information

MDPI

Article

A Rating Prediction Recommendation Model Combined with the Optimizing Allocation for Information Granularity of Attributes

Jianfei Li [1,*], Yongbin Wang [2] and Zhulin Tao [3]

[1] Collaborative Innovation Center, Communication University of China, Beijing 100024, China
[2] State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China; ybwang@cuc.edu.cn
[3] School of Information and Communication, Communication University of China, Beijing 100024, China; taozl@cuc.edu.cn
* Correspondence: lijianfei@cuc.edu.cn

Abstract: In recent years, graph neural networks (GNNS) have been demonstrated to be a powerful way to learn graph data. The existing recommender systems based on the implicit factor models mainly use the interactive information between users and items for training and learning. A user–item graph, a user–attribute graph, and an item–attribute graph are constructed according to the interactions between users and items. The latent factors of users and items can be learned in these graph structure data. There are many methods for learning the latent factors of users and items. Still, they do not fully consider the influence of node attribute information on the representation of the latent factors of users and items. We propose a rating prediction recommendation model, short for LNNSR, utilizing the level of information granularity allocated on each attribute by developing a granular neural network. The different granularity distribution proportion weights of each attribute can be learned in the granular neural network. The learned granularity allocation proportion weights are integrated into the latent factor representation of users and items. Thus, we can capture user-embedding representations and item-embedding representations more accurately, and it can also provide a reasonable explanation for the recommendation results. Finally, we concatenate the user latent factor-embedding and the item latent factor-embedding and then feed it into a multi-layer perceptron for rating prediction. Extensive experiments on two real-world datasets demonstrate the effectiveness of the proposed framework.

Keywords: recommender systems; graph neural networks; rating prediction; granular neural network; latent factor

Citation: Li, J.; Wang, Y.; Tao, Z. A Rating Prediction Recommendation Model Combined with the Optimizing Allocation for Information Granularity of Attributes. *Information* **2022**, *13*, 21. https://doi.org/10.3390/info13010021

Academic Editor: María N. Moreno García

Received: 22 December 2021
Accepted: 30 December 2021
Published: 5 January 2022

1. Introduction

With the rapid development of the Internet, especially the mobile Internet, the amount of information on the Internet is growing explosively. Obtaining meaningful content for users from a large amount of digital information has become a hot research task. A personalized recommendation can help users obtain useful information efficiently and accurately in massive information resources. The collaborative filtering framework is often used in recommender systems. For example, some recommender models utilize matrix factorization to learn the representations of users and items based on their historical interactions [1,2]. The typical models include matrix factorization (MF) [3,4], probabilistic matrix factorization (PMF) [5,6], singular value decomposition (SVD) [5,6], Bayesian personalized ranking (BPR) [7], etc. However, matrix factorization-based recommender systems can only use matrix scores for modeling and recommending, while we cannot fully utilize some important feature information. In 2010, Rendle proposed an FM algorithm, aiming at the problem that sparse data has no interaction and cannot estimate parameters. The model

characterizes each eigenvalue by embedding it into the implicit vector and characterizes the combination between the two features by the inner product between the two implicit vectors [8]. DCF is an early hybrid recommendation algorithm combining probability matrix decomposition and a noise-reduction self-encoder [9]. In 2016, deep-learning technology began to be applied in the field of recommender systems. Referring to the idea of word2vec, the item2vec algorithm takes the user's behavior sequence as a sentence for representation and learning [10]. DeepFM can capture the low-order intersections of features and make full use of the depth model to extract and capture high-order features to achieve better generalization ability [11]. Many classical models have been proposed in the same period, such as NFM, AFM, etc. [12,13].

With the development of graph neural networks, graph neural networks have proved to be able to learn graph structure data [14–17], which provides a new research idea for the development of recommendation systems. For example, NeuMF, proposed by Xiangnan He [18], is a state-of-the-art matrix factorization model with a neural network architecture. IGMC, proposed by Zhang M, is an inductive matrix completion model that does not use side information, which trains a graph neural network base on 1-hop subgraphs [19]. In order to solve the link prediction problem on bipartite graphs, a self-coding framework is used for graph information aggregation in GCMC [20]. The NGCF model proposed by Wang x et al. [21] exploits the user–item graph structure by propagating embeddings. This leads to the expressive modeling of high-order connectivity in the user–item graph, effectively injecting the collaborative signal into the embedding process in an explicit manner. GraphRec, proposed by Wenqi fan et al. [22], is based on the architecture of matrix decomposition. While generating user-hidden vectors and item-hidden vectors, it aggregates the information of user–item interaction graphs and social network graphs through a deep neural network and attention mechanism.

One of the most critical challenges faced is considering user and item attributes to enhance the prediction performance. Learning node embeddings (user or item) and attribute embeddings have proven essential in providing useful information for more accurate predictions [23,24]. In the task of the recommendation system, the user–item interaction includes the user's scoring data of the item, which is typical graphic data. The main idea is how to use a neural network to iteratively aggregate the feature information from the neighborhood of the local graph. The node information can be propagated again after aggregation. In the case of the social relationships among users cannot be obtained easily, it is particularly important to use user attribute information and item attribute information to represent the embedding of users and items. For example, the AFM (attentional factorization machine) model adopts the attention mechanism to improve the performance compared with the NFM model. Graphrec also introduces the attention mechanism to calculate the similarity between users and items through a multi-layer perceptron. Chen S et al. propose a deep-learning framework for explicit recommender systems, ACAE, based on the denoising autoencoder, which can extract the global latent factors in a non-linear fashion from the sparse rating matrices [25]. However, in the traditional methods for aggregating features of different dimensions, the influence of the node's attributes on the representation of latent factors of users and items is not fully considered, let alone the influence degree of different feature attributes on the representation of users and items. For example, for the same user, it is also to buy lipstick. Whether the user is male or female, the representation of the user's image or characteristics will be different. Moreover, the behavior of a 20-year-old woman buying lipstick is completely different from that of a 70-year-old man.

Recently, the application of granular neural networks on pattern recognition and remote sensing image classification has been explored with various optimization methods in [26,27]. There also have been some studies about system modeling with granular space of parameters. Song et al. developed a granular input space for neural networks to evaluate the influence of input features (variables) on the output results [28]. By distributing information granularities across input variables, the output of a network has different

levels of sensitivity on various input variables. Capturing the relationship between input variables and output result becomes of great help for mining knowledge from the data. In this way, important features of the data can be easily found. We employ interval mathematics in neural networks for input granularity of a user's attributes and an item's attributes. The different granularity distribution of the attributes indicates that they are of different importance. Therefore, in this paper, we use the user's attributes and behavior to aggregate the latent factors for representing a user's preference vector. We use the item's attributes and the item's behavior to aggregate the latent factors for representing an item's preference vector.

2. The Proposed Framework

In the proposed recommendation system, as shown in Figure 1, the user–item graph, user–attribute graph, and item–attribute graph are constructed according to the interactive information between users and items.

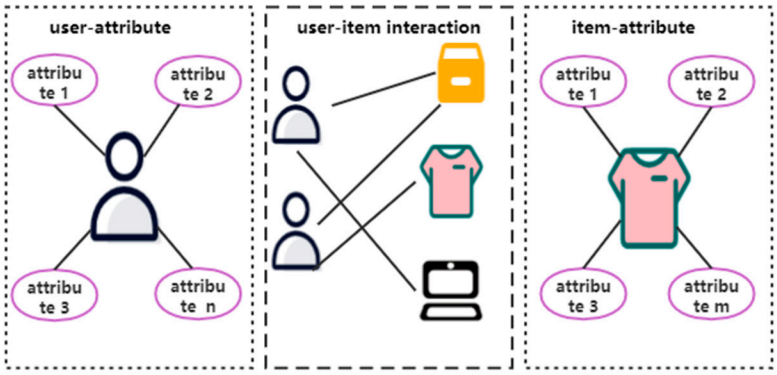

Figure 1. Graph Data contains the user–item graph and the user–attribute graph, and the item–attribute graph.

As can be seen from the above figure, according to the interaction relationship of data, we can get the interaction behavior representation of all users and items and the attribute representation of all users and items. From the user–item interaction graph, we can capture the user's preference information for items. For example, user 1 likes laptops, and user 2 likes T-shirts. At the same time, combining with the user's attribute information, it can also help capture the user's preference information. For example, user 1 is a male, age 32, and an IT engineer. From the attribute information, we can also learn and capture some behavior preference characteristics of the user. With the help of graph neural network technology, in this paper, we can learn the hidden latent factors representation of users by aggregating user–item interaction graphs and user–attribute graphs. By aggregating the user–item interaction graph and item–attribute graph, we can learn the hidden latent factors representation of the items. Secondly, we use a granular neural network to learn the optimal allocation for information granularity. We integrate the proportional weight of the optimal allocation into the user and item implicit factor representation to capture the user's embedding vector and the item's embedding vector more accurately. Finally, after concatenating the user latent factor-embedding and the item latent factor-embedding, the final rating prediction of the model is obtained through a multi-layer perceptron calculation.

2.1. Granular Neural Networks

For traditional neural networks, we have done extensive research. Usually, the conventional neural network can only deal with crisp numerical input–output data. Compared with the traditional point-valued neural network, the research of a general interval neural network is of great significance to the theoretical system of the neural network [29]. When processing data, the granular output of granular neural network can provide an interval prediction result instead of only a point-value output, so as to provide different levels and

comprehensive guidance for data evaluation. We augment neural networks by distributing information granularities across the user attributes and item attributes and assigning different information granularities to each attribute. Finally, an improved Parthenon genetic algorithm is used to optimize the allocation of information granularity. We integrate the proportional weight of the optimal allocation into the user- and item-implicit factor representation to capture the user's embedding vector and the item's embedding vector more accurately. The structure of the granular neural network is shown in Figure 2, below.

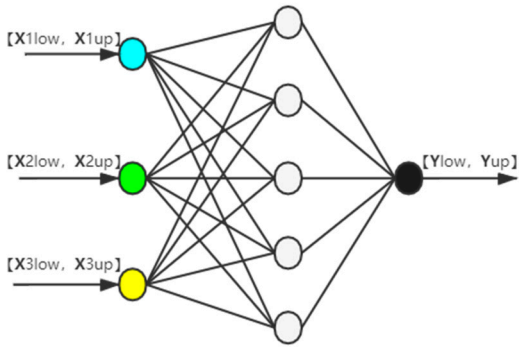

Figure 2. A neural network with interval input space.

2.1.1. The Interval of Information Granules

If "ε" is given, the core issue of allocation of information granularity is to discover the best group of sub-granularities that optimizes a given objective function. The goal of the allocation of information granularity is to discover the preference of input features. We also require that the following overall summation formula of granularity should be retained, which is expressed in the form:

$$\varepsilon * n = \sum_{k=1}^{n} \varepsilon_n \tag{1}$$

where $\varepsilon_i (i\, \epsilon [1, n])$ is the information granularity bound with different input characteristics in a group of data, which are the core problems in the subsequent optimization problems, and N is the total number of granularities. If the optimized values of $\varepsilon_1\ \varepsilon_2 \ldots\ \varepsilon_n$ differ greatly, it means different granularities are allocated for different attributes, and different attributes have different effects on the output results. Here, intervals are used as the framework of information granules. The constructed intervals are described as follows:

$$x\ \rightarrow [x - \varepsilon_{-} * range,\ x + \varepsilon_{+} * range] \tag{2}$$

where the "*range*" is the range of the variable x (maximum value of x subtracts the minimum value of x). In Formula (2), ε_{-} is the granularity associated with the lower part of the interval, and ε_{+} is the granularity associated with the upper part of the interval.

2.1.2. Training Objectives

For traditional neural networks, our goal is to minimize the error E between the output and the actual output, as follows:

$$E = \sum (y_{target} - y)^2 \tag{3}$$

where y_{target} represents the output value of the neural network. Our goal is to minimize the error E between the output and the actual output.

In our study, each input attribute feature is expanded into an interval by the formula:

$$\varepsilon_1 + \varepsilon_2\ \ \ldots\ldots + \varepsilon_n = n\ *\ \varepsilon\quad \varepsilon_j \epsilon [0, 1] \tag{4}$$

After the input space is partitioned, its corresponding output space also becomes an interval. For each instance in the input space, after the GNN operation, there is a corresponding output interval (YG), which can be expressed as follows:

$$yg_i = \left[yg_{ilow}, yg_{iup}\right] \tag{5}$$

Our goal is to optimize the granularity distribution to maximize the specificity of the interval output and the coverage of the interval output. The objective function mainly depends on specificity and coverage. Here, the specificity is expressed by the length of the output interval, and the expression formula is as follows:

$$Q_1 = \frac{1}{e^{\frac{\sum_{i=1}^{p}(yg_{iup} - yg_{ilow})}{p}}} \tag{6}$$

The coverage is defined as the ratio of the target output falling into the output interval. For example, there are p instances in dataset D. D_e is the dataset obtained after interval expansion. T is the output interval obtained after GNN operation, and its total number is p. The definition of coverage is the number p_{in} of P granularity output intervals obtained by GNN operation for the corresponding real data of P instances in dataset D. The coverage can be expressed by the following formula:

$$Q_{cover} = \frac{p_{in}}{p} \tag{7}$$

These two criteria, depending on their characters, are more likely in conflict. Considering this, a two-objective combined optimization is optional. The objective function can be set as:

$$Q = Q_{cover} * Q_1 \tag{8}$$

For traditional neural networks, for each instance of the input data, there is only one output, which is denoted as y_{target} in Formula (3). In our following study, each input feature is expanded into an interval by Formula (4). For each input feature, we assign a granularity $\varepsilon_i (i \, \epsilon [1, n])$, where n is the total number of input features. Thus, there is a corresponding interval output (yg_i) obtained by the operation with granular neural networks for each instance of the input data. As expressed in Formula (4), yg_i is an interval output. What we aim to do in this study is to optimize the distribution of ε_i so that the specificity of interval outputs is maximized (and the coverage of interval outputs is maximized), where the specificity and the coverage is expressed as Q_1 and Q_{cover}.

In the process of training the granularity neural network, we use an improved Partheno genetic algorithm to continuously optimize the granularity allocation until the Q value of the above formula is optimal.

2.2. The Overall Architecture of the Proposed Model

Based on the interaction between users and items, in this paper, we construct a user–item graph, a user–attribute graph, and an item–attribute graph. We can learn the latent factor of the user's attribute side from the user–attribute graph and learn the latent factor of the user's behavior side from the user–item interaction graph. Finally, we connect the aggregated attribute side feature vector and behavior side feature vector to generate the final user's latent factor feature vector. The granularity optimal allocation proportion weight of user's attributes is optimized through granularity neural network training, which is integrated into the attention mechanism. By adding the granularity optimal allocation proportion weight of attributes, the SoftMax function can enlarge or reduce the original input and generate a new weight set to characterize the user's hidden factor vector more accurately. Similarly, we also can learn the latent factor of the item's attribute side from the item–attribute graph and learn the latent factor of the item's behavior side from the user–item interaction graph. Finally, we connect the aggregated attribute side feature vector

and behavior side feature vector to generate the final item's latent factor feature vector. The prediction score of the proposed recommendation algorithm model is generated by aggregating the user's feature vector and item's feature vector. The overall framework of the model is shown in Figure 3, below:

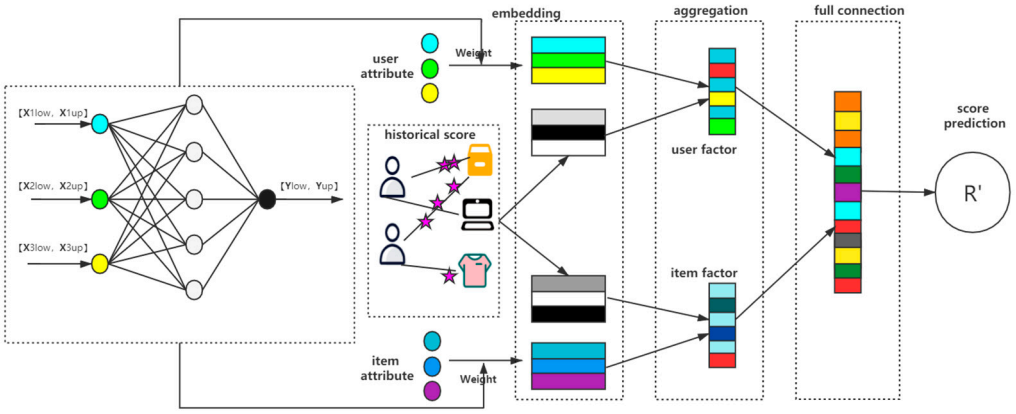

Figure 3. The overall architecture of the proposed model.

2.2.1. User Latent Factors

The user latent factor vector can be captured from the user–attribute graph and user–item interaction graph. From the user–attribute graph, we can capture the latent factor representation vector of the user's attribute side. From the user–item interaction graph, we can capture the latent factor representation vector of the user's behavior side and aggregate the two latent factor vector forms to obtain the final user latent factor vector.

Latent factor of the user's attribute side. Different users can represent different user attribute side latent factors through the embedding representation of user attribute information. For each user, we aggregate the user's attribute information to obtain the user's attribute side latent factor, shown as the following function:

$$U^{pro} = \sigma\left(W \cdot Att_{Upro}\left(h_p^I, \forall p \in M(i)\right) + b\right) \tag{9}$$

where U^{pro} denotes the latent factor of the user's attribute side. h_p^I denotes the embedding vector of each attribute P of user I, and M is the attribute sets of users I. The total number of attributes is m. σ is the non-linear activation function; W and b are the weight and bias of a neural network, respectively. Att_{Upro} is the user's attribute aggregation function. According to the attention mechanism [21], each user's attribute vector can be assigned a personalized weight. The latent factor of the user's attribute side is defined with the following function:

$$U^{pro} = \sigma(W \cdot \left\{\sum_{p=0}^{m} a_i * h_p^I\right\} + b) \tag{10}$$

where a_i denotes the attention weight of the attribute p in contributing to the latent factor of the user's attribute side when characterizing the user's preference for the attributes. σ denotes the non-linear activation function, and W and b are the weight and bias of a neural network, respectively. Here, we use the attention mechanism proposed in this paper, which combines with the optimizing allocation for information granularity of user attributes by developing a granular neural network. The calculation formula is as follows:

$$a^*{}_i = W_2^T \cdot \sigma\left(W_1 \cdot \left[h_p^I \oplus P_i\right] + b1\right) + b2 \tag{11}$$

$$a_i = \frac{\varepsilon_k * a^*{}_i}{\sum_{k \epsilon M(i)} exp(\varepsilon_k * a^*{}_i)} \tag{12}$$

$$\varepsilon * n = \sum_{k=1}^{n} \varepsilon_k \tag{13}$$

where P_i is the embedding layer representation vector of user I. $\oplus$ denotes the concatenation operation between two vectors. W_1 and $b1$ are the weight and bias of the first layer of a neural network, and W_2 and $b2$ are the weight and bias of the second layer of a neural network, respectively. n is the number of user's attributes, and ε_k is the level of information granularity allocated on each attribute of the user.

Latent factor of the user's behavior side. The user–item graph contains interactions between users and items and users' rating scores on items. Inspired by the GraphRec model, we can capture the user's behavior latent factor vector in the user–item graph to get user preferences by considering the items with which the user interacts and the user's scoring on these items:

$$U^{act} = \sigma(W \cdot Att_{Uact}\ (h_{ia}, \forall a \in C(i)) + b) \tag{14}$$

Similarly, a personalized weight can be assigned to each user's opinion-aware interaction vector, expressed in the following form:

$$U^{act} = \sigma(W \cdot (\sum_{k=0}^{n} \beta_i h_{ia} + b)) \tag{15}$$

where h_{ia} is a representation vector to denote opinion-aware interaction between user i and an item a, which is to aggregate the user's interaction with the item and the user's score on the item to represent the latent factor vector on the user's behavior side. $C(i)$ is the set of items that user i has interacted with. β_i represents the contribution of different opinion-aware interaction vectors to characterize the latent factor vector of user's behavior side.

A user can express his/her opinions or rating scores to items during user–item interactions, denoted as r. These opinions on items can capture users' preferences for items, which can help model latent factors of a user's behavior side. We introduce an opinion-embedding vector, $E_r \in R^d$, which denotes each opinion r as a dense vector representation. For a user's opinions on the items, we combine item-embedding q_a and opinion-embedding E_r via a Multi-Layer Perceptron (MLP) to construct an opinion-aware interaction model; f is the aggregation function to fuse the interaction information with the opinion information, as follows:

$$h_{ia} = f([q_a \oplus E_r]) \tag{16}$$

In the above formula, f is the aggregation function to fuse the interaction information with the opinion information as shown.

The attention contribution of different behaviors on the user behavior side to the user latent factor vector in Formula (15) is calculated as follows:

$$\beta^*{}_i = W_2^T \cdot \sigma(W_1 \cdot [h_{ia} \oplus P_i] + b1) + b2 \tag{17}$$

$$\beta_i = \frac{\beta^*{}_i}{\sum_{k \epsilon M(i)} exp(\varepsilon_k * \beta^*{}_i)} \tag{18}$$

Finally, the latent factor of the user's attribute side and the latent factor of the user's behavior side need to be considered together since the user–attribute graph and the user–item graph provide information about users from different perspectives. We combine these two latent factors to the final user latent factor via a standard MLP where the latent factor of the user's attribute side U^{pro} and the latent factor of user's behavior side U^{act} are concatenated before feeding into MLP. Formally, the user latent factor is defined as follows, where $\oplus$ denotes the concatenation operation between two vectors and l is the index of a hidden layer:

$$U_1 = \left[U^{pro} \oplus U^{act}\right] \tag{19}$$

$$U_2 = \sigma(W_2 \cdot U_1 + b_2) \tag{20}$$

$$\cdots\cdots$$

$$U = \sigma(W_l \cdot U_{l-1} + b_{l-1}) \tag{21}$$

2.2.2. Item Latent Factors

The item latent factor vector can be captured from the item–attribute graph and the user–item interaction graph. From the item–attribute graph, we can capture the latent factor representation vector of item's attribute side. From the user–item interaction graph, we can capture the latent factor representation vector of item's interaction. Finally, we aggregate the two latent factor vectors to obtain the final item latent factor vector.

Latent factor of the item's attribute side. Different items can represent different item attribute side latent factors through the embedding representation of item attribute information. For each item, we aggregate the item's attribute information to obtain the latent factor of the item's attribute side, shown as the following function:

$$I^{pro} = \sigma\Big(W \cdot Att_{Ipro}\left(h_q^J, \forall q \in G(j)\right) + b\Big) \tag{22}$$

According to the attention mechanism [30], each item's attribute vector can be assigned a personalized weight. The latent factor of an item's attribute side is defined as the following function:

$$I^{pro} = \sigma(W \cdot \{\sum_{q=0}^{n} \mu_j * h_q^J\} + b) \tag{23}$$

where h_q^J denotes the embedding vector of each attribute q of Item J, and G is the attribute sets of Item J; the total number of attributes is k. μ_j denotes the attention weight of the attribute q in contributing to the latent factor of item's attribute side when characterizing item's characteristics from the attributes. Here, we also use the attention mechanism proposed in this paper, which is combined with the optimizing allocation for information granularity of item attributes by developing a granular neural network. The calculation formula is as follows:

$$\mu^*{}_j = W_2^T \cdot \sigma\Big(W_1 \cdot \left[h_q^J \oplus Q_j\right] + b1\Big) + b2 \tag{24}$$

$$\mu_j = \frac{\varepsilon_k * \mu^*{}_j}{\sum_{k \epsilon G(j)} exp\left(\varepsilon_k * \mu^*{}_j\right)} \tag{25}$$

$$\varepsilon * n = \sum_{k=1}^{n} \varepsilon_k \tag{26}$$

where k is the number of item's attributes, and ε_k is the level of information granularity allocated on each attribute of the item. Q_j is the embedding layer representation vector of item J.

Latent factor of the item's behavior side. We can capture the item's latent factor vector in the user–item graph according to the item's interaction and scoring on them. In the user–item graph, we can capture the latent factor of the item's behavior side according to the interaction and scoring information from different users. Users may express different opinions or rating scores on the item in different periods, even for the same item. With the help of these opinions from different users, we can capture the characteristics of the item in different ways provided by users, which is helpful to model the latent factors of the item:

$$I^{act} = \sigma\big(W \cdot Att_{Iact}\left(f_{jt}, \forall t \in B(j)\right) + b\big) \tag{27}$$

Similarly, a personalized weight can be assigned to each item's opinion-aware interaction vector, expressed in the following form:

$$I^{act} = \sigma(W \cdot (\sum_{k=0}^{n} \gamma_{jt} f_{jt} + b)) \tag{28}$$

where f_{jt} is a representation vector to denote opinion-aware interaction between an item j and a user t, which is to aggregate the item's interaction with the user and the user's score on them to represent the latent factor vector on the item's behavior side. $B(j)$ is the set of users who has interacted with the item j. γ_{jt} represents the contribution of different opinion-aware interaction vectors to characterize the latent factor vector of the item's behavior side.

We combine user-embedding u_j and opinion-embedding E_r via a Multi-Layer Perceptron; f is the aggregation function to fuse the interaction information with the opinion information, as follows:

$$f_{jt} = f\big(\big[u_j \oplus E_r\big]\big) \tag{29}$$

The attention contribution of different users' behaviors on the item to the item latent factor vector in Formula (28) is calculated as follows:

$$\gamma^*{}_{jt} = W_2^T \cdot \sigma\big(W_1 \cdot \big[f_{jt} \oplus u_j\big] + b1\big) + b2 \tag{30}$$

$$\gamma_{jt} = \frac{\gamma^*{}_{jt}}{\sum_{t \epsilon B(j)} exp\big(\gamma^*{}_{jt}\big)} \tag{31}$$

Finally, the latent factor of the item's attribute side and the latent factor of the item's behavior side need to be considered together since the item–attribute graph and the user–item graph provides information about items from different perspectives. We combine these two latent factors to the final item latent factor via a standard MLP, where the latent factor of the item's attribute side I^{pro} and the latent factor of item's behavior side I^{pro} are concatenated before feeding into MLP. Formally, the item latent factor is defined as follows:

$$I_1 = \big[I^{pro} \oplus I^{act}\big] \tag{32}$$

$$I_2 = \sigma(W_2 \cdot I_1 + b_2) \tag{33}$$

$$\cdots\cdots$$

$$I = \sigma(W_l \cdot I_{l-1} + b_{l-1}) \tag{34}$$

2.2.3. Rating Prediction

In this paper, our proposed model is designed for the rating prediction recommendation task. With the latent factors of users and items (i.e., U and I), we can first concatenate them and then feed it into MLP for rating prediction as:

$$R_1 = [U \oplus I] \tag{35}$$

$$R_2 = \sigma(W_2 \cdot R_1 + b_2) \tag{36}$$

$$\cdots\cdots$$

$$R_l = \sigma(W_{l-1} \cdot R_{l-1} + b_{l-1}) \tag{37}$$

$$R' = W^t * R_l \tag{38}$$

where l is the index of a hidden layer and R' is the predicted rating.

3. Experiment

3.1. Dataset Description

We present a series of numeric experiments using the following two datasets: Book-Crossing and MovieLens-1M, containing user and item attributes. The MovieLens-1M dataset is a classic film-scoring dataset obtained from the MovieLens film-scoring recommendation website. It contains nearly one million scoring records of 3706 films by 6040 users. The users' ratings of books range from 0 to 10. The Book-Crossing dataset is the book scoring data compiled by Cai Nicolas Ziegler according to the data of a Book social networking website. In this website system, each book is given a special identity in order to track and connect its readers. The readers' ratings of books are range from 0 to 10. A score of 0 indicates that the reader did not give a score after reading. The higher the score, the more the users like the corresponding books. In order to facilitate processing and experiment, we mapped 1–10 points to 1–5 points for the Book-Crossing dataset. The statistics of the two datasets are given in Table 1.

Table 1. Statistics of the datasets.

Datasets	MovieLens-1M	Book-Crossing
# of Users	6040	278,858
# of Items	3706	271,379
# of Ratings	1,000,209	1,031,175
Rating Density	95.53	99.99

3.2. Experimental Settings

In the experiment, to ensure the quality of the dataset, we retain users and items with at least five interactions. We set the embedding size of the model to 16, 32, 64, 128, and 256 for testing, respectively. The learning rate is set to 0.001, the number of iterations is 100, and the activation function is RuLU. We employed three hidden layers for all the neural components. The early stopping strategy was performed, where we stopped training if the RMSE on the validation set increased for five successive epochs. We randomly initialized model parameters with a Gaussian distribution for all neural network methods, where the mean and standard deviation are 0 and 0.1, respectively. We randomly used 80% of the data as training data to learn model parameters, used the remaining 10% of the data as validation data to tune super parameters, and used the other 10% of the data as test data for the final performance test. We optimized all models with the Adam optimizer, where the batch size was fixed at 512. Additionally, we employed the node-dropout technique for GC-MC, NGCF, GraphRec, and our proposed LNNSR, where the dropout ratio was tuned in {0.3, 0.1,..., 0.8}.

3.3. Evaluation Metrics

In order to evaluate the quality of the recommendation algorithms, two popular metrics were adopted to evaluate the predictive accuracy, namely, Mean Absolute Error (*MAE*) and Root Mean Square Error (RMSE). Smaller values of *MAE* and RMSE indicate better predictive accuracy. Moreover, to evaluate the effectiveness of the top-K recommendation and preference ranking, we adopted the widely-used evaluation protocols *MRR@K*, *RECALL@K*, and *HR@K*. By default, we set $K = 10$. We reported the average metrics for all users in the test set. The calculation method of RMSE and MAE are shown as the following formulas:

$$MAE = \frac{\sum_{u,i \epsilon T} \left| R'_{u,i} - R_{u,i} \right|}{|T|} \tag{39}$$

$$MASE = \sqrt{\frac{\sum_{u,i \epsilon T} \left(R'_{u,i} - R_{u,i} \right)^2}{|T|}} \tag{40}$$

where T is the test data set, $R_{u,i}$ is the real score of user u on item I, and $R'_{u,i}$ is the score predicted by the recommendation algorithm.

The calculation method of *MRR@K*, *RECALL@K*, and *HR@K* are shown as the following formulas:

$$MRR@K = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{41}$$

where $|Q|$ is the total number of users. $rank_i$ is the ranking position of the first item in the real result in the recommendation list for the ith user;

$$RECALL@K = \frac{TP}{TP+FP} \tag{42}$$

where TP is the True Positives and FP is the False Positives;

$$HR@K = \frac{Nemberof Hits@K}{|GT|} \tag{43}$$

where $|GT|$ is all the test data set. *NemberofHits@K* is the sum of the number of test sets in the first k of each user.

3.4. Baselines

To demonstrate the effectiveness, we compared our proposed LNNSR with the following traditional recommender systems. In order to compare the accuracy of the score prediction of the model, we used the MSNR and MAE calculators in the first experiment. To evaluate the effectiveness of the top-K recommendations and preference ranking, we used the *MRR@10*, *RECALL@10*, and *HR@10* calculators in another experiment.

PMF [3]: Probabilistic Matrix Factorization is a classical factorization model of probability matrices that utilizes a user–item rating matrix only and models latent factors of users and items by Gaussian distributions. The model is constructed from the aspect of probability interpretation.

DCF [9]: This method is a hybrid recommendation algorithm combining probability matrix decomposition and a noise-reduction self-encoder. It is an early method for integrating the matrix decomposition model and the features of auxiliary information extracted by deep learning.

NeuMF [18]: This method is a state-of-the-art matrix factorization model with a neural network architecture. The original implementation is for a recommendation ranking task. We also calculated Mean Absolute Error and Root Mean Square Error for rating prediction.

IGMC [19]: This method is an inductive matrix completion model without using side information, which trains a graph neural network (GNN) based purely on 1-hop subgraphs around (user, item) pairs generated from the rating matrix and maps these subgraphs to their corresponding ratings.

GCMC [20]: Graph Convolutional Matrix Completion considers the matrix completion of the recommendation system from the perspective of link prediction on bipartite graphs. To solve the link prediction problem on bipartite graphs, a self-coding framework is used for graph information aggregation. At the same time, edge information is aggregated in information transmission.

NGCF [21]: Neural Graph Collaborative Filtering (NGCF) can capture the collaborative signal that is latent in user–item interactions, which exploits the user–item graph structure by propagating embeddings on it. This leads to the expressive modeling of high-order connectivity in the user–item graph, effectively injecting the collaborative signal into the embedding process in an explicit manner.

GraphRec [22]: GraphRec is a recommender system based on a graph neural network, which captures user and item feature representation vectors from a user–user social graph and a user–item graph; it is also based on a matrix-decomposition architecture. When learning latent factors of users and items, it aggregates the information of user–

item interaction graphs and social-network graphs through a deep neural network and attention mechanism.

F-EAE [31]: F-EAE uses exchangeable matrix layers to perform inductive matrix completion without using content. It uses a deep neural network to learn the interaction between two or more groups of objects.

PinSage [32]: PinSage is an inductive node-level GNN-based model using content, which was originally used to predict related pins and is here adapted to predicting ratings.

3.5. Experimental Results

In order to demonstrate the effectiveness of the proposed framework, LNNSR, we make an experimental comparison with the representative baseline model on the above two real data sets. Table 2 reports the performance comparison results, which are the average of 10 experiments.

Table 2. Performance comparison of different recommender systems.

Dataset	**MovieLens-1M**		**Book-Crossing**	
Metrics	**RMSE**	**MAE**	**RMSE**	**MAE**
PMF	0.9112	0.7988	1.4322	1.1616
DCF	0.8568	0.7446	1.2572	1.0215
GCMC	0.8371	0.6528	1.0941	0.9093
GraphRec	0.8604	0.7815	1.1307	0.9453
F-EAE	0.8409	0.6624	1.1032	0.9285
IGMC	0.8351	0.6587	1.0767	0.9073
LNNSR	0.7974	0.6519	1.0032	0.8504

From the above experimental results, it can be proved that our method consistently outperforms the other methods. DCF obtains much better performance than PMF, which is only based on matrix factorization, and shows the effectiveness of using deep learning to extract auxiliary information. F-EAE performs better than these models based on matrix decomposition, which suggests the power of neural network models in recommender systems. GCMC shows better performance, which supports that the GNNs are powerful in representation-learning for graph data. The GraphRec model takes advantage of both rating and social network information. When the user's social network information cannot be obtained, or for cold-start users, its performance may not be the best. Our method, LNNSR, consistently outperforms all the baseline methods. Compared to IGMC and GCMC, our model, LNNSR, made a new attempt to integrate the rating and latent factors of the user's attribute side and the item's attribute side. Moreover, our model considers different granularity distribution proportion weights for each attribute.

To evaluate the effectiveness of top-K recommendation and preference ranking, we selected some representative models of the baselines for comparison. Each method outputs the user's preference scores over all the items. For MovieLens-1M, we treat it as positive implicit feedback when the user's score for the item is greater than 3. For Book-Crossing, we treat it as positive implicit feedback when the user's score for the item is greater than 0, and we randomly select the same number of items as the negative implicit feedback for the user from those items where the user's score is 0. In addition, we filter out the users who do not have any positive implicit feedback. The average metrics of 10 random experiments is shown in Table 3.

From the above experimental results, we can find that our method outperforms almost the most baseline methods on MovieLens-1M. Especially for Book-Crossing, our method performs even better than NGCF. This may be because that there are some valuable attribute information on Book-Crossing for each user and item, but the interactions of users and items are relatively few. Compared with these baselines, we use a granular neural network to learn the optimal allocation for information granularity and integrate the proportional weight of the information granularity into the user- and item-implicit

factor representation, it can capture user-embedding representations and item-embedding representations more accurately. In addition, we consider interactions and opinions in the user–item graph, and we also consider the importance of different attribute information in the user–attribute graph and item–attribute graph. At present, our method utilizes the first-order neighbors to guide the representation learning; we will explore high-order connectivity in future work. We show that our proposed method has highly competitive performance compared to the state-of-the-art baselines. We hope LNNSR can provide a new idea to recommender systems.

Table 3. Performance comparison of top-10 recommendations.

Dataset	**MovieLens-1M**			**Book-Crossing**		
Metrics	***MRR@10***	***RECALL@10***	***HR@10***	***MRR@10***	***RECALL@10***	***HR@10***
PMF	0.3235	0.1535	0.5916	0.2235	0.1135	0.4316
NeuMF	0.4246	0.1794	0.7031	0.2835	0.1335	0.4671
GCMC	0.3784	0.1605	0.6063	0.3274	0.1485	0.5017
PinSage	0.3741	0.1625	0.5987	0.3111	0.1342	0.4827
NGCF	0.4563	0.1935	0.7376	0.3502	0.1531	0.5429
LNNSR	0.4502	0.1812	0.7035	0.3513	0.1483	0.5478

3.6. Effect of Embedding Size

For the neural network recommendation model, the embedding dimension is one of the important super parameters. We compare the performance of the model under different embedding dimension sizes. All parameters here will be consistent with the previous text. The RMSE results of models with different embedding dimension sizes under the MovieLens-1M dataset are shown in Figure 4. The experimental results on the Book-Crossing dataset are similar and are not shown here. It can be seen that with the increase of embedding size, the training effect of the model gradually becomes better, and the best embedding size is 64. With the increase of embedding dimension, the more detailed the representation of implicit preference features of users and items, the more the corresponding model accuracy is improved. Formally speaking, embedding is to "represent" an object with a low-dimensional dense vector. In the field of recommender systems, users' ratings of items are mostly a very sparse matrix. The high-dimensional sparse feature vector is not suitable for the training of multilayer complex neural networks. When using the deep-learning model to process the high-dimensional sparse feature vector, an embedded layer is added between the input layer and the full connection layer to convert the high-dimensional sparse feature vector to the low-dimensional dense feature vector to improve the performance of the model. However, larger embedding sizes are not better. It can be seen that when the embedding size is 128, the RMSE increases, compared with 64. This shows that although the embedding size gradually increases, it has a better representation ability but will bring overfitting and other problems, reducing the generalization ability of the model. On the other hand, if the dimension of the input vector in the embedding layer is too large, it will lead to a huge number of parameters in the whole embedding layer. This will slow down the convergence speed of the whole neural network. Therefore, we have to find a proper length of embedding. In our experiment, we use the embedding size of 64 to balance the relationship between performance and complexity.

3.7. Effect of User's Attribute Information and Item's Attribute Information

To verify the effectiveness of the proposed model framework, we generate a variant model named LNNSR-U by removing the latent factor of the user's attribute side, and generate another variant model named LNNSR-I by removing the latent factor of the item's attribute side. Experiments were carried out in the above two datasets, and the results are as following, in Figures 5 and 6.

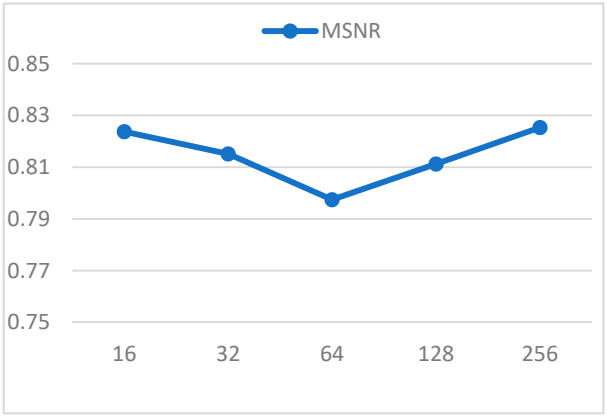

Figure 4. Effect of embedding size on MovieLens-1M datasets.

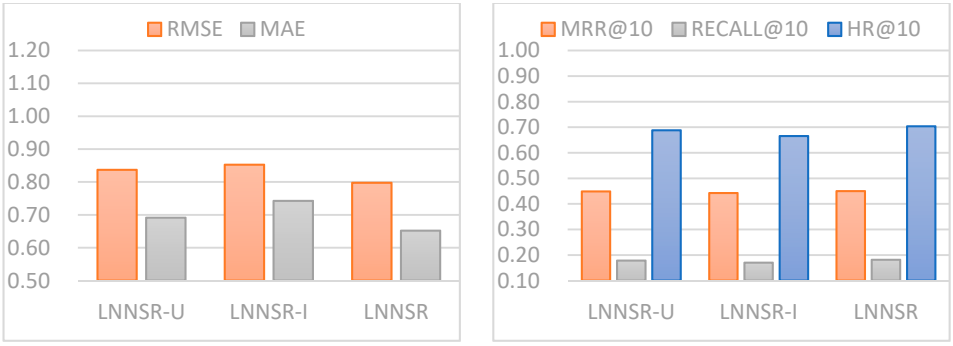

Figure 5. Effect of user's attribute information and item's attribute information on the MovieLens-1M dataset.

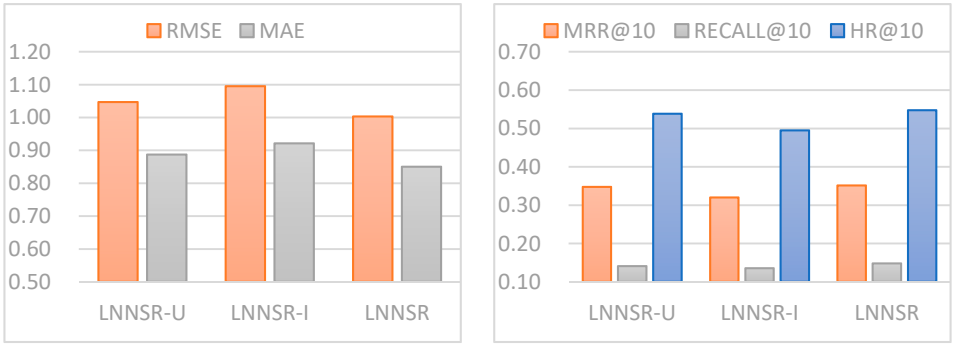

Figure 6. Effect of user's attribute information and item's attribute information on the Book-Crossing dataset.

It can be seen from the experimental results that both LNNSR-U and LNNSR-I lead to the degradation of model performance. It is proved that the user's attribute and the item's attribute play a certain role in characterizing the user latent factor vector and the item latent factor vector. Especially in the experimental results of the Book-Crossing dataset, its performance will decline more when removing the latent factor of the item's attribute. That may be due to the greater role of attribute information in the item latent factor vector on the Book-Crossing dataset.

3.8. Effect of Granularity Distribution Proportion Weights on Node's Attributes

In this paper, the different granularity distribution proportion weights of each attribute of nodes can be calculated by a granularity neural network. The optimized levels of information granularity allocated on each attribute can be integrated into the latent factor

representation of users and items. In order to illustrate the effectiveness of this strategy, we compare the performance of the model with combining attribute granularity allocation with proportional weight with that of the model without combining attribute granularity allocation with proportional weight (named as LNNSR-N). For the LNNSR-N, we evenly allocate the granularity weight of each attribute of the node. The experimental results are shown in Figures 7 and 8.

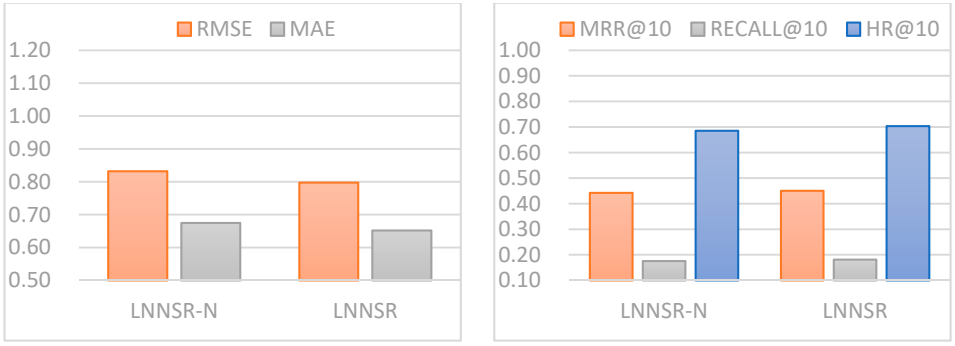

Figure 7. Effect of granularity distribution proportion weights of node's attributes on MovieLens-1M dataset.

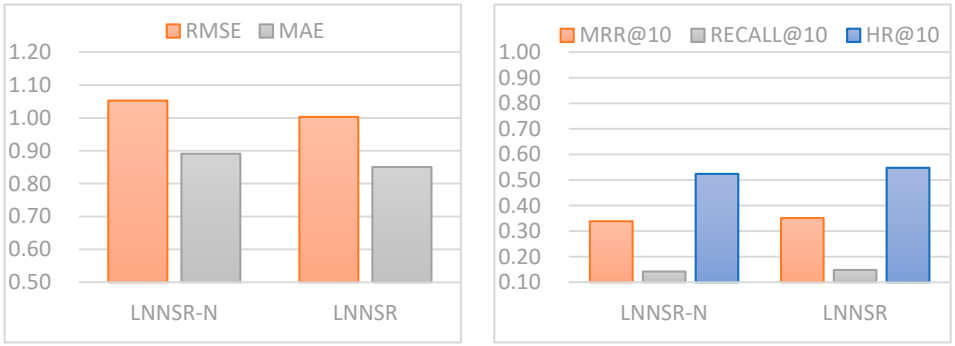

Figure 8. Effect of granularity distribution proportion weights of node's attributes on the Book-Crossing dataset.

From the results, we can find that not all attributes of one user contribute equally to the latent factor of the user's attribute side. Not all attributes of one item have the same importance for learning the latent factor of the item's attribute side. Different attributes have different importance. We obtain the level of information granularity allocated on each attribute by constructing interval granular neural networks. As shown in Formula (8), we can use an improved Partheno genetic algorithm to continuously optimize the granularity allocation $(\varepsilon_1, \varepsilon_2 \ldots \varepsilon_n)$ until the Q is optimal. We obtain the level of information granularity according to the "specificity" $Q1$ and the "coverage" Q_{cover} of the target output. Through the above experimental results, it is proved that our idea is feasible. By aggregating user attributes to represent the user latent factor vector and aggregating item attributes to represent the item latent factor vector, we can more accurately capture user preferences and item characteristics.

4. Conclusions and Future Work

In this paper, we construct a user–item graph, a user–attribute graph, and an item–attribute graph according to the interaction information between users and items. A graph neural network carries out the depth representation of the user and item node information. In order to capture the latent factor-embedding representation of nodes more accurately, we have presented a rating prediction recommendation model (LNNSR), which combines with the optimizing allocation for information granularity of the node attribute by developing a

granular neural network. The granularity neural network is used to calculate the proportion weight of each attribute of each node. The proportion weight of attribute granularity is fused into the representation of user and item latent factors to capture the user- and item-embedding representation accurately. Experimental results on two real-world datasets demonstrate the effectiveness of the proposed framework, LNNSR.

However, our model also has some limitations. First of all, for the implicit factor representation of users and items, we use the interaction and scoring behavior, user attribute information, and item attribute information to capture the implicit factor vector of nodes without considering the dynamic propagation of the implicit factor vector. In fact, the interaction and scoring behavior of users and items change and propagate dynamically in the whole graph network. In the future, we will try to add its dynamic and communicative feature representation. Secondly, the granular neural network combined with interval computation and neural network used in this paper may be deficient in data generalization ability. There is still a lot of room for improvement. For example, in this paper, we use the method of interval mathematics to granulate the input space of attributes. We extend the numerical data to an interval, which has some limitations. We use an improved Partheno genetic algorithm to optimize the granularity allocation of input space; the optimization effect for other types of tasks and huge capacity level input space is not very good. In the experimental data set, we only optimize the granularity allocation of a small-scale input space. However, the optimization effect is not very good for large-scale input spaces. Therefore, we need to explore more appropriate and universal population optimization algorithms. Thirdly, in the experimental results of the top-K recommendations, our method did not outperform all the baseline methods. This might be because our method utilizes the first-order neighbors to guide the representation learning; we will explore the high-order connectivity. Moreover, we only carried out experiments in single-mode data; we will extend this model to the multi-modal data in future work.

Author Contributions: Investigation, J.L.; methodology, J.L. and Y.W.; experiment, J.L.; supervision, Y.W. and Z.T.; validation, J.L. and Y.W.; writing—original draft, J.L.; writing—review and editing, Y.W. and Z.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Acknowledgments: This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1406201 and Grant 2019YFB1406204.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
2. Wang, X.; Zhang, R.; Sun, Y.; Qi, J. Combating selection biases in recommender systems with a few unbiased ratings. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM), Online, 8–12 March 2021; pp. 427–435.
3. Salakhutdinov, R.; Mnih, A. Probabilistic matrix factorization. *Adv. Neural Inf. Processing Syst.* **2007**, *20*, 1257–1264.
4. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the 4th ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; ACM: New York, NY, USA, 2010; pp. 135–142.
5. Zhou, X.; He, J.; Huang, G.; Zhang, Y. SVD-based incremental approaches for recommender systems. *J. Comput. Syst. Sci.* **2015**, *81*, 717–733. [CrossRef]
6. Guibing, G.; Jie, Z.; Neil, Y.-S. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; AAAI Press: Palo Alto, CA, USA, 2015; pp. 123–129.

7. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, ON, Canada, 18–21 June 2009; AUAI Press: Arlington, VA, USA, 2009; pp. 452–461.
8. Rendle, S. Factorization machines. In Proceedings of the 10th IEEE International Conference on Data Mining, Bradford, UK, 29 June–1 July 2010; IEEE: New York, NY, USA, 2010; pp. 995–1000.
9. Li, S.; Kawale, J.; Fu, Y. Deep collaborative filtering via marginalized denoising auto-encoder. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, VIC, Australia, 18–23 October 2015; pp. 811–820.
10. Barkan, O.; Koenigstein, N. Item2vec: Neural item embedding for collaborative filtering. In Proceedings of the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), Vietri sul Mare, Italy, 13–16 September 2016.
11. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. Deepfm: A factorization-machine based neural network for ctr prediction. *arXiv* **2017**, arXiv:1703.04247.
12. He, X.; Chua, T.-S. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017.
13. Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; Chua, T.-S. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv* **2017**, arXiv:1708.04617.
14. Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; VanderGheynst, P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* **2017**, *34*, 18–42. [CrossRef]
15. Derr, T.; Ma, Y.; Tang, J. Signed graph convolutional networks. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; IEEE: New York, NY, USA, 2018; pp. 929–934.
16. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
17. Zhao, Y.; Qi, J.; Liu, Q.; Zhang, R. WGCN: Graph Convolutional Networks with Weighted Structural Features. *arXiv* **2021**, arXiv:2104.14060.
18. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.-S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, WA, Australia, 3–7 April 2017; pp. 173–182.
19. Zhang, M.; Chen, Y. Inductive matrix completion based on graph neural networks. *arXiv* **2019**, arXiv:1904.12058.
20. Van den Berg, T.; Kipf, T.N.; Welling, M. Graph convolutional matrix completion. *arXiv* **2017**, arXiv:1706.02263.
21. Wang, X.; He, X.; Wang, M.; Nie, L.; He, X.; Hong, R.; Chua, T.S. Neural graph collaborative filtering. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 165–174.
22. Fan, W.; Ma, Y.; Li, Q.; Wang, J.; Cai, G.; Tang, J.; Yin, D. Graph neural networks for social recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; ACM: New York, NY, USA, 2019; pp. 417–426.
23. Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; Tang, J. Autoint: Automatic feature interaction learning via self-attentive neural networks. In Proceedings of the 28th International Conference on Information and Knowledge Management (CIKM), Beijing, China, 3–7 November 2019; pp. 1161–1170.
24. Su, Y.; Erfani, S.M.; Zhang, R. MMF: Attribute interpretable collaborative filtering. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; IEEE: New York, NY, USA, 2019; pp. 1–8.
25. Chen, S.; Wu, M. Attention Collaborative Autoencoder for Explicit Recommender Systems. *Electronics* **2020**, *9*, 1716. [CrossRef]
26. Sánchez, D.; Melin, P.; Castillo, O. Optimization of modular granular neural networks using a firefly algorithm for human recognition. *Eng. Appl. Artif. Intell.* **2017**, *64*, 172–186. [CrossRef]
27. Kumar, D.A.; Meher, S.K.; Kumari, K.P. Knowledge-Based Progressive Granular Neural Networks for Remote Sensing Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2017**, *10*, 5201–5212. [CrossRef]
28. Song, M.; Jing, Y.; Pedrycz, W. Networks, A study of optimizing allocation of information granularity in input space. *Appl. Soft Comput.* **2019**, *77*, 67–75. [CrossRef]
29. Song, M.; Jing, Y. Networks, The development of granular input spaces and parameters spaces through a hierarchical allocation of information granularity. *Inf. Sci.* **2020**, *517*, 148–166. [CrossRef]
30. Chen, C.; Zhang, M.; Liu, Y.; Ma, S. Neural attentional rating regression with review-level explanations. In Proceedings of the 27th International Conference on World Wide Web, Lyon, France, 23–27 April 2018; pp. 1583–1592.
31. Hartford, J.; Graham, D.R.; Leyton-Brown, K.; Ravanbakhsh, S. Deep models of interactions across sets. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; p. 10.
32. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; ACM: New York, NY, USA, 2018; pp. 974–983.