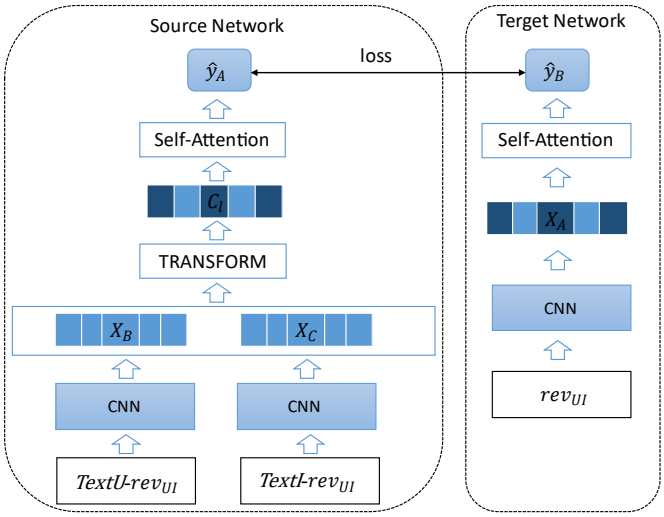

Fig. 1. The SAFMR Model

As shown in **Fig. 1**, the target network uses the CNN text processing layer (*conv*) to process the reviews that $user_U$ has written about $item_I$, rev_{UI}. Review text is introduced to the input layer of the CNN, and the review information is mapped to the word-embedding matrix D_A. Then, the feature matrix X_A of the target network is obtained by Eqs. (1)-(4). The self-attention mechanism is used for X_A using Eq. (6). By decomposing the feature matrix X_A of the target network, three different sub-matrices are obtained: query matrix R_A, key matrix H_A, and value matrix Y_A. The attention distribution is calculated according to Eq. (7), and the attention distribution matrix $\hat{X}_A$ of the target network is obtained. Eq. (8) is applied to the target network attention distribution to obtain the prediction rating $\hat{y}_A$.

The source network contained two CNN text processing layers. The reviews of $user_U$ and $item_I$ ($Text_U$, $Text_I$), which do not include rev_{UI} are processed separately. The outputs of the word-embedding matrices D_B and D_C are mapped by the CNN.

The eigenmatrices X_B and X_C of the source network are obtained through the convolution processing of Eqs. (1)-(4).

$$C_0 = [X_B X_C] \tag{9}$$

where X_B and X_C are the user and item review feature matrices that do not contain rev_{UI}. The two feature matrices X_B and X_C are horizontally spliced according to the dimension using Eq. (9), and a new matrix C_0 is obtained.

There is also a Transform layer in the source network that covers C_l into the target network review matrix X_A approximation. The Transform layer is an L-layer nonlinear fully connected layer network. Each layer had a weight matrix G_l and bias gl. The weight matrix initially followed a truncated normal distribution with the mean value of 0 and a standard deviation of 0.1, and all biases were initialized to 0.1. The transfer equation of layers l and l+1 is:

$$C_l = \phi(C_{l-1} G_l + gl) \tag{10}$$

where C_l is the output of the lth layer of the Transform, $G_l \in R^{n \times n}$, $gl \in R^n$. During the training process, the output C_l of the transform layer was kept as close as possible to X_A.

The query matrix R_B, key matrix H_B, and value matrix S_B of the source network were

obtained using in Eq. (6) for C_l in the source network. The attention distribution matrix $\hat{X}_B$ of the source network was obtained using Eq. (7) and Eq. (8) was used for the source network attention distribution to obtained the prediction rating $\hat{y}_B$.

User preferences and item feature information are reflected in the SAFMR model based on review text. The SAFMR recommendation model helps users find reviews that are most similar to the reviews written by the user, and the most similar reviews, in turn, allow users to make informed decisions. In the SAFMR model, to predict the preference of $user_U$ for an unknown $item_Q$, reviews of the most similar users are sought. The prediction process is as follows. First, C_l is constructed using the reviews of $user_U$ and $item_Q$ in the source network. Second, all reviews written by other users for $item_Q$ were processed separately to obtain the X_A in the target network. Of all the user reviews processed by the target network, a review written by a particular user can help $user_U$ recommend $item_Q$ if the review written by that user is most similar to the potential representation (C_l) constructed by $user_U$ and $item_Q$.

5. The SAFMR model training

All training samples underwent forward and backpropagation in the neural network. This process is called an epoch. However, the number of epoch training samples at one time may be too large, and it must be divided into multiple small pieces, that is, into multiple batches for training. The number of training samples in each batch is called batch size. Usually, the accuracy of the model converges to a stable value after several rounds, which means that model training is completed. The training of the model can use different types of loss functions, such as the minimum absolute value deviation (L1 norm), least square error (L2 norm), and logic loss [43]. In SAFMR model training, the L1 norm was better than the L2 norm. Therefore, the L1 norm is used in the loss function of the target network and the source network training, and the model loss function is defined as:

$$loss = \sum |y_{UI} - \hat{y}_{UI}| \tag{11}$$

where y_{UI} is the actual rating of $user_U$ on $item_P$, and y_{UI} is the predicted rating. The SAFMR model training was divided into two steps.

5.1 Training of the target network

The loss function for the target network is the L1 norm between actual and predicted ratings.

$$loss_A = | y_{UI} - \hat{y}_A | \tag{12}$$

where $loss_A$ denotes the target network-loss function. The L1 norm between the minimum y_{UI} and the predicted rating $\hat{y}_A$ was constantly updated during the target network training.

5.2 Training of the source network

The loss function is the L2 norm between C_l and reviews the vector X_A output by the CNN layer of the target network. The remaining trainable parameter of the source network is $\hat{y}_S$ and the loss function is the L1 norm between y_{UI} and prediction rating $\hat{y}_B$.

$$loss_{transform} = \| C_l - X_A \|_2 \tag{13}$$

$$loss_B = | y_{UI} - \hat{y}_B | \tag{14}$$

where $loss_{transfrom}$ is the loss function before introducing the attention mechanism layer and $loss_B$ is the loss function of the rest of the source network. The L1 norm between the minimum y_{UI} and predicted rating $\hat{y}_B$ is constantly updated during source network training.

In the SAFMR model, user reviews of the target item have high predictive value. These reviews only take effect during training and are not available for testing. By training the model, its optimal training parameters were determined. When the model is tested in the test set, for a given $user_U$ and unknown $item_Q$, the SAFMR model obtains potential representations of user and item reviews with the help of the source network and then computes predictions based on these potential representations. The most similar reviews of the target network were determined by making the predicted rating $\hat{y}_B$ of the source network infinitely close to the actual rating $\hat{y}_A$ of the target network.

The SAFMR model evaluates the recommendation performance by comparing the predicted results of the source network with the actual results of the target network, and helps $user_U$ make recommendations by using the most similar reviews of the target network. The pseudocode of the SAFMR model training is given in **Algorithm1** and **Algorithm2**.

Algorithm1: The training of the target network

Input: $TextU$, $TextI$, rev_{UI}, y_{UI}

1) $X_A \leftarrow conv(rev_{UI})$
2) $R_A \leftarrow W_r X_A, H_A \leftarrow W_h X_A, S_A \leftarrow W_s X_A$
3) $\hat{X}_A \leftarrow Softmax(\frac{H_A^{\mathrm{T}} R_A}{\sqrt{k}})$
4) $\hat{y}_A \leftarrow \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} S_A \hat{X}_A + w_0$
5) $loss_A \leftarrow | y_{UT} - \hat{y}_A |$
6) new_$loss_1$=backward($loss_A$)

Output: $\hat{y}_A$

The feature-dependency relationship in the review was learned using the SAFMR model. The internal structure of the sentence is captured, so more feature relevance can be obtained and context information can be better considered. In addition, in the long-range dependency problem, the SAFMR model ignores the distance between features and calculates the dependency of the features directly. Therefore, the calculation time was shorter. Finally, when the input text length n is less than the representation dimension d, the self-attention mechanism can calculate the time complexity of each layer [32], and the complexity decreases from $O(kn^2)$ to $O(kn)$.

Algorithm2: The training of the source network

Input: $TextU$, $TextI$, rev_{UI}, y_{UI}

1) $X_B \leftarrow conv(TextU - rev_{UI})$, $X_C \leftarrow conv(TextC - rev_{UI})$
2) $C_0 \leftarrow [X_B X_C]$
3) Transform_input(C_0)
4) For *layer* $l \in$ L do
   $C_l \leftarrow \phi(C_{l-1} G_l + gl)$
   Return C_l

5) $loss_{transform} \leftarrow \| C_l - X_A \|_2$

6) new_$loss_2$=backward($loss_{transform}$)

7) $R_B \leftarrow W_r C_l, H_B \leftarrow W_h C_l, S_B \leftarrow W_s C_l$

8) $\hat{X}_B \leftarrow Softmax(\frac{H_B^{\top} R_B}{\sqrt{k}})$

9) $\hat{y}_B \leftarrow \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} S_B \hat{X}_B + w_0$

10) $loss_B \leftarrow | y_{UI} - \hat{y}_B |$

11) new_$loss3$=backward($loss_B$)

Output: $\hat{y}_B$

6. Experiments

6.1 Datasets

The Amazon dataset was used as experimental data (http://jmcauley.ucsd.edu/data/amazon/). These datasets mainly collected information from the Amazon website from May 1996 to July 2014, including user and item reviews and ratings, a total of 142.8 million reviews, with ratings ranging from 1 to 5. First, the data density was preprocessed using the Skip-Gram [44] model to retain 50,000 words with the highest word frequency in the user and item review data. Second, deactivated words (the, and, is, etc.) and useless punctuation marks were retained, and these words were then subjected to word form reduction and other processes. The datasets statistics and datasets partitions are listed in **Tables 1** and **Table 2**.

Table 1. Datasets statistics

Dataset	Reviews	Users	Items
Digital Music	64705	5541	3368
Beauty	198475	22365	12101
Clothing, Shoes and Jewelry	278653	39387	23033
Home and Kitchen	551466	66519	28237
Kindle Store	982597	68223	61934
Electronics	1688117	192403	63001

Table 2. Datasets partitions

Dataset	Train	Valid	Test
Digital Music	51764	6470	6471
Beauty	158780	19847	19848
Clothing, Shoes and Jewelry	222922	27865	27866
Home and Kitchen	441171	55147	55147
Kindle Store	786077	98260	98260
Electronics	1350493	168812	168812

6.2 Evaluation criteria

The Amazon dataset contains user rating data for items, and the commonly used evaluation indicator in the Recommender Systems of the rating prediction class is the Mean Square Error (MSE). MSE is used as an indicator to measure the prediction results, and it is often used to measure the performance of a recommender system. It is also used in the recommender system competitions held by Baidu, Netflix, and Alibaba. The MSE is used to measure the performance of the model proposed in this paper regarding the accuracy, which is defined as follows.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{15}$$

where N is the number of test data, y_i is the actual rating, $\hat{y}_i$ is the predicted rating. The smaller the MSE value, the higher the recommendation accuracy.

And to measure the relevance of the recommendation results of the model, this study also uses the Normalized Discounted Cumulative Gain (NDCG) as the evaluation index. The NDCG value range is [0, 1], and the larger the NDCG, the higher the recommendation correlation. This is calculated as follows:

$$\mathrm{NDCG}_p = \frac{\sum_{i=1}^{p}\frac{2^{rel_i}-1}{\log_2(i+1)}}{\mathrm{IDCG}} \tag{16}$$

where rel_i represents the relevance score of the recommendation result in position i, IDCG represents the list of the best recommendation results returned by the recommendation system for a certain user, and p represents the length of the recommendation list to be examined.

6.3 Experimental Settings

In each task, we conducted experimental validation on the datasets (Digital Music, Beauty, Clothing, Shoes\Jewelry, Home and Kitchen, Kindle Store, Electronics) and divided the datasets into training, validation, and test sets in a ratio of 8:1:1. The experimental configuration used Ubuntu 18.04 operating system, GPU is RTX 2080 Ti, 11GB video memory, CPU is 7-core Intel(R) Xeon(R) 2.40GHz, and 4GB RAM. The environment required for the experiments is CUDA 11.0, Python 3.8.0, and the main Python Third-Party Libraries are PyTorch 1.7.0, Scipy 1.9.0, NLTK 3.7.0, Numpy 1.21.2 and Pandas 1.4.3.
All input reviews were processed using a natural language processing kit [45] and then degraded. The stop words and punctuation marks in the reviews were individually marked and retained. The review text vectorization of the model is acquired by the Word Embedding Tool [46], which maps lexical information into the semantic space and finally obtains a word vector model. Adaptive Moment Estimation [47] is used to optimize the training optimizer of the model, which is an optimizer based on a random gradient with adaptive features. In the experiment, 15 epochs were trained, each epoch was divided into 500 batches and each batch processed 128 batch sizes. The Mean Square Error was calculated once for the validation set. If a smaller MSE is found, it is saved to the current model. Experiments show that other hyperparameter values have different effects on the training of the recommendation model and subsequently affect the recommendation effect. The names and values of the hyperparameters used in this study are presented in **Table 3**.

Table 3. Hyperparameter value

Number	Hyperparameter	value
1	Epochs	15
2	Batch size	128
3	Learning rate	0.008
4	Learning rate decay	0.99
5	Kernel count	100
6	Kernel size	3
7	Dropout prob	0.5
8	CNN out dim	50
9	Review count	10
10	Review length	80
11	Lowest review count	2
12	Transform layer	2

6.4 Influence of hyperparameters on the SAFMR model

The learning rate is an essential hyperparameter for model training that controls the learning speed and number of errors assigned to the model. The weights of the model were updated at the end of each batch of training instances. In general, a higher learning rate allows the model to learn faster at the expense of a suboptimal final set of weights. A smaller learning rate enables the model to learn a more optimized or globally optimal set of importance but may take longer to train [48].

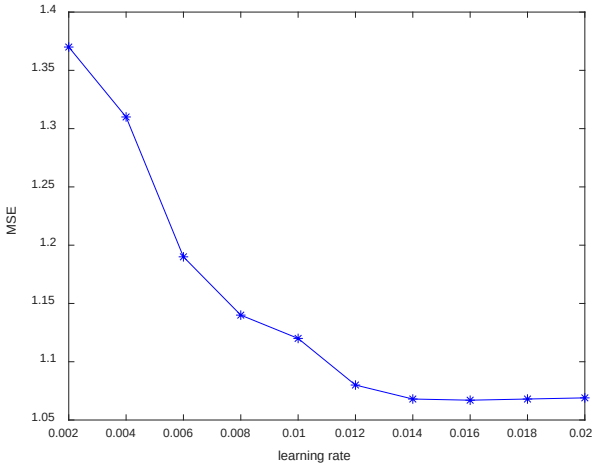

Fig. 2. Variation in MSE with different learning rate

With a perfectly configured learning rate, the model learns the best approximation function for a given available resource in a given number of training periods. On the Home and Kitchen datasets, we plotted the effects of different learning rates on the MSE during the training of the SAFMR model, as shown in **Fig. 2** Several experiments have demonstrated better performance with a learning rate of 0.014. However, to shorten the time for model training and control the number of errors assigned, the SAFMR and control models were trained with the learning rate set at 0.008.

6.5 Performance Comparison

In this section, we compare the recommendation performance of the SAFMR, TransNet [29] and TAFA [33] models on six datasets from Amazon and plot a histogram.

Table 4. Performance of the recommendation models

Dataset	Method	MSE	NDCG	Time(s)
Digital Music (31.3MB)	SAFMR	**1.154**	0.1525	**217**
	TransNet	1.533	0.1015	383
	TAFA	1.486	**0.1723**	559
Beauty (42.7MB)	SAFMR	**1.324**	**0.0762**	**1239**
	TransNet	1.650	0.0586	4747
	TAFA	1.536	0.0662	5987
Clothing, Shoes and Jewelry (46.2MB)	SAFMR	**1.253**	**0.0923**	**1903**
	TransNet	1.669	0.0898	5804
	TAFA	1.330	0.0738	7583
Home and Kitchen (134.9MB)	SAFMR	**1.062**	**0.0902**	**3993**
	TransNet	1.463	0.0716	9297
	TAFA	1.178	0.0762	14084

Kindle Store (265.8MB)	SAFMR	**0.823**	**0.1214**	**23165**
	TransNet	1.164	0.0623	35734
	TAFA	1.074	0.0827	71942
Electronics (484.2MB)	SAFMR	**1.294**	**0.1054**	**13728**
	TransNet	1.765	0.0962	43164
	TAFA	1.496	0.0842	57972

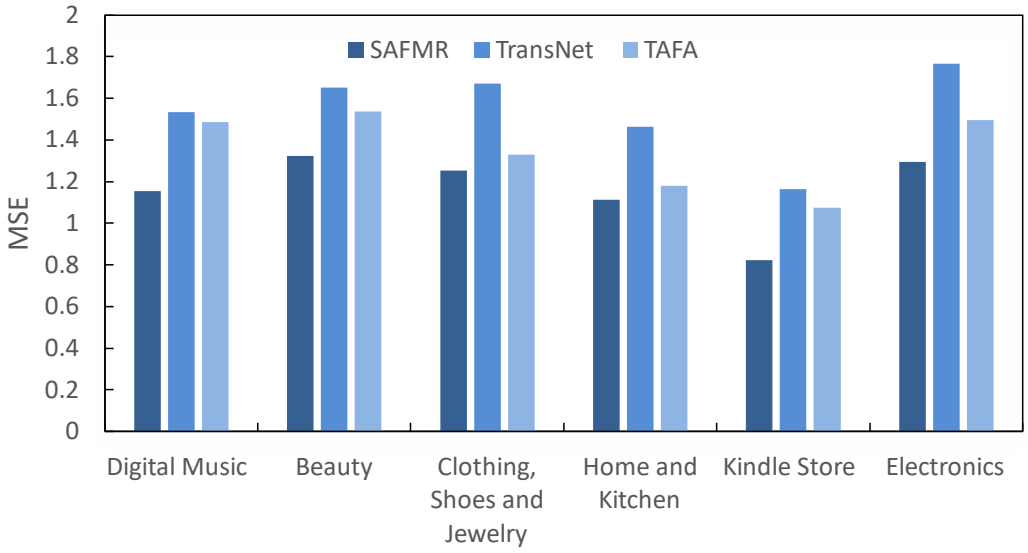

Fig. 3. The MSE performance of the recommendation models

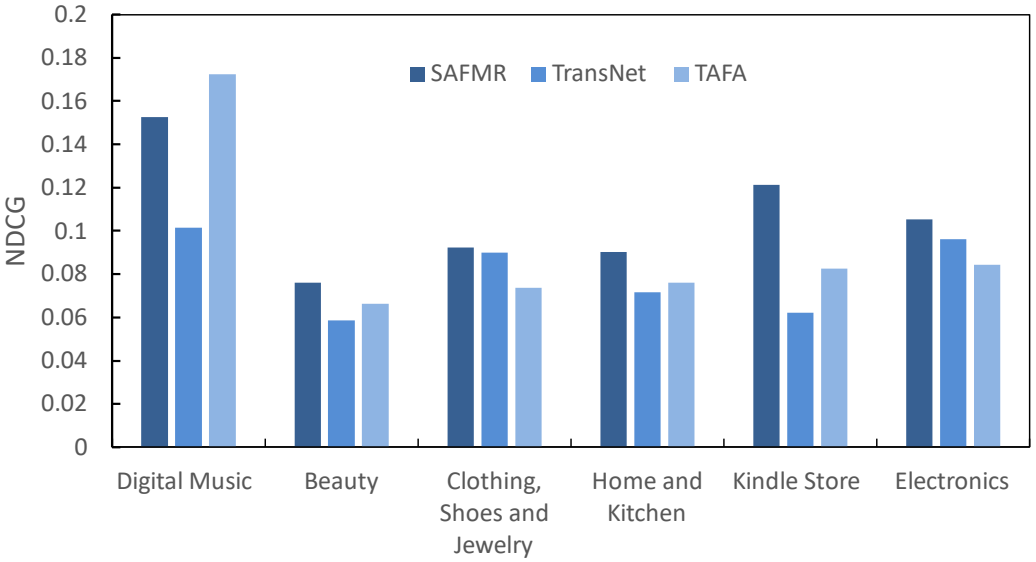

Fig. 4. The NDCG performance of the recommendation model

As shown in **Table 4**, **Fig. 3** and **Fig. 4**, the recommendation performance of this model is better than that of the other models in the six datasets. Under the same training parameters, the MSE and NDCG of the six datasets were improved, respectively, and had the fastest training time. The improved SAFMR model has a smaller error, indicating that the SAFMR model can enhance the internal interpretability of the model. By assigning different weights to features, the data expressed is more accurate, and the performance of the recommendation model is improved

7. Conclusion

The SAFMR model was proposed to express the relevance of user and item features in the reviews. The model automatically learns the intrinsic correlation between features through the self-attention network and allocates more attention resources to essential features. The experimental results show that the SAFMR model achieves better rating prediction results with significantly higher accuracy on the dataset of four Amazon categories, thus verifying the effectiveness of the proposed SAFMR model.

In addition, because of the large amount of information in the review documents, the recommendation model requires a lot of time in the training process and is less efficient. Therefore, GPU clusters can be used to process a large amount of data and shorten the training time using parallel computing.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant 62172351.

References

[1] L. E. Mandilian, P. Diefenbach, and Y. Kim, “Information Overload: A Collaborative Dance Performance,” *IEEE MultiMedia*, vol. 17, pp. 8-13, 2010. Article (CrossRef Link)

[2] P. Resnick and H. R. Varian, “Recommender systems,” *Communications of the ACM*, vol. 40, no. 3, pp. 56-58, 1997. Article (CrossRef Link)

[3] Q. Liu and L. Feng, “Parallel Microblog Information Recommendation Algorithm Based on MapReduce,” *Journal of Chinese Computer Systems*, vol. 38, pp. 1518-1522, 2017. Article (CrossRef Link)

[4] H. Yu, L. Sun, and F. Zhang, “A Robust Bayesian Probabilistic Matrix Factorization Model for Collaborative Filtering Recommender Systems Based on User Anomaly Rating Behavior Detection,” *KSII Transactions on Internet and Information Systems*, vol. 13, no. 9, pp. 4684-4705, 2019. Article (CrossRef Link)

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proc. of the 10th international conference on World Wide Web*, Hong Kong, CHINA, pp. 285-295, 2001. Article (CrossRef Link)

[6] M. Dacrema, F. Parroni, P. Cremonesi, and D. Jannach, “Critically examining the claimed value of convolutions over user-item embedding maps for recommender systems,” in *Proc. of the 29th ACM International Conference on Information & Knowledge Management*, Ireland, pp. 355-363, 2020. Article (CrossRef Link)

[7] J. Pan, J. Xu, A. Ruiz, W. Zhao, S. Pan, S. Yu, and L. Quan, “Field-weighted factorization machines for click-through rate prediction in display advertising,” in *Proc. of the 2018 World Wide Web Conference*, Lyon, FRANCE, pp. 1349-1357, 2018. Article (CrossRef Link)

[8] B. Hao, J. Zhang, H. Yin, C. Li, and H. Chen, "Pre-training graph neural networks for cold-start users and items representation," in *Proc. of the 14th ACM International Conference on Web Search and Data Mining*, Israel, pp. 265-273, 2021. Article (CrossRef Link)
[9] L. Zhong, Y. Wei, H. Yao, W. Deng, Z. Wang, and M. Tong, "Review of deep learning-based personalized learning recommendation," in *Proc. of the 2020 11th International Conference on E-Education, E-Business, E-Management, and E-Learning*, New York, USA, pp. 145-149, 2020. Article (CrossRef Link)
[10] J. Li, C. Wang, X. Fang, K. Yu, J. Zhao, X. Wu, and J. Long, "Multi-label text classification via hierarchical Transformer-CNN," in *Proc. of 2022 14th International Conference on Machine Learning and Computing*, Guangzhou, CHINA, pp. 120-125, 2022. Article (CrossRef Link)
[11] J. Liu, Z. Du, and Y. Liu, "Res-RNN Network and Its Application in Case Text Recognition," in *Proc. of 2019 International Conference on Robotics Systems and Vehicle Technology*, Wuhan, CHINA, pp. 47-50, 2019. Article (CrossRef Link)
[12] J. Kong, J. Sun, M. Jiang, and J. Hou, "A Novel Text Sample Selection Model for Scene Text Detection via Bootstrap Learning," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 2, pp. 771-789, 2019. Article (CrossRef Link)
[13] V. Rho and R. G. Pensa, "Concept-Enhanced Multi-view Co-clustering of Document Data," in *Proc. of 23rd International Symposium on Methodologies for Intelligent Systems*, Warsaw, Poland, pp. 457-467, 2017. Article (CrossRef Link)
[14] B. Diallo, J. Hu, T. Li, G. A. Khan, and A. S. Hussein, "Multi-view document clustering based on geometrical similarity measurement," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 3, pp. 663-675, Mar. 2022. Article (CrossRef Link)
[15] C. Zhang and M. Zhou, "Aspect Level Sentiment Classification Based on Double Attention Mechanism," in *Proc. of the 2019 2nd International Conference on E-Business, Information Management and Computer Science*, Kuala Lumpur, Malaysia, pp. 1-6, 2019. Article (CrossRef Link)
[16] Y. Peng, S. Tian, L. Yu, Y. Lv, and R. Wang, "Malicious url recognition and detection using attention-based cnn-lstm," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 11, pp. 5580-5593, 2019. Article (CrossRef Link)
[17] Y. Chai, W. Yuan, L. Wang, and Z. Liu, "Cross-domain recommendation model based on dual attention mechanism and transfer learning," *Chinese Journal of Computers*, vol. 43, no. 10, pp.1924-1942, 2020. Article (CrossRef Link)
[18] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. of the 26th International Conference on World Wide Web*, Perth, AUSTRALIA, pp. 173-182, 2017. Article (CrossRef Link)
[19] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *Proc. of the 12th ACM Conference on Recommender Systems*, New York, USA, pp. 311-319, 2018. Article (CrossRef Link)
[20] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," in *Proc. of SIGIR'19*, Paris, FRANCE, pp. 165-174, 2019. Article (CrossRef Link)
[21] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, CHINA, pp. 639-648, 2020. Article (CrossRef Link)
[22] J. Xia, D. Li, H. Gu, T. Lu, P. Zhang, and N. Gu, "Incremental Graph Convolutional Network for Collaborative Filtering," in *Proc. of the 30th ACM International Conference on Information and Knowledge Management*, Queensland, Australia, pp. 2170-2179, 2021. Article (CrossRef Link)
[23] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proc. of the 34th International Conference on Machine Learning*, Sydney, NSW, AUSTRALIA, pp. 1126-1135, 2017. Article (CrossRefLink)
[24] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-Aware Message-Passing GCN for Recommendation," in *Proc. of the Web Conference 2021*, Ljubljana, SLOVENIA, pp. 1296-1305, 2021. Article (CrossRef Link)

[25] M. H. Farooq Butt, X. Zhang, G. A. Khan, A. Masood, M. A. Farooq Butt, and O. Khudayberdiev, "A Novel Recommender Model Using Trust Based Networks," in *Proc. of the 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing*, Chengdu, CHINA, pp. 81-84, 2019. Article (CrossRef Link)

[26] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. of the Tenth ACM International Conference on Web Search and Data Mining*, Cambridge, UK, pp. 425-434, 2017. Article (CrossRef Link)

[27] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural Attentional Rating Regression with Review-Level Explanations," in *Proc. of the 2018 World Wide Web Conference*, Lyon, FRANCE, pp. 1583-1592, 2018. Article (CrossRef Link)

[28] L. Wu, C. Quan, C. Li, and D. Ji, "PARL: Let Strangers Speak Out What You Like," in *Proc. of the 27th ACM International Conference on Information and Knowledge Management*, Torino, ITALY, pp. 677-686, 2018. Article (CrossRef Link)

[29] R. Catherine and W. Cohen, "TransNet: Learning to transform for recommendation," in *Proc. of the Eleventh ACM Conference on Recommender Systems*, Como, ITALY, pp. 288-296, 2017. Article (CrossRef Link)

[30] S. Rendle, "Factorization machines," in *Proc. 2010 IEEE International Conference on Data Mining*, Sydney, AUSTRALIA, pp. 995-1000, 2010. Article (CrossRef Link)

[31] L. Wu, C. Quan, C. Li, Q. Wang, B. Zheng, and X. Luo, "A Context-Aware User-Item Representation Learning for Item Recommendation," *ACM Transactions on Information Systems*, vol. 37, no. 2, pp. 1-29, 2019. Article (CrossRef Link)

[32] S. Zhang, Y. Tay, L. Yao, and A. Sun, "Next Item Recommendation with Self-Attention," *aiXiv preprint aiXiv: 1808.06414*, 2018. Article (CrossRef Link)

[33] J. P. Zhou, Z. Cheng, F. Perez, and M. Volkovs, "TAFA: Two-headed Attention Fused Autoencoder for Context-Aware Recommendations," in *Proc. of the 14th ACM Conference on Recommender*, Virtual Event Brazil, pp. 338-347, 2020. Article (CrossRef Link)

[34] X. Shen, C. Yin, and X. Hou, "Self-Attention for Deep Reinforcement Learning," in *Proc. of the 2019 4th International Conference on Mathematics and Artificial Intelligence*, Chengdu, CHINA, pp. 71-75, 2019. Article (CrossRef Link)

[35] A. G. Govindaswamy, E. Montague, D. S. Raicu, and J. Furst, "Cnn as a feature extractor in gaze recognition," in *Proc. of 2020 3rd Artificial Intelligenceand Cloud Computing Conference*, New York, NY, USA, pp. 31-37, 2020. Article (CrossRef Link)

[36] X. Li and H. Ning, "Chinese Text Classification Based on Hybrid Model of CNN and LSTM," in *Proc. of the 3rd International Conference on Data Science and Information Technolog*, Xiamen, CHINA, pp. 129-134, 2020. Article (CrossRef Link)

[37] M. Lim, D. Lee, H. Park, Y. Kang, J. Oh, J. Park, G. Jang, and J. Kim, "Convolutional Neural Network based Audio Event Classification," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 6, pp. 2748-2760, 2018. Article (CrossRef Link)

[38] F. Z. Lahlou, H. Benbrahim, and I. Kassou, "Textual Context Aware Factorization Machines: Improving Recommendation by Leveraging Users' Reviews," in *Proc. of the 2nd International Conference on Smart Digital Environment*, Rabat, Morocco, pp. 64-69, 2018. Article (CrossRef Link)

[39] M. Li, K. Tei, and Y. Fukazawa, "An efficient co-Attention Neural Network for Social Recommendation," in *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, Thessaloniki, GREECE, pp. 34-42, 2019. Article (CrossRef Link)

[40] Z. Guo, D. Yang, B. Liu, L. Xue, and Y. Xiao, "Co-Refining User and Item Representations with Feature-Level Self-Attention for Enhanced Recommendation," in *Proc. of 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Hague, NETHERLANDS, pp. 494-501, 2020. Article (CrossRef Link)

[41] D. Jap, Y.-S. Won, and S. Bhasin, "Fault Injection Attacks on SoftMax Function in Deep Neural Networks," in *Proc. of the 18th ACM International Conference on Computing Frontiers*, Italy, pp. 238-240, 2021. Article (CrossRef Link)

[42] Z. Zheng, S. Huang, R. Weng, X.-Y. Dai, and J. Chen, "Improving Self-Attention Networks with Sequential Relations," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1707-1716, 2020. Article (CrossRef Link)
[43] V. Gupta, D. Choudhary, and P. Tang, "Training Recommender Systems at Scale: Communication-Efficient Model and Data Parallelism," in *Proc. of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Singapore, pp. 2928-2936, 2021. Article (CrossRef Link)
[44] Q. Hu, Y. Pei, and Q. Chen, "SG++: Word Representation with Sentiment and Negation for Twitter Sentiment Classification," in *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pisa, ITALY, pp. 997-1000, 2016. Article (CrossRef Link)
[45] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," in *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, MARYLAND, pp. 55-60, 2014. Article (CrossRefLink)
[46] D. Roy, D. Ganguly, S. Bhatia, S. Bedathur, and M. Mitra, "Using word embeddings for information retrieval: How collection and term normalization choices affect performance. in *Proc. CIKM '18*, New York, NY, USA, pp. 1835-1838, 2018. Article (CrossRef Link)
[47] M. Y. Qiao, X. X. Tang, J. K. Shi, and J. Y. Lan, "Bearing fault diagnosis based on natural adaptive moment estimation algorithm and improved octave convolution," *IEEE Access*, vol. 8, pp. 196790-196803, 2018. Article (CrossRef Link)
[48] Y. Ding, "The Impact of Learning Rate Decay and Periodical Learning Rate Restart on Artificial Neural Network," in *Proc. of 2021 2nd International Conference on Artificial Intelligence in Electronics Engineering*, Phuket, THAILAND, pp. 6-14, 2021. Article (CrossRef Link)

MA HONGSHUANG is currently pursuing the M.S. degree with the School of Computer and Control Engineering, Yantai University. Her research interests include recommended systems and parallel computing.

LIU QICHENG received the Ph.D. degree in engineering from China University of Petroleum, Beijing. He was a Postdoctoral Researcher with the Department of Computer Science, Tsinghua University. He is currently a Professor with the School of Computer and Control Engineering, Yantai University. His research interests include intelligent information processing, big data, and data mining.