symmetry

MDPI

Article

Matrix Factorization Recommendation Algorithm Based on Attention Interaction

Chengzhi Mao [1], Zhifeng Wu [1,*], Yingjie Liu [1] and Zhiwei Shi [2]

[1] School of Information Technology Engineering, Tianjin University of Technology and Education, Hexi District, Tianjin 300222, China; mdoc_dream@163.com (C.M.); yjliu@tute.edu.cn (Y.L.)
[2] School of Computer Science, Shaanxi Normal University, 620 West Chang'an Street, Xi'an 710119, China; zhiweishi@snnu.edu.cn
* Correspondence: zhifeng.wu@tute.edu.cn

Abstract: Recommender systems are widely used in e-commerce, movies, music, social media, and other fields because of their personalized recommendation functions. The recommendation algorithm is used to capture user preferences, item characteristics, and the items that users are interested in are recommended to users. Matrix factorization is widely used in collaborative filtering algorithms because of its simplicity and efficiency. However, the simple dot-product method cannot establish a nonlinear relationship between user latent features and item latent features or make full use of their personalized information. The model of a neural network combined with an attention mechanism can effectively establish a nonlinear relationship between the potential features of users and items and improve the recommendation accuracy of the model. However, it is difficult for the general attention mechanism algorithm to solve the problem of attention interaction when the number of features between the users and items is not the same. To solve the above problems, this paper proposes an attention interaction matrix factorization (AIMF) model. The AIMF model adopts a symmetric structure using MLP calculation. This structure can simultaneously extract the nonlinear features of user latent features and item latent features, thus reducing the computation time of the model. In addition, an improved attention algorithm named slide-attention is included in the model. The algorithm uses the sliding query method to obtain the user's attention to the latent features of the item and solves the interaction problem among different dimensions of the user, and the latent features of the item.

Keywords: artificial intelligence; recommender system; collaborative filtering; slide attention; attention mechanism

check for updates

Citation: Mao, C.; Wu, Z.; Liu, Y.; Shi, Z. Matrix Factorization Recommendation Algorithm Based on Attention Interaction. *Symmetry* **2024**, *16*, 267. https://doi.org/10.3390/sym16030267

Academic Editors: Jian-Qiang Wang, Shiyuan Wang and Guobing Qian

Received: 14 January 2024
Revised: 13 February 2024
Accepted: 19 February 2024
Published: 22 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the development of Internet technology, the amount of network information has increased rapidly, resulting in information overload. The problem of information overload renders users unable to obtain effective information quickly [1–6]. A recommendation system [7] has an excellent effect in addressing this problem. A recommender system is an online personalized recommendation system that helps users find and recommend items of interest by searching, and analyzing user preferences [8,9]. Currently, the main methods for recommender systems to learn user preferences fall into three categories [10,11]: content-based methods [12], collaborative filtering methods [13–21], and hybrid recommendation methods [22]. In essence, these algorithms solve the common problems of data sparsity, data scalability, and cold starts in recommender systems from different perspectives. For instance, Gulzar et al. [23] proposed a new clustering algorithm based on an ordered clustering algorithm (OCA), which aims to reduce the impact of cold starts, and data sparsity. Wu et al. [24] proposed the sampled SoftMax (SSM) loss as an efficient substitute for SoftMax loss, which can optimize long-tail recommendation.

Collaborative filtering is a popular recommendation algorithm. Based on the user behavior data (such as the ratings of users over items, user comments on items, and

so on), it determines the correlation between users and items to provide personalized recommendations for users. The model-based recommendation is the most common collaborative filtering algorithm. Especially in recent years, models integrating an attention mechanism have become a research hotspot in the field of recommendation systems. The basic principle of the attention mechanism imitates the human visual system to assign different weights when processing information at different positions in the input sequence.

More and more researchers have integrated an attention mechanism to different degrees to solve the problems of data sparsity, data scalability, and cold starts.

Wang et al. [25] proposed a multi-attention deep neural network (MADNN) recommendation model based on embedding, and matrix factorization that can effectively alleviate data sparsity and cold start problems. The model enhances the interactivity of user/item embedding through a multi-attention mechanism. However, the dimensions of interactive user/item embeddings must be consistent, which may cause information loss caused by the number of features of different users, and items embedded into the same dimension.

Zhang et al. [26] proposed a probabilistic matrix factorization recommendation for self-attention mechanism convolutional neural networks with item auxiliary information, which can solve the problem of data sparsity in recommendation systems. The model adds a self-attention mechanism to the convolutional layer to establish the interaction between the auxiliary information of different channels. However, there is also the limitation that the dimensions of the auxiliary information of different channels must be consistent.

To verify the existence of the above limitations, we take the features of the movie domain as an example. In the MovieLens100K dataset, the user features include user ID, age, gender, occupation, and zip code, and the movie features include movie ID, movie title, release date, video release date, IMDb URL, unknown, action, and so on. In order to make the user, and item interact, they are usually encoded into a vector with the same dimensions (similar to the two models above), which may lead to a decrease in recommendation accuracy.

To solve the interaction problem, we propose an attention interaction matrix factorization (AIMF) model that integrates the interactions among the users, latent feature relationships of the items, and the users of the original rating matrix.

The main contributions of this paper are as follows:

- We propose a novel collaborative filtering model that leverages the attention mechanism, and matrix factorization for item recommendation. To prevent unreasonable model initialization parameters, which lead to too large or too small model training loss, we used the nonnegative matrix factorization (NMF) technique to improve the information of the two latent features.
- To fully explore users' preferences, we propose the use of the user–item interactive attention mechanism, and the original rating matrix self-attention mechanism to explore the implicit preferences between the users and item potential features and the explicit preferences among the user ratings, respectively, to improve the recommendation accuracy.
- To effectively carry out the user item interaction, and improve the generalization ability of the model, this study proposes a slide-attention algorithm to establish the global attention of the potential features of users and items and to solve the problem of the attention interaction among the feature vectors of different dimensions.
- To verify the effectiveness and feasibility of the model, this study verifies the impact of the latent feature dimension, implicit preference factor, and explicit preference factor on recommendation performance using a large number of experiments on two public datasets in the movie domain. The experimental results show that the model achieved excellent results for the root mean squared error (RMSE) and mean absolute error (MAE).

The remainder of this paper is organized as follows. The related work is presented in Section 2. Then, in Section 3, the proposed model is depicted. The experiment and analysis of the results are discussed in Section 4. The experimental results are discussed in Section 5. Finally, we conclude this paper with future research directions in Section 6.

2. Related Work

Collaborative filtering algorithms based on matrix factorization (MF) are widely used, owing to their simplicity and ease of implementation. However, the rating matrix obtained by MF generally has the features of highly sparse data and uneven distribution, which lead to problems such as low recommendation performance, cold starts, and long tails [27,28]. To solve these problems, many researchers have proposed improved MF algorithms. Koren et al. [14] proved that a latent factor vector can enhance the ability of a model to deal with coefficient features. Badrul et al. [29] proposed singular value decomposition (SVD) to learn a user–item rating information matrix. However, the MF model learned using SVD was prone to overfitting. Subsequently, Funk [30] proposed the FunkSVD model, which adds a regularizer to the conventional SVD method to avoid overfitting the MF model. Koren et al. [14] Proposed a BiasSVD model with a bias term to solve the problem of large fluctuations in user-rating information. Based on BiasSVD, Koren [31,32] proposed the SVD++ model with implicit information to solve the cold-start problem caused by score sparsity. The abovementioned improved matrix factorization algorithm achieved excellent results in solving the sparsity problem of user–item rating information. Nevertheless, the simple vector dot product cannot establish a nonlinear relationship between the latent features of users and items, and the user and item features may not make full use of the latent space, leading to limited recommendation performance.

Deep learning combined with matrix factorization has gradually become a mainstream research topic because of its nonlinear expression ability to establish a model. Li et al., [33] proposed a POI recommendation method fusing auxiliary attribute information based on the neural matrix factorization, integrating the convolutional neural network and attention mechanism (NueMF-CAA) to alleviate the data-sparsity problem. He et al., [34] introduced neural networks based on generalized matrix factorization (GMF). To express the nonlinear relationship between latent features, Tian et al. [35] proposed a deep matrix factorization (DMF) model that combines deep neural networks and matrix factorization techniques. DMF adds multiple hidden layers after the fully connected layers of the neural network to model higher-order interactions between users and items.

Deep neural networks (DNNs) establish nonlinear relationships using weighted summation and reactivation, which may not highlight the key feature information. Therefore, the fusion of deep neural networks and attention mechanisms [36] has become a popular research topic. Wang et al. [37] proposed a convolutional neural network model based on the attention mechanism for CAPTCHA recognition, and their experimental results showed that the accuracy of CAPTCHA recognition was 93.27%. He et al. [38] proposed the inner attention-based recurrent neural network GATE function (IARNN-GATE), which uses the attention mechanism in the gate function of an RNN to control the information transmission of states between the hidden layers. Zhou et al. [39] proposed a recurrent neural network–attention mechanism model (RNN-AM) for microblog sentiment classification. Zhou et al. [40] proposed an image-denoising algorithm based on an attention mechanism and residual block, which effectively solved the problem of real image noise.

3. Attention Interaction Matrix Factorization Model

In this section, we introduce a method for solving the interaction problem of a user and item with different dimensions and the data sparsity problem. We introduce the definition of the problem in Section 3.1. We summarize all notations in this paper as well as functions in Section 3.2. Then, in Section 3.3, the proposed model is depicted in detail. Finally, we propose the loss function for training in Section 3.4.

3.1. Problem Definition

The main tasks of the recommendation algorithm are to explore user preferences and recommend items to users by processing the feature information of the users and items. Suppose that the user set is $U = \{u_1, u_2, \ldots, u_M\}$, the item set is $I = \{i_1, i_2, \ldots, i_N\}$, M is the number of users, and N is the number of items. The raw rating matrix $Raw_Rating \in \mathbb{R}^{M \times N}$ rep-

resents a user's actual rating of an item. Given user $u \in U$ and item $i \in I$, suppose user u has not rated item i. We must construct a user preference for item i and predict r^{ui} (rating of i by user u) based on the rating information of the actual rating matrix $Rating \in \mathbb{R}^{M \times N}$ ($R \in \mathbb{R}^{M \times N}$).

3.2. Related Terminology

We summarize the meaning of each term in Table 1.

Table 1. Relative terms.

Notation	Description
$Rating/R \in \mathbb{R}^{M \times N}$	User–item rating matrix
U	User set
I	Item set
M/N	Number of users/number of items
d	Latent feature dimension
u_o/i_o	User o/item o
$UI - att$	User item latent feature attention interaction output
$R - att$	Inter-users rating attention output
r^{ui}/r'	Predicted rating of item i by user u
$P_u \in \mathbb{R}^{M \times d}$	User latent feature matrix
$Q_i \in \mathbb{R}^{d \times N}$	Item latent feature matrix
$P_{u-mlp} \in \mathbb{R}^{M \times d}$	MLP layer output of user latent features
$Q_{i-mlp} \in \mathbb{R}^{d \times N}$	MLP layer output of item latent features
$Query = (query_1, query_2, \ldots, query_M)$	Query matrix
$Key = (key_1, key_2, \ldots, key_d)$	Key matrix
$Value = (value_1, value_2, \ldots, value_d)$	Value matrix
$\boldsymbol{q}$	Query features
$\boldsymbol{k}$	Key features
$\boldsymbol{v}$	Value feature
$SoftMax(\cdot)$	Normalization function
$\boldsymbol{norm}(\cdot)$	Min–max normalization function
$\boldsymbol{concat}(\cdot)$	Concatenates multiple vectors into a matrix

3.3. AIMF Model

This section describes the AIMF model in detail. In the UI-attention module, NMF, and slide-attention algorithms are used to capture deeper features of users and construct a potential feature correlation between users and items. Simultaneously, personalized recommendations are performed by combining the correlation among user ratings in the self-attention module. This model is illustrated in Figure 1.

The model consists of four parts: input, attention, feature fusion, and output layers. The model establishes an attention interaction between the potential features of the users and items in the UI-attention module to capture the implicit preferences of the users. In the self-attention module, a rating correlation among the users is established to capture their explicit preferences.

Next, the working principle of AIMF is briefly introduced.

Normalized (min–max normalization) real rating data of users are obtained for the input layer. In the UI-attention module, the input layer data are decomposed into two non-negative latent feature matrices, P_u and Q_i, using the NMF technique. Next, the nonlinear features of the two are extracted using a multilayer perceptron (MLP). Then, the output results are input into slide-attention to establish the correlation between the potential feature relationships between the users and items and to obtain the user item attention matrix $UI - att$. Simultaneously, in the self-attention module, the one-hot encoding dot product of the input layer data and the user ID is used to obtain the rating vector of a single user. Next, these rating vectors are used as the query and key-value for attention weight allocation,

and the rating correlation score matrix between each user is obtained. Subsequently, the results are inputted into *add&norm* for re-centralization and rescaling adjustment, and the attention scoring matrix $R-att$ is obtained. After data processing in the attention layer, $UI-att$ and $R-att$ are input to the feature fusion layer for data integration. Finally, the user prediction score for the item is obtained in the output layer.

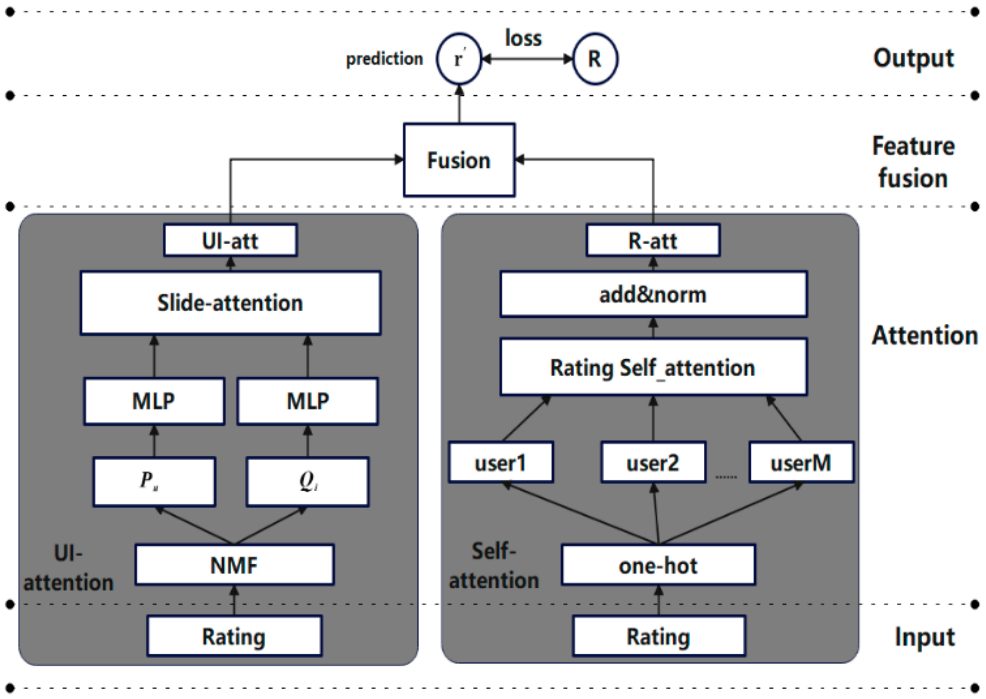


Figure 1. AIMF model.

(1). Input layer

In this study, simple data extraction was carried out on the datasets, and the user's rating matrix data for the item were obtained. To reduce the loss of model training and improve the prediction ability of the model and the generalization ability of the model to the datasets, we normalized the original user item rating matrix.

$$Rating = \boldsymbol{norm}(Raw_Rating) \tag{1}$$

Equation (1) refers to the normalization of the raw rating matrix, where $\boldsymbol{norm}(\cdot)$ represents the min–max normalization function. Raw_Rating is a raw rating matrix (user ratings of items). $Rating$ is a normalized rating matrix (range 0 to 1).

(2). Attention layer

(a). UI-attention module

The main task of this module is to explore the implicit preferences of users. In addition, the UI-attention module uses the slide-attention algorithm to solve the problem of attention interactions where the user feature dimension is inconsistent with the item feature dimension.

The details are as follows:

Step 1: In this step, we perform an NMF operation on $Rating$. Methods based on matrix factorization perform well in sparse matrices. This method also has the advantages of high recommendation accuracy, scalability, and high flexibility [41]. NMF is a variant of the MF technique [42–44]. NMF technology is used to reduce the dimensions of the data, and high-dimensional data are mapped to the low-dimensional space while retaining the main information. This operation can reduce the complexity of the data and improve the training efficiency of the machine learning model.

The specific treatment is shown in Equations (2) and (3):

$$E_{err} = \boldsymbol{min} \parallel Rating - P_u \cdot Q_i \parallel_2 \tag{2}$$

$$Rating \approx P_u \cdot Q_i \tag{3}$$

where E_{err} denotes computational loss. $Rating \in \mathbb{R}^{M \times N}$ represents the rating matrix. $P_u \in \mathbb{R}^{M \times d}$ ($P_u > 0$) represents the latent feature matrix of the user. $Q_i \in \mathbb{R}^{d \times N}$ ($Q_i > 0$) represents the latent feature matrix of an item. $\| \cdot \|_2$ denotes the 2-Norm. d is the dimension of the latent features of users/items. P_u and Q_i are updated by gradient descent (GD) [45], and when a certain number of iterations is reached, the dot product of the two will approximate $Rating$. In this study, P_u and Q_i were obtained by 100 iterations.

Step 2: In this step, the nonlinear features of P_u and Q_i are extracted using MLP. To have a certain generalization ability of the model, here, we use MLP to raise and then reduce the dimensions of P_u and Q_i, respectively, and obtain their implicit nonlinear features. The P_u and Q_i obtained in **Step 1** are respectively input into the MLP for calculation to obtain $P_{u-mlp} \in \mathbb{R}^{M \times d}$ and $Q_{i-mlp} \in \mathbb{R}^{d \times N}$. Figure 2 shows the MLP design used in this study.

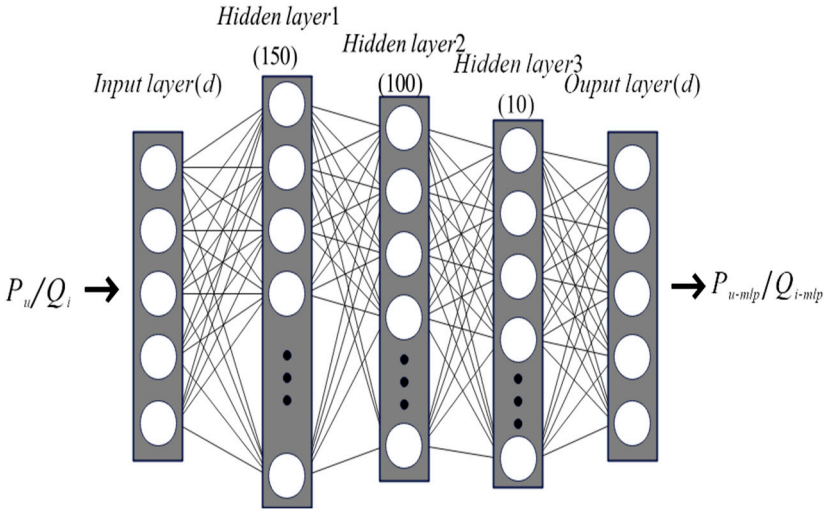


Figure 2. Multilayer perceptron.

The calculation is as follows:

I. Hidden layer-1

$$out_1 = \mathrm{F}(x \cdot W_1 + b_1) \tag{4}$$

where out_1 denotes the output of the first hidden layer. x is the input of the MLP, and the inputs in this study were P_u and Q_i, respectively. W_1 and b_1 are the weight matrix and the bias of the first hidden layer, respectively. F($\cdot$) is an activation function (in this study, ReLU was chosen as the activation function in the MLP).

II. Hidden layer-L

$$out_L = \mathrm{F}(out_{L-1} \cdot W_L + b_L) \tag{5}$$

where out_L is the output of the first hidden layer. W_L and b_L are the weight matrix and bias of the Lth hidden layer, respectively.

III. Output layer

$$final_out = F(out_L \cdot W_o + b_o) \tag{6}$$

where $final_out$ is the final output. W_L and b_L are the weight matrix and the bias of the output layer, respectively.

Therefore, P_{u-mlp} and Q_{i-mlp} can be calculated as follows:

$$P_{u-mlp} = MLP(P_u) \tag{7}$$

$$Q_{i-mlp} = MLP(Q_i) \tag{8}$$

where $MLP(\cdot)$ denotes the I–III computational processes. P_{u-mlp} and Q_{i-mlp} denote the outputs of the MLP layer.

Step 3: In this step, we apply the attention interaction between P_{u-mlp} and Q_{i-mlp} obtained from **Step 2** to obtain $UI - att \in \mathbb{R}^{M \times N}$. However, the user latent feature matrix represented by P_{u-mlp} and the item latent feature matrix represented by Q_{i-mlp} cannot directly establish the attention relationship by conventional attention mechanism. The reason is that the latent feature dimension of each user representing the item is d, while the number of items is N.

In this study, we propose a slide-attention algorithm to establish the relevance among different dimensions of user and item latent features. Based on the attention mechanism, the algorithm adjusts the score calculation to adapt to different numbers of query and key-value attention interactions with different feature dimensions, thereby making attention technology more widely used. In the AIMF model, P_{u-mlp} is used as $Query$, and Q_{i-mlp} as Key, $Value$.

Next, the calculation process of the method is introduced. In **Step 3-1**, we show the general computational process of the attention mechanism. The main difference between slide-attention and the attention mechanism is shown in **Step 3-2**.

Step 3-1. Attention mechanism

The attention mechanism interacts with the query and key through attention convergence to generate a score for the value [46]. This works for the mechanism in Figure 3.

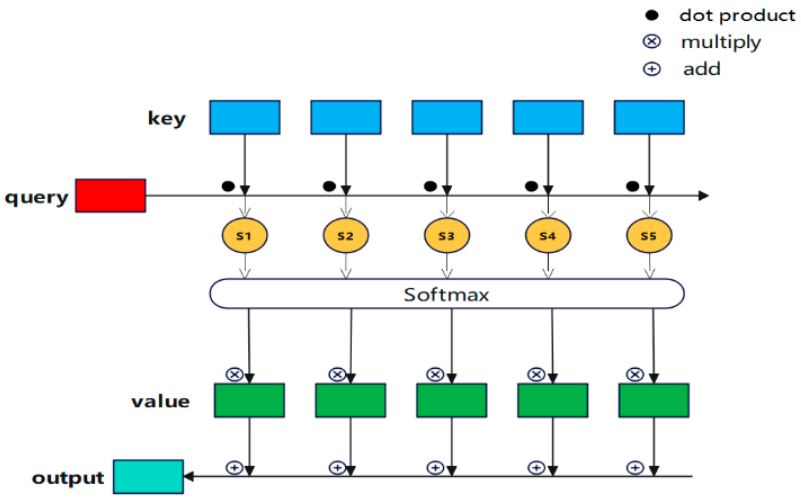


Figure 3. How attention works.

The similarity between the query and the key is computed as follows:

$$s_i = \boldsymbol{q} \cdot \boldsymbol{k}_i \tag{9}$$

where s_i represents the $\boldsymbol{v}_i$ score. $\boldsymbol{q}$ can represent any vector in the set $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m\}$ (where the value of m is problem-specific). $\boldsymbol{k}_i$ is a key feature of $\boldsymbol{v}_i$ ($\boldsymbol{v}_i$ is the ith value feature). Equation (9) reflects the process of determining the $\boldsymbol{v}_i$ score, and Figure 3 shows the single $\boldsymbol{q}$ feature calculation score process. The full attention process computes each $\boldsymbol{q}$ feature that interacts with the $\boldsymbol{k} - \boldsymbol{v}$ feature.

The SoftMax function is used to numerically convert the attention scores:

$$\alpha_i = SoftMax(s_i) = \frac{exp(s_i)}{\sum_j exp(s_j)} \tag{10}$$

where α_i is the score of the v_i features obtained after a single $\boldsymbol{q}$ feature query and normalization using the SoftMax function.

The weighted sum of the values based on the score is calculated as follows:

$$Attention((\boldsymbol{k}, \boldsymbol{v}), \boldsymbol{q}) = \sum_i \alpha_i \boldsymbol{v}_i = \sum_i \frac{exp(s(\boldsymbol{k}_i, \boldsymbol{q}))}{\sum_j exp(s(\boldsymbol{k}_j, \boldsymbol{q}))} \boldsymbol{v}_i \tag{11}$$

Equation (11) calculates the attention of the $\boldsymbol{q}$ feature to the $\boldsymbol{k} - \boldsymbol{v}$ feature, $Attention((\boldsymbol{k}, \boldsymbol{v}), \boldsymbol{q}) \in \mathbb{R}^{1 \times n}$ (the n value depends on the specific problem). $\boldsymbol{k}$ and $\boldsymbol{v}$ denote the key and value features, respectively.

In summary, the output of the attention mechanism is calculated as follows:

$$\boldsymbol{att} = \boldsymbol{concat}(Attention_1, Attention_2, \ldots, Attention_n) \tag{12}$$

where ***concat*** represents the concatenation operation, which concatenates $attention_1, attention_2, \ldots, attention_n$ into a matrix $\boldsymbol{att} \in \mathbb{R}^{m \times n}$.

Step 3-2. Slide-attention

Compared with conventional attention, slide-attention achieves the purpose of the attention interaction of different dimensions by adjusting the interaction scores of $\boldsymbol{q}$ and $\boldsymbol{k}_i$. Given that $\boldsymbol{p}_1 \in \mathbb{R}^{1 \times d}$ and $\boldsymbol{q}_1 \in \mathbb{R}^{1 \times N}$, which represent the latent features rated by user 1 for all items and the latent features rated by all items themselves, our main task is to establish the correlation between them, that is, to query all features in $\boldsymbol{q}_1$ using $\boldsymbol{p}_1$ and generate an interaction. The calculation principle is as follows.

The $\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$ values are calculated as follows:

$$\boldsymbol{q}_i = \boldsymbol{W}^q \cdot query_i \tag{13}$$

$$\boldsymbol{k}_j = \boldsymbol{W}^k \cdot key_j \tag{14}$$

$$\boldsymbol{v}_j = \boldsymbol{W}^v \cdot value_j \tag{15}$$

Equations (13)–(15) comprise the calculation processes of the query feature, key feature, and value feature, respectively. $\boldsymbol{q}_i$ represents the ith query feature. $\boldsymbol{W}^q \in \mathbb{R}^{d \times d}$ represents the trainable weight parameter matrix of the query. $query_i$ denotes the ith query vector. $\boldsymbol{k}_j$ is the jth key feature. $\boldsymbol{W}^k \in \mathbb{R}^{d \times d}$ represents the trainable weight parameter matrix of the key. Let key_j denote the jth key vector. $\boldsymbol{v}_j$ is the jth value feature. $\boldsymbol{W}^v \in \mathbb{R}^{d \times d}$ represents the trainable weight parameter matrix of the value. $value_j$ is the jth value vector. Here, $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, d$.

Calculating the slide-attention score:

Figure 4 shows the score calculation process of query feature $\boldsymbol{q}$ and key feature $\boldsymbol{k}_i$ in slide-attention. It works by taking the dot product of the query feature $\boldsymbol{q}$ (with d feature elements) and the corresponding elements of $\boldsymbol{k}_i$ and then averaging all the scores to obtain the final score of $\boldsymbol{q}$ against $\boldsymbol{k}_i$. Here, $\boldsymbol{q}$ is any vector in the set $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M\}$ and $i = 1, 2, \ldots, d$, M is the number of users.

The calculation process of the score of $\boldsymbol{q}$ against $\boldsymbol{k}_i$ is as follows:

$\exists \boldsymbol{q}, \boldsymbol{k}_i$, $\boldsymbol{q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_d)$, $\boldsymbol{k}_i = (k_{i,1}, \ldots, k_{i,d}, \ldots, k_{i,N})$, and d are the feature dimensions. The dot product of $\boldsymbol{q}$ and $\boldsymbol{k}_i^1 = (k_{1,1}, \ldots, k_{1,d})$ yields the $score1$ value. To ensure that $\boldsymbol{q}$ is sufficient for $\boldsymbol{k}_i$, in this paper, $\boldsymbol{k}_i^1$ is merged into the tail of $\boldsymbol{k}_i$ to obtain $\boldsymbol{k}_i^+$ ($\boldsymbol{k}_i^+ = (\boldsymbol{k}_{i,1}, \boldsymbol{k}_{i,2}, \ldots, \boldsymbol{k}_{i,d}, \ldots, \boldsymbol{k}_{i,N}, \ldots, \boldsymbol{k}_{i,N+d})$). Thus, after $\boldsymbol{q}$ sufficiently interacts with $\boldsymbol{k}_i^+$, the score vector $\boldsymbol{score}_i$ is obtained ($\boldsymbol{score}_i = (score1, score2, \ldots, scoreN)$). Finally, the score of $\boldsymbol{q}$ over $\boldsymbol{k}_i$ was obtained by averaging the sum of all the elements in $\boldsymbol{score}_i$ and yielding $\boldsymbol{Score}_i$.

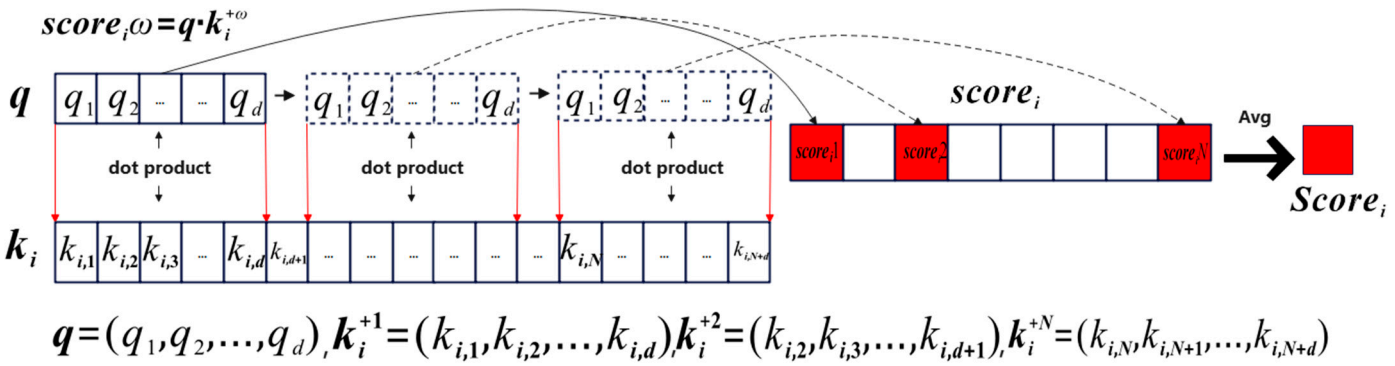


Figure 4. Slide-attention.

The relevant calculation formula is as follows:

$$\boldsymbol{score}_i\boldsymbol{\omega} = \boldsymbol{q}\cdot\boldsymbol{k}_i^{+\omega} \tag{16}$$

$$\boldsymbol{Score}_i\text{=}\boldsymbol{Avg}(\boldsymbol{score}_i) \tag{17}$$

Equation (16) represents the multiplication of $\boldsymbol{q}$ by the corresponding elements of the $\boldsymbol{\omega} \sim \boldsymbol{\omega} + d$-elements of $\boldsymbol{k}_i^+$ ($\boldsymbol{\omega} = 1, 2, \ldots, N$). Here, $\boldsymbol{score}_i\boldsymbol{\omega}$ denotes the score obtained by the $\boldsymbol{\omega}$th dot product. Equation (17) represents the calculation of the v_i score. $\boldsymbol{Score}_i$ represents the final interaction score between $\boldsymbol{q}$ and $\boldsymbol{k}_i$, that is, all $\boldsymbol{score}_i\boldsymbol{\omega}$ values calculated by $\boldsymbol{q}$ and $\boldsymbol{k}_i^+$ are averaged. Here, $\boldsymbol{q}$ represents any member in the $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M\}$ set. $\boldsymbol{Avg}(\cdot)$ is the mean function.

Combining Equations (11) and (17), $slide - attention_l$ is obtained as follows:

$$slide - attention_l((\boldsymbol{k}, \boldsymbol{v}), \boldsymbol{q}_l) = \sum_{i=1}^{N} \alpha_i \boldsymbol{v}_i = \sum_{i=1}^{N} \frac{exp(\boldsymbol{Score}_i(\boldsymbol{k}_i, \boldsymbol{q}_l))}{\sum_j exp(\boldsymbol{Score}_i(\boldsymbol{k}_j, \boldsymbol{q}_l))} \boldsymbol{v}_i \tag{18}$$

Substituting Equation (18) into Equation (12), the UI-attention module output $UI - att$ is obtained.

$$UI - att = \boldsymbol{concat}(slide - attention_1, \ldots, slide - attention_N) \tag{19}$$

where $UI - att \in \mathbb{R}^{M \times N}$ and $slide - attention_l \in \mathbb{R}^{1 \times N}$. $UI - att$ is the attention score of the user latent features to the item latent features obtained by P_{u-mlp} and Q_{i-mlp} after the slide-attention operation, which is the implicit preference of the user.

The process of the slide-attention Algorithm 1 is as follows:

Algorithm 1: Slide-attention process

Require: Deep user latent features $P_{u-mlp} \in \mathbb{R}^{M \times d}$, deep item latent features $Q_{i-mlp} \in \mathbb{R}^{d \times N}$.
Ensure: $UI - att \in \mathbb{R}^{M \times N}$.
Input: $P_{u-mlp} \in \mathbb{R}^{M \times d}$, $Q_{i-mlp} \in \mathbb{R}^{d \times N}$.
Step1: Determining query features $\boldsymbol{Q} \in \mathbb{R}^{M \times d}$, key features $\boldsymbol{K} \in \mathbb{R}^{d \times N}$, value features $\boldsymbol{V} \in \mathbb{R}^{d \times N}$. We take the dot product of $\boldsymbol{P}_{u-mlp}$ with the query weight matrix $\boldsymbol{W}^q \in \mathbb{R}^{d \times d}$ to obtain the query features ($\boldsymbol{Q}$). We take the dot product of Q_{i-mlp} with the key weight matrix $\boldsymbol{W}^k \in \mathbb{R}^{d \times d}$ and the value weight matrix $\boldsymbol{W}^v \in \mathbb{R}^{d \times d}$, respectively, to obtain the key and value features ($\boldsymbol{K}$ and $\boldsymbol{V}$)

$$\boldsymbol{Q} \leftarrow \boldsymbol{W}^q \cdot P_{u-mlp}\ ,\ \boldsymbol{K} \leftarrow \boldsymbol{W}^k \cdot Q_{i-mlp}\ ,\ \boldsymbol{V} \leftarrow \boldsymbol{W}^v \cdot Q_{i-mlp}$$

Step2: We extend $\boldsymbol{K} \in \mathbb{R}^{d \times N}$ to $\boldsymbol{K} \in \mathbb{R}^{d \times (N+d)}$ (add the first d column elements of $\boldsymbol{K}$ to the end of $\boldsymbol{K}$).
Step3: Expand the dimensions of $\boldsymbol{Q} \in \mathbb{R}^{M \times d}$, $\boldsymbol{K} \in \mathbb{R}^{d \times N}$, and $\boldsymbol{V} \in \mathbb{R}^{d \times N}$ to $\boldsymbol{Q} \in \mathbb{R}^{M \times 1 \times d}$, $\boldsymbol{K} \in \mathbb{R}^{M \times d \times N}$ and $\boldsymbol{V} \in \mathbb{R}^{M \times d \times N}$, respectively. The purpose is to let $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ perform batch dot multiplication to improve computational efficiency.

Algorithm 1: *Cont.*

Step4: Calculate the score between the latent features of the user and item.

FOR *Count* **in** N

$$score \leftarrow \textbf{Batch_matrix_multiplication}(\boldsymbol{Q}, \boldsymbol{k}_{Count})$$
$$\boldsymbol{score} \leftarrow \boldsymbol{concat}(\boldsymbol{score}, score)$$

END FOR

Here, the count goes from 1 to N. $\boldsymbol{k}_{Count} \in \mathbb{R}^{M\times d\times d}$ is the matrix in $\boldsymbol{K}$ consisting of columns *Count* to *Count* + d. $score \in \mathbb{R}^{M\times 1\times d}$ represents the attention score computed by $\boldsymbol{Q}$ on the sub-features of $\boldsymbol{K}$. $\boldsymbol{score} \in \mathbb{R}^{M\times N\times d}$ represents the query feature $\boldsymbol{Q}$ operation on the key feature $\boldsymbol{K}$ to obtain the total score matrix. *concat*(A, *B*) means to attach B to A in the second dimension. N denotes the number of keys (the number of columns of $\boldsymbol{K}$).

Step5: The masking *SoftMax* operation is performed on ***score***.

$$\boldsymbol{score} \leftarrow SoftMax(\boldsymbol{score}, invlide - data)$$

where *invlide-data* is a two-dimensional matrix used to mask some elements of ***score***.

Step6: Averaging $\boldsymbol{score} \in \mathbb{R}^{M\times N\times d}$ yields $\boldsymbol{score} \in \mathbb{R}^{M\times d\times 1}$. $\boldsymbol{score} \in \mathbb{R}^{M\times d\times 1}$ is the final score of the queries for the keys.

Step7: The resulting $\boldsymbol{score} \in \mathbb{R}^{M\times d\times 1}$ is a batch-inner product with the value feature $\boldsymbol{V} \in \mathbb{R}^{M\times d\times 1}$, and the dimension is reduced to obtain $UI - att \in \mathbb{R}^{M\times N}$

$$UI - att \leftarrow \textbf{Batch_matrix_multiplication}(\boldsymbol{score}, \boldsymbol{V})$$
$$UI - att \leftarrow \textbf{dimensionality_reduction}(UI - att)$$

where $\textbf{dimensionality_reduction}(UI - att)$ is the compressed dimension function. Here, it means that the dimension reduction operation is performed on $UI - att$ in order to reduce unnecessary space loss.

Step8: return $UI - att$

Output: $UI - att$

(b). Self-attention module

Compared with the UI-attention module capturing the implicit preferences of the user, the main task of this module is to establish a rating correlation among users in order to explore their explicit preferences.

The details are as follows:

Step 1: All the user IDs were encoded using one-hot encoding. The main idea is to map each class into a vector, where only one element is one and the rest are zero. The one indicates the class in which the data point is located. For example, to encode the user ID, the code of user 1 is denoted as 100,000..., and the code of user 2 is denoted as 010000..., where the length of the code is the number of users M [39].

Step 2: The one-hot encoding of each user from **Step 1** is a dot product with the rating matrix. The purpose of this part of the one-hot operation is to obtain the rating of each user for the item. The calculation is given by Equation (20):

$$\boldsymbol{user}_l = One - hot_l \cdot Rating \tag{20}$$

where $\boldsymbol{user}_l \in \mathbb{R}^{1\times N}$ is the rating of the lth user ($l = 1, 2, \ldots, M$). $One - hot_l \in \mathbb{R}^{1\times N}$ denotes the one-hot encoding of user l.

Step 3: We assigned the attention score from each user's rating of the item obtained in **Step 2**, in order to obtain the similarity relationship among users about the rating of the item. Compared with the attention mechanism, the self-attention mechanism pays more attention to finding connections among the inputs themselves (internal) and is commonly used to calculate [47] the scaled dot-product attention [38].

$$Self - attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = SoftMax(\frac{\boldsymbol{Q}\cdot\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{21}$$

Equation (21) provides a brief formula to calculate the self-attention mechanism. $\boldsymbol{Q}$ represents the set of query features. $\boldsymbol{K}$ represents a set of key features. $\boldsymbol{V}$ represents a set of value features. Different from traditional attention, here, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are obtained by the dot product of the same inputs with $\boldsymbol{W}^q$, $\boldsymbol{W}^k$, and $\boldsymbol{W}^v$, respectively ($\boldsymbol{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M\}$, $\boldsymbol{K} = \{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_N\}$, and $\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N\}$).

Thus, similar to Equations (13)–(15), the $\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$ of $Self-attention$ are calculated as follows:

$$\boldsymbol{q}_l = \boldsymbol{W}^q \cdot \boldsymbol{user}_l \tag{22}$$

$$\boldsymbol{k}_l = \boldsymbol{W}^k \cdot \boldsymbol{user}_l \tag{23}$$

$$\boldsymbol{v}_l = \boldsymbol{W}^v \cdot \boldsymbol{user}_l \tag{24}$$

where $\boldsymbol{q}_l \in \mathbb{R}^{1\times N}$, $\boldsymbol{k}_l \in \mathbb{R}^{1\times N}$, and $\boldsymbol{v}_l \in \mathbb{R}^{1\times N}$ denote the query, key, and value features of user l, respectively.

Next, the rating relevance among users is obtained by substituting Equations (22)–(24) into Equations (11) and (12), respectively.

$$\textbf{Self-attention} = \boldsymbol{concat}(Self-attention_1, \ldots, Self-attention_N) \tag{25}$$

In this study, $Rating \in \mathbb{R}^{M\times N}$ of the self-attention mechanism was input. To facilitate understanding and distinguish the difference between the attention mechanism and the self-attention mechanism, one-hot coding was used to separate the scores of each user for all items in the rating matrix $Rating$, and an attention calculation was performed.

Step 4: We performed a residual operation on the similarity matrix obtained among the users' ratings of items in **Step 3** (**Self-attention**) and the rating matrix ($Rating$). To preserve the original rating information and improve the structure of the model stability and training efficiency, we normalized the sum of **Self-attention** and $Rating$. The residual network can maintain the performance of the network model with an increase in the number of network layers and then slow down the degradation of the model performance [48].

$$R-att = \boldsymbol{norm}(\textbf{Self-attention} + Rating) \tag{26}$$

Equation (26) represents the attention calculation method of the rating matrix, that is, the user display preference is explored through the rating correlation among users. $\boldsymbol{norm}(\cdot)$ represents the min–max normalization operation. **Self-attention** is the similarity matrix obtained among the users' ratings of items. $Rating$ represents the rating matrix.

(3). Feature fusion layer

This layer shows the degree of influence of implicit and explicit user preferences ($UI-att$ and $R-att$) on the accuracy of personalized user recommendations. For a recommendation system, when the user's preference information is not detailed, mining the implicit information can reduce the difficulty of user decision making and improve the accuracy of recommendation. This study experimentally proved the influence of implicit and explicit preferences on recommendation accuracy.

The formula is as follows:

$$F-att = UI-att + R-att \tag{27}$$

where $F-att \in \mathbb{R}^{M\times N}$ represents the output result of the feature fusion layer. $UI-att \in \mathbb{R}^{M\times N}$ represents the user–item implicit attention. $R-att \in \mathbb{R}^{M\times N}$ denotes the explicit attention among users.

(4). Output layer

The output of this layer is the model's prediction of the user ratings of an item. In this layer, we activate the output result of the feature fusion layer to achieve a result value within a certain range. The parametric rectified linear unit (PReLU) function is used as the activation function of the final feature fusion layer output in this study. Compared with ReLU, the PReLU function improves model fitting by adding a parameter to adjust the negative value (non-0) in the face of negative inputs, with almost no additional computational cost and little risk of overfitting [49]. The formula is as follows:

$$f(y_i) = \begin{cases} y_i, & if\ y_i > 0 \\ a_i y_i, & if\ y_i \le 0 \end{cases} \tag{28}$$

where y_i is the input channel of the ith nonlinear activation function $f(\cdot)$. a_i is a coefficient that controls the slope of the curve (a_i is learnable). The subscript i in a_i indicates that nonlinear activation is allowed to vary across channels. $a_i = 0$ is a ReLU function, in which a_i was initially set to 0.25.

Therefore, AIMF's prediction r' is calculated as follows:

$$r' = \mathrm{PReLU}(F - att) \tag{29}$$

3.4. Loss Function

To optimize the parameter values of the AIMF model better, in this study, we used the latent features of users and items as the mean score difference loss of the L2 regularizer, which is calculated as follows:

$$L = \varphi * \sum (R - r')^2 + \mu * \|P_u\|_2 + \theta * \|Q_i\|_2 \tag{30}$$

where R and r' represent the true and predicted scores of the training set, respectively. P_u and Q_i represent the latent user and item features, respectively, where φ is the error factor, φ=1/num_R, and num_R is the true number of ratings. μ represents the user latent feature factor, that is, μ = 1/num_u, where num_u is the number of users. θ represents the latent feature factor of the item, that is, θ=1/num_i, where num_i represents the number of items. $\| \cdot \|_2$ denotes the 2-Norm.

4. Experiment and Result Analysis

The main task presented in this section is the training of the model and predicting the results on two public datasets in the movie domain: MovieLens100K and MovieLens1M [50]. In this study, RMSE and MAE were used to verify the feasibility and effectiveness of the model in comparison with other methods. This study focused on the following three aspects:

(1) The recommendation performance of the attention interaction model.
(2) The effect of the number of neurons on model performance.
(3) The effect of the number of hidden layers on model performance.

This section is divided into three subsections. In Section 4.1, we present the setting for the experiment, including the dataset, evaluation criterion, baseline, and parameter settings. Then, the comparison and analysis of the experimental results are presented in Section 4.2. Finally, in Section 4.3, we present an ablation study.

4.1. Experiment Setting

4.1.1. Dataset

This study verifies the feasibility and effectiveness of the AIMF model on the public datasets MovieLens100K and MovieLens1M [50]. MovieLens is a movie dataset widely used in recommendation systems. It is mainly composed of movie rating data, user demographic data, and movie attribute data and contains multiple versions of different data sizes. See Table 2 for further details.

- The MovieLens100K dataset contains features such as the ID, age, gender, and occupation of 943 users, as well as 100,000 ratings for 1682 items. The ratings range from one to five. Each user rated at least twenty movies. The data were collected through the studio website (movielens.umn.edu) over a seven-month period from 19 September 1997 to 22 April 1998.
- The MovieLens1M dataset contains 10,000,209 ratings from 6040 users for 3952 items. The ratings range from one to five. Each user rated at least twenty movies. One million sampled users joined the system in the year 2000.

We normalized the rating matrix in both datasets to reduce the loss of model training, improve the prediction ability of the model, and improve the generalization ability of the model to the dataset. The rating data have a score of [0, 1], where a score of 0 indicates that the user has not rated the item.

Table 2. Description of the MovieLens data.

Feature	MovieLens	
	100 K	**1 M**
Users	943	6040
Items	1682	3952
Ratings	100,000	1,000,209
Range of ratings	1~5	1~5
Rating sparsity	93.70%	95.81%

4.1.2. Evaluation Criterion

In this study, two commonly used recommender system indicators were used to verify the effectiveness and feasibility of the AIMF model: MAE and RMSE [51]. Equations (31) and (32) define MAE and RMSE, respectively.

$$MAE = \frac{1}{T}\sum_{i=1}^{T}\left|R - r'\right| \tag{31}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}|R - r'|^2} \tag{32}$$

where R and r' represent the true and predicted ratings of the test set, respectively. T represents the number of true ratings in the test set.

4.1.3. Comparison of Methods and Parameter Settings

We compared and analyzed the performance of the AIMF model using the following models to verify its feasibility:

- SVD++ [33,34]. SVD++ is an improved SVD algorithm proposed by Koren. It adds user bias, item bias, and overall bias to the SVD algorithm and uses these bias scores to replace the actual score to evaluate the model and then fit the user preferences, item features, and global features.
- NMF [52]. The basic idea is that, given a nonnegative matrix X, the NMF algorithm can find a nonnegative matrix W and a nonnegative matrix H, such that the dot product of matrix W and H is equal to X so that a nonnegative matrix can be decomposed into two nonnegative matrices.
- DeepHAMF [53]. The deep hierarchical attention matrix factorization model can fully explore the user's preference information by establishing the interactions among the original input ID information (users and items), the self-attention layer information, and the hierarchical attention information. This paper shows that the self-attention layer is used to capture the explicit preference information of the target and that hierarchical attention is used to capture the potential information of the target.
- DNNMF [54]. The deep nonlinear nonnegative matrix factorization model limits the embedding layer data to be greater than 0 by using nonnegative matrix factorization technology and then uses a deep neural network to establish a nonlinear relationship between the potential features of the users and items to solve the problem of data sparsity.

All the models in this paper were implemented using pytorch (vision 1.13.1). The training data were 80% randomly drawn from the user–item rating matrix, and the test data were the remaining 20%. We used the Adam [55] optimizer, with a learning rate $lr = 0.5$. MLP used three hidden layers with 150, 100, and 10 neurons in each layer; the hidden layer used ReLU as the activation function, and the output layer used PReLU as the activation function. For the MLP network parameter settings, we used a normal distribution with a mean of 0 and a variance of 0.01 for initialization. For the slide-attention and self-attention structures, we randomly initialized the model parameters and set the number of attention heads in self-attention to 2. For the output layer activation function PReLU,

we used $a_i = 0.25$ as the initialization parameter. We manually adjusted all latent feature dimensions $d = 5$ (including the NMF latent feature dimension, $\boldsymbol{W}^q$, $\boldsymbol{W}^k$, and $\boldsymbol{W}^v$ in slide-attention and self-attention). The number of NMF iterations was 100.

4.2. Comparison and Analysis of Experimental Results

The main task presented in this section is the analysis of the AIMF model and its comparison with the selected methods to verify its effectiveness and feasibility.

4.2.1. Comparison of the MAE and RMSE Performance Metrics of the Models

To answer question (1), we compared the performance of the SVD++ algorithm, NMF algorithm, DeepHAMF, and DNNMF algorithms. The results are listed in Table 3.

Table 3. Comparison of the MAE and RMSE results.

Method	MovieLens100K		MovieLens1M	
	MAE	**RMSE**	**MAE**	**RMSE**
SVD++	0.726	0.9224	0.6629	0.8510
NMF	0.766	0.9688	0.7354	0.9180
DeepHAMF	0.701	0.900	0.659	0.851
DNNMF	0.696	0.8434	0.6425	0.8245
AIMF	0.3201	0.4562	0.1847	0.2360

We can intuitively see that the AIMF model outperformed the other methods in terms of the MAE and RMSE metrics for both datasets. In Table 3, the recommendation performance of SVD++ with implicit factors was better than that of NMF with a single matrix factorization and reconstruction score. The performance of DeepHAMF with an attention mechanism based on implicit preferences was better than that of SVD++. Compared with DeepHAMF, DNNMF with MLP performed slightly better than DeepHAMF without an attention mechanism. Our model, AIMF, which combines the nonlinear expressive power of MLP with the attention interaction ability of different dimensions, outperformed the baselines above.

4.2.2. Effect of the Number of Neurons on Model Performance

In order to determine the number of neurons, we used group experiments. Under the condition that the number of hidden layers was three, we conducted three sets of experiments as follows:

(1) We determined the number of neurons in the first hidden layer. The experimental parameters were: 50, 100, 10; 100, 100, 10; and 150, 100, 10.
(2) We determined the second hidden layer. After determining the first hidden layer by (1), we determined the experimental parameters of this group: 150, 50, 10; 150, 100, 10; and 150, 50, 30.
(3) We identified the third hidden layer. After determining the number of neurons in the first two hidden layers by (1) and (2), we determined the experimental parameters of this group: 150, 100, 10; 150, 100, 20; and 150, 100, 30.

The experimental results for MovieLens100K and MovieLens1M are shown in Figures 5 and 6.

Figure 5 shows the results of each set of experiments on MovieLens100K. Figure 5a shows the results when the number of neurons in the first hidden layer is 50, 100, and 150, respectively (assuming that the numbers of neurons in the second and third layers are 100 and 10, respectively). It can be clearly seen that when the number of neurons in the first hidden layer is 150, the recommendation performance of the model is relatively excellent. Under the condition that Figure 5a the number of neurons in the first hidden layer is fixed, Figure 5b shows the results when the numbers of neurons in the second hidden layer are 50, 100, and 150, respectively. When the number of neurons in the second hidden layer is 100, the model recommendation performance is excellent. Figure 5c shows the results of the third

hidden layer with 10, 20, and 30 neurons. At this point, we determined the number of neurons in the first two layers to be 150 and 100. Finally, according to the results shown in Figure 5c, when the number of neurons in the third hidden layer is 10, the recommendation performance of the model is good. After we performed the experiments on groups (1), (2), and (3) one at a time, the final numbers of neurons in the MLP were 150, 100, and 10.

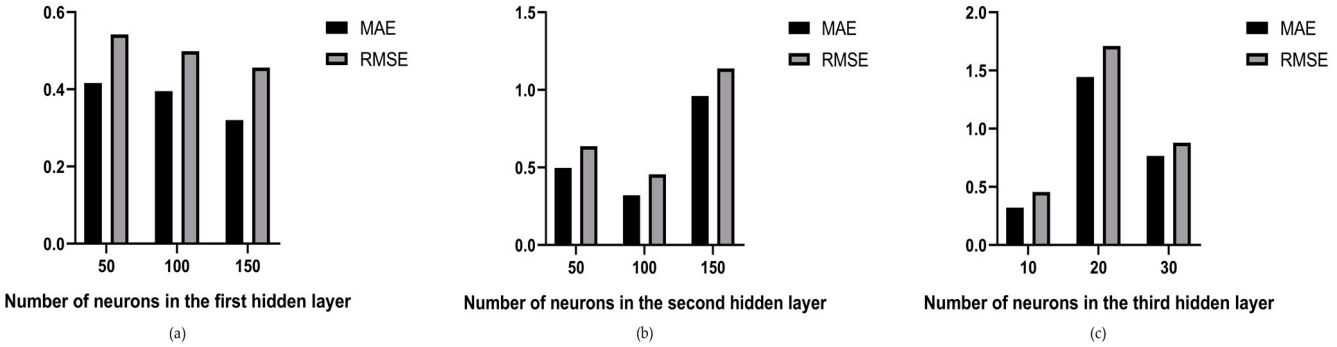


Figure 5. Three sets of experiments on MovieLens100K. The subfigure (**a**) shows that 100 neurons in the second hidden layer and 10 neurons in the third hidden layer are fixed, and 50, 100 and 150 neurons in the first hidden layer are respectively selected to verify the performance of the recommendation. The subfigure (**b**) shows that the first hidden layer 150 neurons and the third hidden layer 10 neurons are fixed, and the second hidden layer 50, 100, 150 neurons are taken respectively to verify the performance of the recommendation. The subfigure (**c**) shows that the first hidden layer of 150 neurons and the second hidden layer of 100 neurons are fixed, and the third hidden layer of 10, 20, 30 neurons are taken respectively to verify the performance of the recommendation.

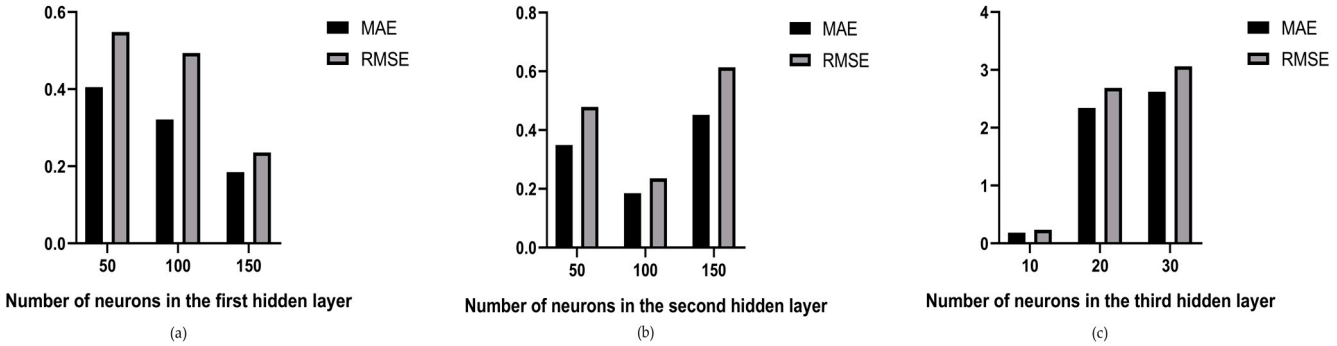


Figure 6. Three sets of experiments on MovieLens1M. The subfigure (**a**) shows that 100 neurons in the second hidden layer and 10 neurons in the third hidden layer are fixed, and 50, 100 and 150 neurons in the first hidden layer are respectively selected to verify the performance of the recommendation. The subfigure (**b**) shows that the first hidden layer 150 neurons and the third hidden layer 10 neurons are fixed, and the second hidden layer 50, 100, 150 neurons are taken respectively to verify the performance of the recommendation. The subfigure (**c**) shows that the first hidden layer of 150 neurons and the second hidden layer of 100 neurons are fixed, and the third hidden layer of 10, 20, 30 neurons are taken respectively to verify the performance of the recommendation.

Figure 6 shows the results of each set of experiments on MovieLens1M. Similar to the steps on MovieLens100K, we again experimented on groups (1), (2), and (3) one at a time. It can clearly be seen that the experimental results of MovieLens1M are basically the same as that of MovieLens100K. The final number of neurons in the hidden layer is 150, 100, and 10, respectively.

4.2.3. Effect of the Number of Hidden Layers on Model Performance

Based on the experiments in Section 4.2.2, we determined the number of neurons in each hidden layer. However, the above results are based on the assumption that the

number of hidden layers is three. Therefore, the main task presented in this section is the determination of the specific number of hidden layers.

The experimental setting of this subsection includes the following. The number of digits represents the number of layers, and the value of the number is the number of neurons. The order from left to right represents the first layer, the second layer, . . . For example, (150, 100) means two hidden layers with 150 neurons in the first hidden layer and 100 neurons in the second hidden layer.

(1) Number of hidden layers and neurons: (150, 100);
(2) Number of hidden layers and neurons: (150, 100, 10);
(3) Number of hidden layers and neurons: (150, 100, 10, 10);
(4) Number of hidden layers and neurons: (150, 100, 10, 10, 10).

The experimental results for MovieLens100K and MovieLens1M are shown in Figures 7 and 8.

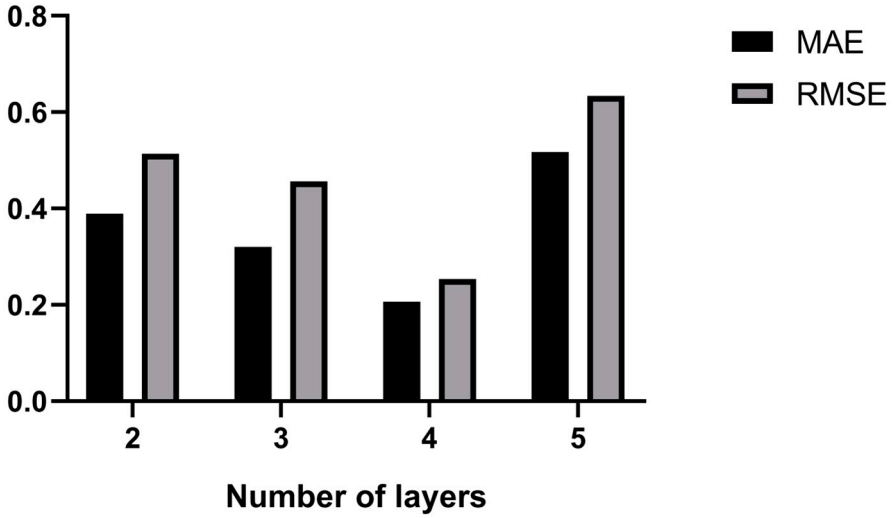


Figure 7. Experiments on MovieLens100K.

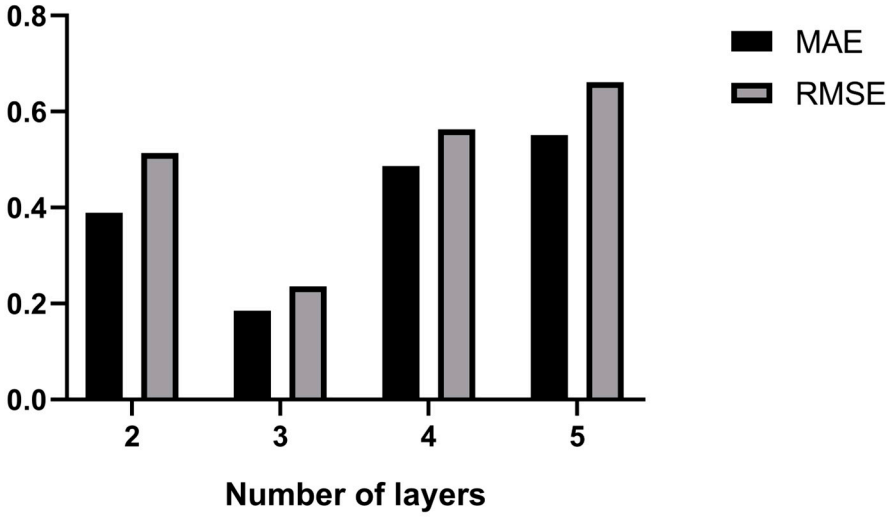


Figure 8. Experiments on MovieLens1M.

Figures 7 and 8 visually show the results of the four sets of experiments. It can be seen that the results on MovieLens100K are not consistent with those on MovieLens1M. On MovieLens100K, the recommendation performance of the model is better when the number of hidden layers is four, followed by when the number of hidden layers is three. On MovieLens1M, the recommendation performance is better when the number of hidden

layers is three, followed by two hidden layers. This indicates that the number of layers of the MLP is an important factor affecting the recommendation performance. In addition, the corresponding MAE and RMSE are better than the other benchmarks regardless of if the number of layers is three or four. Finally, considering the size of the dataset, we chose three hidden layers as the structure of the MLP.

4.3. Ablation Study

In this section, we present the designs of three variants of the model to verify the impact of different modules on the recommendation performance.

Submodules: AIMF-SS, AIMF-SM, and AIMF-MS.

- AIMF-SS: This includes the slide-attention and self-attention components and removes the MLP.
- AIMF-SM: This includes the slide-attention and MLP components and removes the self-attention.
- AIMF-MS: This includes the MLP and self-attention components and removes the slide-attention.

These sub-models were experimented on two datasets, and the experimental results are shown in Table 4.

Table 4. Comparison of the experimental results.

Method	MovieLens100K		MovieLens1M	
	MAE	**RMSE**	**MAE**	**RMSE**
AIMF-SS	0.2841	0.2366	0.3295	0.3851
AIMF-SM	0.7085	0.7433	0.7208	0.7540
AIMF-MS	0.1978	0.2466	0.1860	0.2355
AIMF	0.3201	0.4562	0.1847	0.2360

The results in Table 4 show that the overall performance of AIMF was higher than its variant model, and its variant model performed better than the baseline model. This indicates that the components in the model (slide-attention, MLP, self-attention) play an important role in improving the recommendation performance of the model. By comparing AIMF-SS and AIMF, we found that removing MLP made the variant model perform better than AIMF on MovieLens100K and slightly worse than AIMF on MovieLens1M, which may indicate that the size of the dataset has some influence on the role of MLP. By comparing AIMF-MS and AIMF, we found that removing slide-attention made the variant model perform better than AIMF on both datasets. However, compared with MovieLens100K, the performance of AIMF on MovieLens1M is significantly improved. At the same time, it can solve the problem of an attention interaction between users and item latent features of different dimensions. By comparing AIMF-SM and AIMF, we found that the recommendation performance of the variant model with self-attention removed was much worse than that of AIMF. This indicates that mining explicit user preferences is necessary.

5. Discussion

The deep learning model based on MLP has certain advantages for solving nonlinear problems. In particular, the traditional BP neural network is used to fit the model parameters, which makes the prediction of the model more extensive. Deep learning models based on an attention mechanism are gradually becoming widely used in recommendation systems, with the advantages of fewer parameters, easy training, and better establishment of the similarity relationship between two targets. AIMF combines the advantages of both and improves the recommendation performance in both implicit and explicit user preferences. In addition, compared with the existing attention mechanism, slide-attention makes some improvements based on general attention so that users and items with different

dimensional features can interact with attention, and a similarity relationship between them is obtained.

However, AIMF has the following limitations:

Firstly, it can be seen from the scale of the MoiveLens100K and MovieLens1M datasets that AIMF is more suitable for scenarios with a relatively large amount of data. Secondly, as the difference between the user feature dimension and the item feature dimension increases, its calculation time also increases. Finally, AIMF is currently suitable for the scoring mechanism in the film domain, but there is still a lot of room for improvement in other domains, such as music and e-commerce.

6. Conclusions

Among the many recommendation algorithms based on deep learning, knowledge graphing, and time decay, the research focus is on enhancing the personalized characteristics of recommendation systems. This study captures the characteristic information of user–item ratings and deeply mines users' personalized preferences through users' ratings on one to five items, so as to recommend items of interest to users. In this paper, we propose a novel AIMF model that integrates matrix factorization, MLP, an attention mechanism, and a self-attention mechanism to reflect the user's overall preference from two aspects: implicit preference and explicit preference. In addition, AIMF incorporates an improved attention algorithm to overcome the problem that the dimension of latent features between users and items is not the same, and the attention interaction cannot be directly established. Experiments were conducted on two movie domain public datasets, and the experimental results show that the proposed method outperformed several baselines by at least 54% in RMSE and by at least 46% in MAE.

With the development of information technology and the increasing needs of users, dynamic recommendation has become a mainstream research topic. Therefore, processing massive information in real time and making recommendations according to the dynamic preferences of users will be our next research topic. At the same time, applying the model to a wider field of recommendation systems is also a difficult problem we will work to overcome.

Author Contributions: Conceptualization, C.M. and Z.W.; methodology, C.M.; software, C.M.; validation, C.M., Y.L. and Z.S.; formal analysis, C.M.; investigation, Y.L.; resources, Z.S.; data curation, Z.S.; writing—original draft preparation, C.M.; writing—review and editing, Z.W.; visualization, C.M.; supervision, Z.W.; project administration, Z.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Humanities and Social Sciences Youth Foundation of the Ministry of Education of China (22YJC870018).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of this study; in the collection, analyses, or interpretation of data; in the writing of this manuscript; or in the decision to publish the results.

References

1. Wang, M.Q.; Tang, C.L. Content supply-side information Overload Problem and Optimization Strategy: A Case study of Internet content platform. *Sci-Tech Inf. Dev. Econ.* **2022**, *7*, 30–37.
2. Hu, Q.; Zhu, D.J.; Wu, H.L.; Wu, L.H. Survey on Intelligent recommender Systems. *Comput. Syst. Appl.* **2022**, *31*, 4758.
3. Huang, Y.H.; Wang, W.J.; Liu, H.; Zhou, Z.K. Research Progress on Over-specialization in Personalized Information Recommendation. *Inf. Sci.* **2022**, *40*, 185–192.
4. Zhang, S.Z.; Bai, Z.J.; Li, P.; Chang, Y.Y. Multi-Graph Convolutional Network for Fine-Grained and Personalized POI Recommendation. *Electronics* **2022**, *11*, 2966. [CrossRef]
5. Wang, W.; Du, Y.X.; Zheng, X.L.; Zhang, C. Neural Collaborative Recommendation Algorithm Based on Graph convolutional Self-attention Mechanism. *Comput. Eng. Appl.* **2023**, *59*, 247–258.
6. Li, J.; Kameda, H. Load balancing problems for Multi-class jobs load balancing problems for multi-class jobs in distributed parallel computer systems. *IEEE Trans. Comput.* **1998**, *47*, 322–333.

7. Schafer, J.B.; Konstan, J.; Riedl, J. Recommender systems in e-commerce. In Proceedings of the 1st ACM Conference on Electronic Commerce, Denver, CO, USA, 3–5 November 1999; pp. 158–166.
8. Alberto, C.; Fabio, R. Recommender systems by means of information retrieval. In Proceedings of the International Conference on Web Intelligence, Mining and Semantics, Sogndal, Norway, 25–27 May 2011; p. 57.
9. Francesco, R.; Lior, R.; Bracha, S. Recommender systems: Introduction and Challenges. In *Recommender Systems Handbook*; Springer: New York, NY, USA, 2015; pp. 1–34.
10. Gediminas, A.; Alexander, T. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749.
11. Jure, L.; Anand, R.; Jeffrey, D.U. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, UK, 2014; pp. 307–326.
12. Balabanovic, M.; Shoham, Y. Fab: Content-Based, collaborative recommendation. *Commun. ACM* **1997**, *40*, 66–72. [CrossRef]
13. Linden, G.; Smith, B.; York, J. Amazon.com recommendations: Item-to-Item collaborative filtering. *IEEE Internet Comput.* **2003**, *7*, 76–80. [CrossRef]
14. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–38. [CrossRef]
15. Yang, B.C.; Li, Y.Z. Deep collaborative filtering recommendation algorithm Fusing explicit and Implicit Features. *J. Liaoning Tech. Univ. Nat. Sci.* **2023**, *42*, 354–361.
16. Wu, W.; Xu, S.S.; Guo, S.S.; Li, X.Y. Research on point-of-interest combined recommendation algorithm based on location-based social network. *Netinfo Secur.* **2023**, *23*, 75–84.
17. Lou, L. Research on the Implementation of College Graduate Recommendation System Based on Collaborative Filtering Algorithm. *China Internet Wkly.* **2023**, *15*, 37–39.
18. Zhang, Y.H.; Zheng, J.Y.; Liu, L. Design and implementation of job recommendation system based on collaborative filtering algorithm. *Mod. Comput.* **2023**, *29*, 109–112.
19. Guo, X.Y.; Shen, Y.Q.; Cui, Y. Collaborative filtering recommendation algorithm based on fuzzy clustering and user interest. *Softw. Guide* **2023**, *22*, 124–131.
20. Gu, Y.R.; Shi, J.W.; Huang, L.Y. Neural Collaborative Filtering Recommendation Algorithm Based on Multi-head graph Attention Mechanism. *J. Chin. Comput. Syst.* **2023**, *18*, 1–9.
21. Zhang, Q.; Yu, J.; Zhang, T.; Li, Y.F. Research on Knowledge Recommendation Model Based on Collaborative Filtering. *Softw. Eng.* **2023**, *26*, 36–39.
22. Good, N.; Schafer, J.B.; Konstan, J.A.; Borchers, A.; Sarwar, B.; Herlocker, J.; Riedl, J. Combining collaborative filtering with personal agents for better recommendations. In Proceedings of the National Conference on Artificial Intelligence, Orlando, FL, USA, 18–22 July 1999; pp. 439–446.
23. Gulzar, Y.; Alwan, A.A.; Abdullah, R.M.; Abualkishik, A.Z.; Oumrani, M. OCA: Ordered Clustering-Based Algorithm for E-Commerce Recommendation System. *Sustainability* **2023**, *15*, 2947. [CrossRef]
24. Wu, J.C.; Wang, X.; Gao, X.Y.; Chen, J.W.; Fu, H.C.; Qiu, T.Y.; He, X.N. On the Effectiveness of Sampled Softmax Loss for Item Recommendation. *arXiv* **2022**, arXiv:2201.02327. [CrossRef]
25. Wang, J.; Liu, L. A multi-attention deep neural network model base on embedding and matrix factorization for recommenda-tion. *Int. J. Cogn. Comput. Eng.* **2020**, *1*, 70–77.
26. Zhang, C.K.; Wang, C. Probabilistic Matrix Factorization Recommendation of Self-Attention Mechanism Convolutional Neural Networks with Item Auxiliary Information. *IEEE Access* **2020**, *8*, 208311–208321. [CrossRef]
27. Guan, F.; Zhou, Y.; Zhang, H. Research on optimization of collaborative filtering recommendation algorithm in personalized recommendation system. *Oper. Res. Manag. Sci.* **2022**, *31*, 9–14.
28. Jin, Y.; Chen, H.M.; Luo, C. Interest Capture Recommendation Algorithm Based on Knowledge Graph. *Comput. Sci.* **2023**, *18*, 1–14.
29. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Incremental singular value decomposition algorithms for highly scalable recommender systems. In Proceedings of the Fifth International Conference on Computer and Information Science, Seoul, Republic of Korea, 28–29 November 2002; Volume 1, pp. 27–28.
30. Simon, F. Funk-SVD[EB/OL]. Available online: http://sifter.org/~simon/journal/20061211.html (accessed on 1 January 2024).
31. Koren, Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *Knowl. Discov. Data* **2010**, *4*, 1–24. [CrossRef]
32. Koren, Y. Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 447–456.
33. Li, X.Y.; Xu, S.H.; Jiang, T.; Wang, Y.; Ma, Y.; Liu, Y.M. POI Recommendation Method of Neural Matrix Factorization Integrating Auxiliary Attribute Information. *Mathematics* **2022**, *10*, 3411. [CrossRef]
34. He, X.N.; Liao, L.Z.; Zhang, H.W.; Nie, L.Q.; Hu, X.; Chua, T.S. Neural Collaborative Filtering. In Proceedings of the International World Wide Web Conferences Steering Committee, Perth, Australia, 3–7 May 2017. [CrossRef]
35. Tian, Z.; Pan, L.M.; Yin, P.; Wang, R. Deep Matrix Factorization Recommendation Algorithm. *J. Softw.* **2021**, *32*, 3917–3928.
36. Vaswani, A.; Shazeer, N.M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. *Neural Inf. Process. Syst.* **2017**, *30*.
37. Wang, Z.A.; Li, Y.L.; Shang, Z.Y.; Li, G.B. Attention-based Convolutional Neural Networks for CAPTCHA Recognition. *J. Southwest Minzu Univ. Nat. Sci. Ed.* **2023**, *49*, 303–311.

38. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
39. Zhou, X.Z.; Li, S.; Sui, D. Microblog Sentiment Analysis Based on Deep Learning and Attention Mechanism. *J. Nanjing Norm. Univ. Nat. Sci. Ed.* **2023**, *46*, 115–121.
40. Zhou, L.M.; Zhou, D.M. Real image denoising based on attention mechanism and residual block. *Comput. Eng. Des.* **2023**, *44*, 1451–1458.
41. Xu, S.S.; Zhuang, H.Y.; Sun, F.Z.; Wang, S.Q.; Wu, T.H.; Dong, J.W. Recommendation algorithm of probabilistic matrix factorization based on directed trust. *Comput. Electr. Eng.* **2021**, *93*, 107206. [CrossRef]
42. Jia, Y.H.; Liu, H.; Hou, J.H.; Kwong, S. Semisupervised adaptive symmetric non-negative matrix factorization. *IEEE Trans. Cybern.* **2020**, *51*, 2550–2562. [CrossRef]
43. Luo, X.; Liu, Z.G.; Shang, M.S.; Lou, J.G.; Zhou, M.C. Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization. *IEEE Trans. Netw. Sci. Eng.* **2020**, *8*, 463–476. [CrossRef]
44. Gündüz, N.; Fokoué, E. Understanding students' evaluations of professors using non-negative matrix factorization. *J. Appl. Stat.* **2021**, *48*, 2961–2981. [CrossRef]
45. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
46. Chen, J.Y.; Zhang, H.W.; He, X.N.; Nie, L.Q.; Liu, W.; Chua, T.S. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17), Tokyo, Japan, 7–11 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 335–344.
47. Mao, M.Y.; Wu, C.; Zhong, Y.X.; Chen, Z.C. BERT named entity recognition model with self-attention mechanism. *CAAI Trans. Intell. Syst.* **2020**, *15*, 772–779.
48. Wang, A.Y.; Meng, Q.F.; Wang, M.B. Spectrum sensing method based on residual neural network and attention mechanism. *Radio Eng.* **2023**, *18*, 7791.
49. He, K.M.; Zhang, X.Y.; Ren, S.Q.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
50. Harper, F.M.; Konstan, J.A. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* **2015**, *5*, 19. [CrossRef]
51. Kumar, P.; Thakur, R.S. Recommendation system techniques and related issues: A survey. *Int. J. Inf. Technol.* **2018**, *10*, 495–501. [CrossRef]
52. Lee, D.D.; Seung, H.S. Learning the parts of objects by non-negative matrix factorization. *Nature* **1999**, *401*, 788–791. [CrossRef] [PubMed]
53. Li, J.H.; Su, X.Q.; Wu, C.H. Deep Hierarchical Attention Matrix Factorization. *Comput. Eng. Sci.* **2023**, *45*, 28–36.
54. Behera, G.; Nain, N. DeepNNMF: Deep nonlinear non-negative matrix factorization to address sparsity problem of collabora-tive recommender system. *Int. J. Inf. Technol.* **2022**, *14*, 3637–3645.
55. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.