electronics

MDPI

Article

Attribute-Aware Graph Convolutional Network Recommendation Method

Ning Wei [1,2], Yunfei Li [1,2], Jiashuo Dong [1,2], Xiao Chen [3] and Jingfeng Guo [1,2,*]

[1] College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China; weining1121@stumail.ysu.edu.cn (N.W.); liyunfei@stumail.ysu.edu.cn (Y.L.); boliu408@stumail.ysu.edu.cn (J.D.)
[2] Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao 066004, China
[3] Research Center for Marine Science, Hebei Normal University of Science and Technology, Qinhuangdao 066004, China; chenxiao0604@hevttc.edu.cn
* Correspondence: jfguo@ysu.edu.cn

Abstract: In recent years, recommendation systems have made significant strides through the application of graph neural networks (GNNs). However, most of the existing methods primarily focus on modeling user–item interactions, often failing to account for the distinct roles of different types of neighboring nodes during the graph convolution process. Moreover, the intricate relationships between user and item attributes are frequently underexplored, which constrains further improvement in the performance of recommendation models. To overcome these challenges, this paper proposes an attribute-aware graph convolutional network recommendation model (AAGCNR). This model accounts for the complex interrelationships between user and item attributes while distinguishing between different types of neighboring nodes during graph convolution. First, a multi-head self-attention mechanism is applied to capture the semantic relationships among attributes across various semantic spaces. Additionally, a bilinear interaction module is employed to facilitate interactions between attributes. Since different neighboring nodes exert different influences on target nodes, the model performs convolutional aggregation by leveraging the interrelationships of these attributes throughout the graph convolution process. Experiments conducted on real-world datasets reveal that AAGCNR outperforms other benchmark algorithms, particularly in terms of RECALL and NDCG metrics.

Keywords: recommendation algorithm; graph neural network; attention mechanism; attribute information; interaction features

check for updates

Citation: Wei, N.; Li, Y.; Dong, J.; Chen, X.; Guo, J. Attribute-Aware Graph Convolutional Network Recommendation Method. *Electronics* **2024**, *13*, 4267. https://doi.org/10.3390/electronics13214267

Academic Editor: Domenico Rosaci

Received: 9 October 2024
Revised: 22 October 2024
Accepted: 29 October 2024
Published: 30 October 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

Traditional recommendation algorithms based on graph neural networks (GNNs) primarily learn user and item embedding representations from user–item interaction graphs [1].

The core principle involves utilizing the graph's topological structure to capture similar features of higher-order neighbors. However, user–item interaction data is often sparse, leading to inaccurate embedding representations for both users and items. To address this issue, research has introduced auxiliary information, such as user or item attributes, into recommendation models to alleviate the adverse effects of sparse interaction data [2,3]. Most recommendation studies incorporate either user attributes or item attributes as the sole auxiliary information for modeling. These attributes act as bridging elements, enabling the capture of more similar features within the graph. Some studies have employed both user and item attributes for modeling [4], yet these models generally apply linear weighting to the attribute nodes, resulting in composite attribute representations. This simplistic approach assumes that attribute nodes function independently, thereby overlooking the

interactions between attributes. As a result, the intricate relationships between user and item attributes are insufficiently explored, which ultimately hinders improvements in recommendation performance.

In graph-based recommendation models, researchers have increasingly introduced attributes as entity nodes to address the challenge of data sparsity. However, some studies [2,5] have neglected to distinguish between the types of neighboring nodes during the graph convolution process, overlooking the unique influences that different types of neighboring nodes can have on the target node. While other studies [4,6] have considered the interactions between attribute features, they often allow user–item attribute nodes to directly interact with user–item nodes during graph convolution. This approach, however, fails to account for the fact that different node types reside in distinct spatial feature domains. As a result, calculating node similarity in this manner can lead to inaccurate representations of user preferences. These inaccuracies, in turn, introduce biases into user preference modeling, which ultimately degrade the performance of the recommendation method.

To address the issues mentioned above, this paper proposes an attribute-aware graph convolutional recommendation method (AAGCNR). The model consists of several layers: an input layer, an attribute feature interaction layer, a user attribute preference mining layer, a user attribute preference fusion layer, an attribute co-convolution layer, and a prediction layer. First, the model applies separate embedding operations to the user attribute graph, the item attribute graph, and the user–item interaction graph to obtain the initial embedding representations for users, items, user attribute nodes, and item attribute nodes. Second, a multi-head self-attention mechanism is employed to capture complex semantic information from the user and item attribute features, considering their interactions. This mechanism allows the model to process the attribute information in both the user attribute space and the item attribute space. Linear and nonlinear interaction features between these attributes are then fused, resulting in user-fused attribute embedding representations and item-fused attribute embedding representations, which enhance the embedding quality by capturing fine-grained, complex relationships between attributes. Next, to differentiate between the types of neighboring nodes during the graph convolution process better, the model employs another self-attention mechanism. This mechanism models the influences of both group preferences and personalized preferences on users, thereby improving the accuracy of user–item interaction behavior during the convolution. By incorporating user preference features directly into the graph convolution process, the model enhances the embedding representations of both users and items. Finally, score prediction is carried out through an inner product operation based on the refined embeddings.

The primary contributions of this paper are as follows:

1. To address the feature interaction problem between attribute nodes in the graph, we first introduce user attribute features and item attribute features into the graph. We then model the interaction between these user and item attributes using a multi-head self-attention mechanism and a bilinear interaction module.

2. To address the impact of neighborhood node types on target nodes, we enhance the modeling of user preference features for item attributes. This approach guides the interaction between user and item nodes during the graph convolution process based on the users' preferences for item attributes.

2. Related Work

2.1. Graph Neural Networks

Graph convolutional neural networks (GCNNs) are a subset of graph neural networks (GNNs). Their key principle is to aggregate features from multi-level neighborhoods through iterative convolutional layers [7]. Graph convolutional networks (GCNs) were initially developed in the spectral domain. By randomly sampling adjacent nodes and using an aggregation function to combine their features, new node embeddings are created [8]. Random walk sampling of neighboring nodes and the graph's data structure are used to

generate node embeddings that encode both the graph structure and the neighborhood features, reducing the computational complexity of the model [9]. Knowledge graphs have been integrated into recommendation systems and combined with graph convolutional networks (GCNs) [5,10]. Feature transformation and nonlinear activation functions were removed from GCNs, simplifying the model for learning user and item embeddings [11]. A convergence method was designed for the loss function to address issues from multi-layer stacking, improving the recommendation efficiency [12]. Using the user–item interaction graph and the message-passing concept, a graph convolutional network (GCN) captured the higher-order relationships between nodes, learning the user and item representations [1]. Graph Attention Networks (GATs), another type of graph neural network, employ an attention mechanism to learn node representations. Additionally, fine-grained user intent modeling was applied to the graph [13].

2.2. Recommendation by Integrating Attribute and Behavioral Features

The sparsity of user–item interaction behavior features and user–item attribute features makes joint modeling of the two tasks mutually reinforcing. Initial efforts have been made to explore the correlation between these two tasks via joint modeling [14,15]. These models redefine the available data as attribute graphs and typically rely on classical graph-based methods, such as label propagation and link prediction, to predict the outcomes for both tasks. Optimization is performed through a joint loss function that integrates both tasks. Joint models have shown better performance compared to separate modeling. However, their reliance on classical shallow semi-supervised learning models leads to suboptimal performance. A novel variant of graph neural network (GNN) called the Gated GNN has been proposed to effectively aggregate various types of attribute nodes within a neighborhood [16]. Furthermore, user and item attribute information has been incorporated into recommendations and combined with user–item interaction data [4]. Higher-order convolutions on attribute graphs have been used to improve the user–item embeddings [6]. A fine-grained preference fusion strategy has been proposed to integrate attribute group preferences with individual behavior preferences, helping mitigate the data sparsity issue by leveraging attribute information [17].

2.3. Research on Missing Attribute Features

For most attribute-enhanced recommendation models, a key assumption is that the attribute values are complete. However, annotating user or item attributes is labor-intensive, and many attribute values are frequently incomplete. As a result, obtaining a complete set of attributes is costly, and many datasets suffer from missing attribute values. Research has addressed the problem of missing attributes—for instance, integrating attribute representation inference with graph convolution-based recommendations. Estimated attribute values are used to adjust the graph embedding learning parameters. These learned parameters and user–item embeddings are then fed into an attribute update module, which iteratively infers the missing attribute values, ultimately improving both the attribute inference and recommendation performance [18]. Similarly, an attribute-aware attention-based graph convolutional neural network has been proposed. This network uses graph convolutional networks and applies the message-passing paradigm to capture the relationships among users, items, and attributes, effectively addressing the issue of missing attributes [19].

3. Methods

This paper introduces the attribute-aware graph convolutional network recommendation method (AAGCNR) in detail, with the model framework illustrated in Figure 1. The model is structured with several key components: an input layer, an attribute feature interaction layer, a user attribute preference mining layer, a user attribute preference fusion layer, an attribute collaborative convolution layer, and a prediction layer. First, embedding operations are applied separately to the user attribute graph, the item attribute graph, and the user–item interaction graph to generate the initial embeddings of user nodes, item

nodes, user attribute nodes, and item attribute nodes. Next, the interactions between user attribute features and item attribute features are considered. A multi-head self-attention mechanism is utilized to capture the complex semantic information within both the user attribute space and the item attribute space. The linear and nonlinear interaction features between these attributes are then fused, resulting in user-fused attribute embeddings and item-fused attribute embeddings. By capturing the intricate relationships between attributes at a fine-grained level, the embeddings of users and items are refined further. Finally, a self-attention mechanism is employed to model the influence of group preferences and personalized preferences on users. This mechanism enhances the graph convolution process by incorporating user preference features, improving the accuracy of user–item interaction predictions. The final step involves score prediction, which is carried out through an inner product operation based on the optimized user and item embeddings.

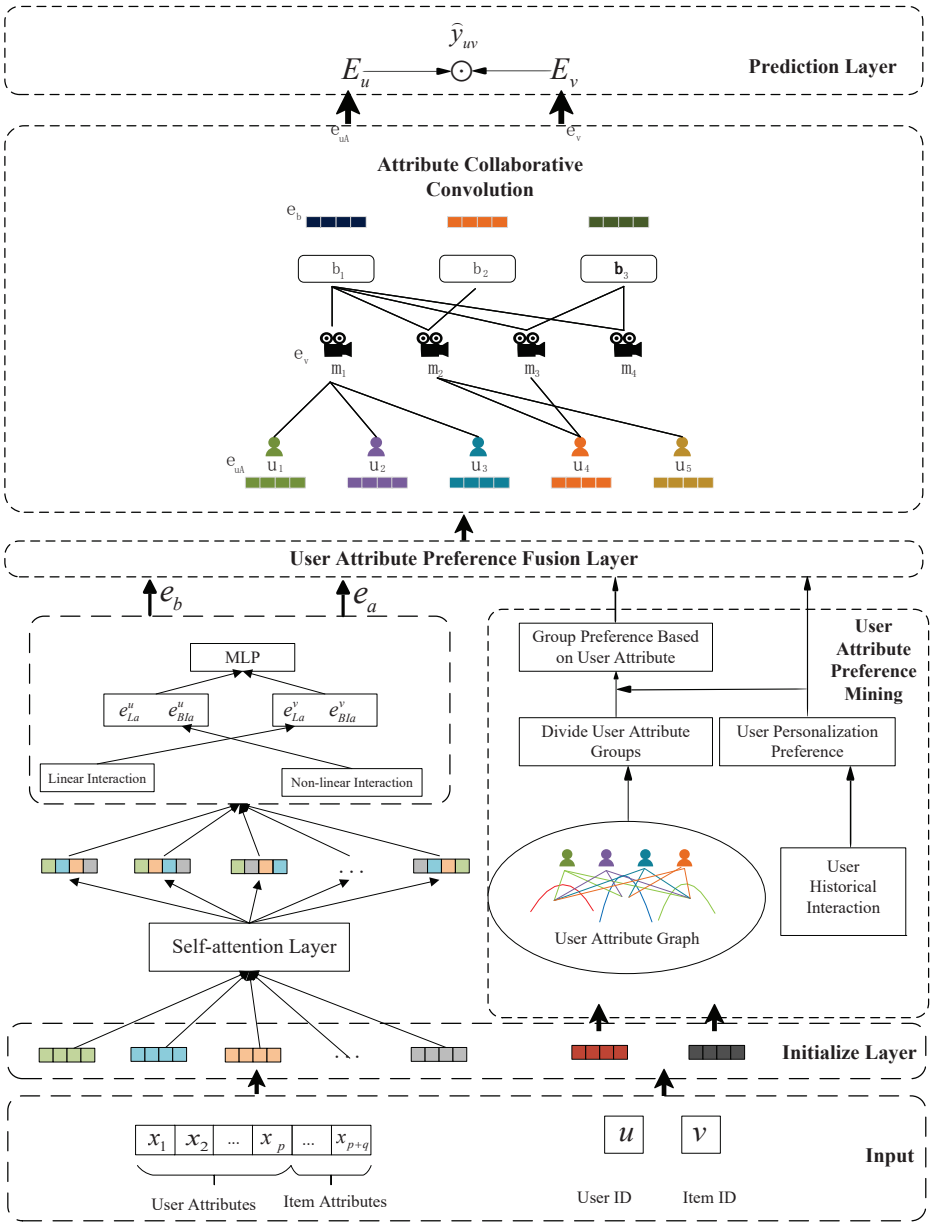


Figure 1. AAGCNR model.

3.1. Node Embedding and Feature Mapping

After preprocessing, the input variables are fed into the initialization mapping layer to obtain embedded representations for each variable. The dataset involved in the model includes variables such as user ID u, item ID v, user attributes x_i, and item attributes x_j. These variables undergo numerical encoding starting from zero, incrementally, following preprocessing. The encoded variables are then used as inputs to initialize the mapping layer, which aims to derive initial vector representations of the variables in the latent space based on the row vectors of the initial embedding matrix corresponding to their encoded values.

In the context of user u, the fully connected layer matrices for item v, user attributes x_i, and item attributes x_j are denoted as $u \in U^{m\times d}$, $v \in V^{n\times d}$, and $x_i \in X^{p\times h}$, respectively. These initialization matrices have their variables initialized based on random sampling from a normal distribution with a given mean and standard deviation. Here, m represents the number of users, n represents the number of items, p denotes the number of user attribute values, and q denotes the number of item attribute values. d signifies the dimensionality of the latent space vectors for users and items, while d_a denotes the dimensionality of the latent space vectors for user attributes and item attributes. After encoding the variables input into the initialization mapping layer, u_i corresponds to the i-th row $U^{i\times d}$ of matrix U, j corresponds to the j-th row $V^{j\times d}$ of matrix V, and x_k corresponds to the k-th row $X^{k\times h}$ of matrix X.

3.2. The Attribute Feature Interaction Layer

3.2.1. The Self-Attention Mechanism Layer

After the input variables are processed by the initialization mapping layer in the previous section, the AAGCN model employs a multi-head self-attention mechanism in the self-attention layer. This approach is taken because user attributes and item attributes do not reside in the same semantic space. Consequently, the multi-head self-attention mechanism captures the semantic features between user and item attributes across different semantic spaces, thereby enriching the semantic representation of both user and item attributes.

Each user $u \in U$ has a set of user attributes $x^u = \{x_1^u, x_2^u, x_3^u, \ldots, x_p^u\}$ as input, with p representing the number of user attributes. Similarly, each item $v \in V$ has a set of item attributes $x^v = \{x_1^v, x_2^v, x_3^v, \ldots, x_q^v\}$ as input, with q representing the number of item attributes. Each attribute is represented as a latent vector of d_a dimensions. To capture the correlation coefficients between these attributes, the initialized user attribute vectors x^u and the initialized item attribute vectors x^v are used as inputs $x = \{x^u, x^v\}$ for the multi-head self-attention mechanism. The attribute initialization embedding vectors x with d_a dimensions are transformed into embedding representations of $d_k = d_a/h$ dimensions, where h denotes the number of attention heads.

To obtain the correlation weight coefficient α_{ij}^h between the i-th attribute vector and the j-th attribute vector under the h-th attention head, the following formula is used: $W_Q^h, W_K^h \in R^{d'\times d}$.

$$e_{ij} = \frac{(x_i W_Q^h)(x_j W_K^h)}{\sqrt{d_k}} \tag{1}$$

$$\alpha_{ij}^h = \frac{\exp\ e_{ij}}{\sum_{k=1}^{p+q} \exp\ e_{ik}} \tag{2}$$

Next, the representation $\tilde{x}_i^h$ under the h-th attention head is obtained through a weighted linear combination of vectors, as shown below, where $W_V^h \in R^{d'\times d}$ is the weight matrix under the h-th attention head:

$$\tilde{x}_i^h = \sum \alpha_{ij}^h (x_j W_V^h) \tag{3}$$

Thus, different interactions between attribute vectors x_i and x_j are constructed under various attention heads h. $\tilde{x}_i^h$ under different attention heads is spliced to obtain $\tilde{x}_i$.

$$\tilde{x}_i = Concat(\tilde{x}_i^1, \tilde{x}_i^2, \tilde{x}_i^3, \ldots, \tilde{x}_i^h) \tag{4}$$

Here, h represents the number of heads in the multi-head self-attention mechanism, and *Concat* refers to the vector concatenation operation. To prevent the loss of original information in the attribute vectors, a ReLU activation function is added, incorporating the original information x_i into the attention-concatenated vector $\bar{x}_i$, as shown in the formula below:

$$\bar{x}_i = max(\tilde{x}_i + x_i,\ 0) \tag{5}$$

The above-described multi-head self-attention mechanism integrates the correlations between user and item attributes into their embedding representations. This process enriches the embedding representations of both the user and item attributes, thereby enhancing their expressive power.

3.2.2. The Bilinear Interaction Layer

In the previous section, the multi-head self-attention mechanism was used to capture the semantic features between user attributes and item attributes, integrating the semantics into the attribute embedding representations. Inspired by the NFM [20], a feature cross-pooling function $f_{BI}(x)$ is designed to capture the second-order interactions between features, which is then fused with a first-order linear weighting function to capture the interaction relationships between features. After these two modules, a fully connected layer is added to model the complex nonlinear interactions between user attributes and item attributes further.

$$y_{NFM} = w_0 + \sum_{i=1}^{N} w_i x_i + f_{BI}(x) \tag{6}$$

Here, w_0 denotes the bias term, N represents the total number of features, w_i indicates the weight value of feature x_i, $f_{BI}(x)$ represents the feature cross-pooling function as shown below, and $\odot$ denotes the inner product operation.

$$f_{BI}(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (x_i v_i) \odot (x_j v_j) \tag{7}$$

Therefore, after obtaining the output from the multi-head self-attention layer in this model, a linear aggregation function $f_L(\bullet)$ is employed to derive the linear representations of the user attributes e_{La}^u and the item attributes e_{La}^v, respectively.

$$e_{La}^u = f_L(x) = \sum_{i=1}^{p} a_i \bar{x}_i \tag{8}$$

$$e_{La}^v = f_L(x) = \sum_{i}^{q} b_i \bar{x}_i \tag{9}$$

In this context, p denotes the number of attributes associated with user u, where a_i represents the attention vector of these user attributes and $\bar{x}_i$ indicates their respective weight values. Likewise, q denotes the number of attributes associated with item v, where b_i represents the attention vector of these item attributes and $\bar{x}_i$ indicates their respective weight values.

Firstly, the embedding representations of the interactions between pairs of user attributes e_{BIa}^u and pairs of item attributes e_{BIa}^v are obtained using an attribute interaction pooling function $f_{BI}(x)$, as shown in Equations (10) and (11).

$$e_{BIa}^u = f_{BI}(x) = \sum_{i=1}^{p} \sum_{j=i+1}^{p} a_i \bar{x}_i \odot a_j \bar{x}_j \tag{10}$$

$$e_{BIa}^{v} = f_{BI}(x) = \sum_{i=1}^{q} \sum_{j=i+1}^{q} a_i \bar{x}_i \odot a_j \bar{x}_j \tag{11}$$

In this context, $\odot$ denotes the operation of the inner product.

After obtaining the first-order linear representation e_{La}^{u} and the second-order nonlinear representation e_{BIa}^{u} of user attributes and the first-order linear representation e_{La}^{v} and the second-order nonlinear representation e_{BIa}^{v} of item attributes, the high-order interaction of the two linear representations of user attributes and the two linear representations of item attributes, respectively, is obtained through a fully connected network $f_{MLP}(x)$, and the higher-order nonlinear embedding representations E_a^u and E_a^v between the attributes are obtained.

$$E_a^u = f_{MLP}^{u}(x) = \text{LeakyReLU}(W_{mlp}^{1} f_{La}^{u}(x) + W_{mlp}^{2} f_{BIa}^{u}(x) + b_1) \tag{12}$$

$$E_a^v = f_{MLP}^{v}(x) = \text{LeakyReLU}(W_{mlp}^{1} f_{La}^{v}(x) + W_{mlp}^{2} f_{BIa}^{v}(x) + b_2) \tag{13}$$

W_{mlp}^{1}, W_{mlp}^{2} represents the parameter matrix of the fully connected layer, while b_1, b_2 denotes the bias constant.

By implementing the aforementioned methods, a reduction in the dimensionality of the embedding representations was achieved. Furthermore, the high-order relationships between user attributes and item attributes were thoroughly explored, resulting in more accurate and comprehensive user–item embeddings.

3.3. The User Attribute Preference Mining Layer

The preference of a user for an item cannot be comprehensively described solely based on personalized attribute preferences. This limitation is particularly evident when dealing with users with a limited interaction history. Furthermore, expressing user preferences exclusively through the preferences of the user group to which they belong lacks personalization. Based on the initialization of the user ID and item ID embeddings from the mapping layer, these embeddings are then input into the user preference mining module. This series of user preference modeling aims to enhance the precision of the interactions between nodes during the convolution process, resulting in more accurate user and item embedding representations. For example, if user u has three attributes, Age, Gender, and Occupation, then it is necessary to identify user groups with the same Age attribute U_{age}, the same Gender attribute U_{gender}, and the same Occupation attribute $U_{occupation}$ first. Subsequently, the personalized preferences for the item attributes within each user attribute group are statistically analyzed and normalized to obtain the user attribute group preference weights.

3.3.1. Modeling Users' Personalized Preferences

First, the frequency of attributes in the historical interaction items of user u is statistically analyzed and normalized to obtain the personalized preference weights of user u for the item attributes. Suppose user u has interacted with n items in the past, represented by $\{v_1, v_2, v_3, \ldots, v_n\}$. Each item $v_i (i \in n)$ corresponds to an original attribute vector $X = [x_1 x_2 x_3, \ldots, x_q]$ that represents all the attribute nodes of the item, where q is the number of item attributes. The personalized preference weights of users for item attributes can be calculated using Equations (14) and (15).

In the process of calculating weights, the method for calculating the attribute weights is divided into two categories based on the data type of the item attribute values: continuous, discrete, or boolean types.

First, the personalized preference weights ω_x^u of user u for boolean item attributes x_i are calculated using Equation (14), where n represents the number of historical interaction items v for user u, $y_{vx_i} = 1$ indicates that item v has attribute x_i, and $y_{vx_i} = 0$ indicates that item v does not have attribute x_i.

$$\omega_{x_i}^{u} = \frac{\sum y_{vx_i}}{n} \tag{14}$$

Second, because the continuous attribute (Release Year), the Actor attribute, and the Director attribute have multiple attribute values, Equation (15) is used to calculate the individualized preference of user u for the items v for continuous (Release Year) attributes and other (Director) attributes x_i and the j-th attribute value x_{ij} for project v. In this context, n represents the number of items v that user u has historically interacted with; $y_{vx_{ij}} = 1$ denotes items in v with attribute type x_i and the attribute value x_{ij}; and $y_{vx_{ij}} = 0$ indicates items in v that do not have attribute value x_{ij} under attribute type x_i.

$$\omega_{x_{ij}}^{u} = \frac{\sum y_{vx_{ij}}}{n} \tag{15}$$

3.3.2. Group Preferences Based on User Attributes

Firstly, a second-order neighborhood hop is performed on the user attribute graph G_{ua}, resulting in the related attribute groups $U_{attribute\,group}$ being obtained for each user attribute u, as illustrated below.

$$U_{attribute\ group} = f_{jump\ 2}(G_{ua}) \tag{16}$$

Based on the personalized attribute preference weights computed, the personalized preferences of users within each attribute group are statistically analyzed and normalized. This process yields the group attribute preference weights for each attribute group concerning the project attributes. The normalization methods for the attribute preference weights corresponding to different attribute value types are presented in Equations (17) and (18).

$$\omega_{x_i}^{U} = \frac{\sum \omega_{x_i}^{u}}{N} \tag{17}$$

Here, $\omega_{x_i}^{U}$ represents the normalized preference weight of user group U for item attribute x_i. N denotes the number of individuals in the attribute group. $\omega_{x_i}^{u}$ indicates the preference weight of users within group U for item attribute x_i.

$$\omega_{x_{ij}}^{U} = \frac{\sum \omega_{x_{ij}}^{u}}{N} \tag{18}$$

In this context, $\omega_{x_{ij}}^{U}$ represents the preference weight of the normalized user attribute group U towards the j-th attribute under item attribute x_i, while $\omega_{x_{ij}}^{u}$ signifies the preference weight of an individual user u within the user group U towards the j-th attribute under item attribute x_i.

3.4. The User Attribute Preference Fusion Layer

Based on the individualized user preferences and the user attribute group preferences derived from the multi-head self-attention-mechanism-based item attribute and user attribute preference mining module, these features are input into the user attribute preference fusion module to construct a fused preference embedding representation for user–item attribute interactions. This representation serves as the attribute embedding for both user nodes and item nodes in the attribute collaboration graph.

When the quantity of user historical interactions is substantial, personalized preferences offer a more comprehensive and accurate portrayal of user preferences, while group attribute preferences serve as a supplementary description of the user attributes. Conversely, when historical interactions are limited, group attribute preferences dominate in characterizing user preferences, with personalized attribute preferences providing complementary insights.

Based on the weights of the users' personalized preferences and the user attribute group preferences derived, weighted fusion is used in conjunction with the item attribute

embeddings outputted by the multi-head self-attention mechanism, ultimately yielding an embedded representation e_{uA} of the user attributes. This fusion of the two attribute preference vectors takes into account the volume of user historical interactions, enabling adaptive fusion of the attribute preference embeddings, as depicted in Equation (19). Here, E_{gp} represents the embedded representation of the user attribute group, E_p denotes the personalized embedded representation of user attributes, and the coefficient α signifies the influence of the embedded representation of the user attribute group preferences on the overall user attribute embedding.

$$e_{uA} = \alpha \times E_{gp} + (1 - \alpha) \times E_p \tag{19}$$

During the process of weighted fusion for the embedded representation of user attribute group preferences E_{gp}, two fusion methods are devised. The first weighted fusion approach involves calculating the mean sum of each attribute group embedding representation e_{gp} for the user, as shown below. N_g denotes the number of attributes possessed by the user.

$$E_{gp} = \frac{1}{N_g} \sum e_{gp} \tag{20}$$

The second approach for computing $sim(\bullet)$ involves first calculating the similarity γ_g between a user's personalized preference weight vector and the user attribute group preference weight vector. This assesses the similarity of the preferences between the user and attribute groups. Subsequently, the similarity coefficient γ_g obtained is utilized to weight the embedding representation e_{gp} for each attribute group, as depicted in Equation (21).

$$E_{gp} = \sum \gamma_g e_{gp} \tag{21}$$

$$e_{gp} = \sum \omega_i^{u_g} \bar{x}_i \tag{22}$$

$$E_p = \sum \omega_i^{u} \bar{x}_i \tag{23}$$

When fusing the embedded representations of multiple attributes of an item, since our model's embedding representation e_{uA} of user attributes is derived from fusing the preferences over each item attribute embedding, each attribute embedding of the item is treated equally as an inherent and immutable factor characterizing the item itself. The fusion of each attribute yields the attribute embedding representation e_{vB} of the item, as detailed below.

$$e_{vB} = \frac{1}{q} \sum \bar{x}_i \tag{24}$$

In this context, e_{vB} represents the attribute embedding representation of an item, where q denotes the number of attributes possessed by the item, and $\bar{x}_i$ signifies the embedded representation corresponding to each individual attribute of the item.

The combined attribute embedding representation e_{uA} of the user output from the user attribute preference fusion layer; the combined attribute embedding representation e_{vB} of the item; and the embedded representation $\bar{x}_i$ of the item attributes input into the multi-head self-attention layer, as well as the user embedding representation e_u and the item embedding representation e_v output from the initial mapping layer, are collectively fed into the attribute collaborative convolution module for graph convolution operations.

3.5. The Attribute Collaborative Convolution Layer

Based on the modeling of user attributes to derive user preferences for item attributes in the previous section, graph convolution operations are performed on the attribute collaboration graph to obtain the embedding representations of users and items. In the

attribute collaboration graph, each user u corresponds to two embedding representations: the embedding representation e_u of the user ID and the embedding representation e_{uA} of user attributes. Meanwhile, each item v corresponds to multiple embedding representations: the embedding representation e_{vB} of the item ID, the attribute embedding representation e_v of the item, and the embedding representations $\{\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_q\}$ corresponding to each attribute node of the item. For the propagation and aggregation processes on the attribute collaboration graph, an information dissemination strategy is adopted, where the embedding representation of a target node is obtained by performing graph convolutional propagation and aggregation of its neighboring nodes on the attribute collaboration graph.

Unlike other convolutional aggregation operations, our approach differentiates between different types of neighboring nodes for the target node rather than treating different node types in the neighborhood equally. Furthermore, interactions of the previously obtained embedding representations of user group attributes with those of item attributes are taken into account to perform similarity calculations. These similarity coefficients are then used as connection weights for nodes during the information aggregation process, enhancing the precision of convolutional aggregation and mitigating overfitting issues that may arise from multiple layers of aggregation, ultimately improving the accuracy of the recommendations.

As both the user's attribute embedding representation e_{uA} and the item's attribute embedding representation e_{vB} are derived through fusion based on the item attribute embedding representations, they possess rich semantics. Consequently, incorporating their similarity as the correlation between nodes during the convolutional aggregation process refines the interaction between nodes, enhancing the interpretability of the convolutional process.

3.5.1. The Embedding Representation of Item Nodes

During the convolutional aggregation on the attribute collaboration graph, an item node v aggregates information from its adjacent user nodes $u \in N_u^v$ and item attribute nodes $b \in N_b^v$. Here, N_u^v denotes the set of neighboring user nodes for item v, while N_b^v represents the set of neighboring attribute nodes for item v.

Given that item attributes are regarded as inherent values of the item, it is assumed that each attribute contributes equally to the embedding representation of the item's attributes. Consequently, an average aggregation approach to the item attributes is employed, denoted as $m_{v \leftarrow b} = (1/q)e_b$. Here, e_b signifies the embedding representation of attribute node b.

In the neighborhood of item v, the user node $u \in N_u^v$ aggregates $m_{v \leftarrow u} = \gamma_u^v e_u$, where e_u denotes the embedded representation of user IDs, and γ_u^v serves as a parameter that controls the flow of information relevant to user u towards item v.

When computing γ_u^v, a similarity calculation is first performed between the user's attribute embedding representation e_{uA} and the item's attribute embedding representation $m_{v \leftarrow b}$, where $b \in N_b^v$, both of which are obtained through fusing multiple attribute embedding based on items. This results in the similarity s_u^v between an item and its neighboring nodes. Finally, the similarity coefficients are normalized using Equation (25) to yield the normalized similarity coefficient γ_u^v between nodes.

$$\gamma_u^v = \frac{\exp(s_u^v)}{\sum_{u' \in N_u^v} \exp(s_{u'}^v)} \tag{25}$$

As the information on the target node itself is also crucial for its embedding representation, a self-connection function is employed to aggregate the information $m_{v \leftarrow v} = e_v$ from the node itself.

Finally, the information from the neighborhood of the target node v is fused to obtain the final embedding representation E_v of the item node, as shown in Equation (26). Specifically, a fully connected layer, followed by a LeakyReLU activation function, is utilized to aggregate the information from the neighboring nodes of the target node.

$$E_v = \text{LeakyReLU}(m_{v \leftarrow v}, \sum_{u \in N_v^u} m_{v \leftarrow u}, \sum_{b \in N_v^b} m_{v \leftarrow b}) \tag{26}$$

wherein $u \in N_u^u$ represents the user node u in the neighborhood of item node v, and $b \in N_v^b$ denotes the attribute node b in the neighborhood of item node v.

3.5.2. The Embedding Representation of User Nodes

During the convolution process on the attribute collaboration graph, the target node u aggregates messages passed from its adjacent item nodes $v \in N_v^u$. This involves the process $m_{u \leftarrow v} = \gamma_v^u e_v$, whereby the embedding representation vector of item v conveys information to user u. γ_v^u represents the parameter that controls the extent of the information flow from item v to user u, and its computation method is analogous to that for transmitting information from items to their surrounding user neighbors. This yields the attention weight s_v^u, reflecting the user's perceived importance of the item's attributes. Finally, through weight normalization, the final normalized attribute-aware attention weight γ_v^u is obtained.

$$\gamma_v^u = \frac{s_v^u}{\sum_{v' \in N_v^u} s_{v'}^u} \tag{27}$$

Analogous to the process of fusing multiple types of attribute information with the item nodes, considering the significance of a user's own node information for their embedding representation, self-connection is employed to incorporate their own information into the embedding representation, denoted as $m_{u \leftarrow u} = e_u$.

Ultimately, the final embedding representation E_u of user node $m_{u \leftarrow u} = e_u$ is derived by fusing the information from the neighborhood of target node u, as detailed in Equation (28). Specifically, the information from the neighboring nodes of the target node is aggregated through a fully connected layer, followed by a LeakyReLU activation function.

$$E_u = \text{LeakyReLU}(m_{u \leftarrow u}, \sum_{v \in N_u^v} m_{u \leftarrow v}) \tag{28}$$

3.6. The Prediction Layer and Optimization

To optimize the AAGCN model, the BPR (Bayesian Personalized Ranking) method is chosen to optimize the model parameters. The BPR model assumes that the interactions observed are indicative of stronger user preferences and should be assigned higher predicted values than unobserved interactions, as shown in Equation (29).

$$L_r = - \sum_{(u,i,j) \in O} \ln(\widehat{y}_{ui} - \widehat{y}_{uj}) + \lambda ||\theta||_2^2 \tag{29}$$

wherein $O = \{(u,i,j)|(u,i) \in O^+, (u,j) \in O^-\}$ represents the training data for the AAGCN model, O^+ denotes positive user–item interactions, and O^- represents non-positive user–item interactions. $\widehat{y}_{ui}$ signifies the preference score of user u towards item i, with the scoring function detailed in Equation (30).

$$\widehat{y}_{uv} = E_u \odot E_v \tag{30}$$

wherein $\odot$ denotes the inner product operation, while E^u and E^v represent the concatenated vectors of the composite attribute embeddings of users and items, respectively, with the graph convolutional behavioral preference embeddings.

4. Experiments

4.1. Datasets

To validate the effectiveness of the model, three publicly available datasets were selected, each containing rich user attribute information and item attribute information and varying in dataset size and sparsity: MovieLens-100K [4,16], MovieLens-1M [4,16], and DoubanBook [3]. MovieLens-100K and MovieLens-1M are a series of movie rating datasets collected and released by the GroupLens research group for the MovieLens website, a movie recommendation service platform. These datasets have been used widely in the fields of machine learning and recommendation systems. They contain integer ratings

ranging from 1.0 to 5.0 given by users to movies, as well as user attribute information, such as Age, Gender, and Occupation, and movie-related attribute information, such as Release Year, Genre, Actor, and Director. DoubanBook is an online book rating dataset from Douban. All of these datasets include multiple attributes for both users and items. Attributes such as user location and group, book author, publisher, and publication year are considered as features.

Based on the input requirements of the model, unformatted and non-standard data need to be converted into the format required by the model. Thus, the three datasets require preprocessing. First, the dataset is transformed into implicit feedback based on users' explicit rating records for items. If a user provides positive feedback, it is marked as 1, indicating an interaction between the user and the item. Negative feedback or a lack of interaction is marked as 0. For the rating threshold, scores greater than or equal to 4 are considered positive interactions. Next, user and item attribute information is converted according to the structure of the data. The relevant statistics for the MovieLens-100K, MovieLens-1M, and DoubanBook datasets after preprocessing are shown in Table 1.

Table 1. Statistics of datasets.

	Users	**Items**	**Interactions**	**Sparsity**	**User Attributes**		**Item Attributes**	
					Category	**Quantity**	**Category**	**Quantity**
Movie-100K	943	1682	100,000	0.063	3	32	4	4723
Movie-1M	6040	3900	1,000,209	0.42	3	32	4	8462
DoubanBook	13,254	22,415	801,120	0.002	2	2974	3	12,684

The aforementioned preprocessed datasets were partitioned into training, validation, and testing sets at an 8:1:1 ratio to conduct relevant experimental comparisons and analyses.

4.2. Evaluation Metrics

The Top-K recommendation system aims to present the K most intriguing items to users, with K typically varying as a parameter within the set 10, 20, 50, 100. Additionally, recall and Normalized Discounted Cumulative Gain (NDCG) serve as pertinent evaluation metrics within the Top-K recommendation system. These metrics are employed to benchmark the recommendation performance of relevant models through comparative analysis.

Recall@K denotes the proportion of items interacted with by the user, out of all of the items the user has interacted with in the test set, that are present within the list of K items recommended to the user. A higher recall value signifies the superior performance of the recommendation system.

$$\text{Recall@}K = \frac{\sum_{u \in U} |R(\text{u}) \cap T(u)|}{|T(u)|} \tag{31}$$

wherein $R(u)$ represents the list of K items recommended by the recommendation system to user u, whereas $T(u)$ signifies the comprehensive list of all items interacted with by user u within the test set. Ultimately, the Recall@K evaluation metric for the model is derived by averaging the Recall@K values across all users, each computed by comparing the recommended items against a user's actual interactions.

To define the Normalized Discounted Cumulative Gain (NDCG@K), it is first necessary to establish the concept of Discounted Cumulative Gain (DCG), as outlined below.

$$\text{DCG@}K = \sum_{i}^{K} \frac{2^{r(i)} - 1}{\log_2(i+1)} \tag{32}$$

wherein $r(i)$ represents the relevance of the item at the i-th position in the recommendation list to the user's interests. $r(i) = 1$ indicates the presence of an actual interaction between

the user and the item, whereas $r(i) = 0$ signifies the absence of such an interaction. Subsequently, NDCG (Normalized Discounted Cumulative Gain) is obtained by normalizing the DCG value by dividing it by the theoretically maximum DCG value. The definition of NDCG is as follows:

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{33}$$

Within this context, IDCG@K (Ideal Discounted Cumulative Gain at position K) represents the optimal DCG that a model could theoretically achieve in predicting the best recommendation list for a user. Compared to recall, Normalized Discounted Cumulative Gain (NDCG) not only considers the number of correctly predicted items but also accounts for their relative positions and relevance within the recommended list, making it a more comprehensive and reliable metric for evaluating the performance of recommendation systems.

4.3. Comparative Approaches

This section delves into the baseline models employed for comparison with the two proposed recommendation models in this paper. These recommendation models are categorized into four distinct groups.

(1) Graph-neural-network-based methods

NGCF [1]. The NGCF model is a GCN-based collaborative filtering recommendation algorithm that leverages graph data structures. It explicitly encodes the high-order connected topological structures of user–item interactions into collaborative information through an information propagation and aggregation mechanism applied to the user–item interaction graph. Subsequently, it utilizes the learned embeddings of users and items, which encapsulate high-order collaborative information, to generate recommendations.

LightGCN [11]. The LightGCN model is an algorithm that builds upon and improves upon the NGCF model. It replaces the nonlinear activation functions and feature transformation operations in the GCN with a simple weighted sum aggregator, employing a lightweight GCN operation to learn the user and item embeddings. This not only simplifies the model but also enhances the training efficiency of the recommendation algorithm and the encoding capability of user–item embedding vectors.

(2) Feature-interaction-focused methods

NFM [20]: The NFM model is an algorithm that builds upon and improves upon Factorization Machines (FMs). The model's feature interaction capabilities are enhanced by replacing the inner product operation of latent vectors in traditional FMs with a more expressive multi-layer neural network, thereby improving the performance of the recommendation system to a certain extent.

SAIN [4]: The SAIN model introduces a self-attention mechanism into feature interactions, enabling it to capture the interplay between user attribute features and item attribute features. This effectively integrates user–item interaction information with user and item attribute information, ultimately utilizing the fused embedding representations learned to generate recommendations.

(3) Attention-mechanism-integrated methods

AFM [21]: The AFM model enhances the FM model by incorporating an attention mechanism. Unlike the NFM model, which simply performs a weighted sum operation after feature interactions across different attribute types, AFM learns the importance weights of the resulting cross-features formed through feature interactions across various attribute types. By assigning greater attention weights to these cross-features, the AFM model improves its ability to model feature interactions, enhancing the overall modeling capacity of the system.

KGAT [2]: The KGAT model is a recommendation model that integrates graph neural networks with an attention mechanism. Based on a collaborative knowledge graph constructed from user–item interaction graphs and item knowledge graphs, KGAT aggregates neighbor information through an attention-based neighbor aggregator. It employs

an attention mechanism to determine the contribution of different neighbor nodes to the representation of the current node. Furthermore, a multi-layer attention network is utilized to learn the distinct node representations at each layer, with node embeddings being propagated and aggregated across layers. This effectively integrates graph topological information with item attribute features, enhancing the model's ability to learn graph embedding representations.

(4) Hybrid approaches that fuse attribute and behavioral methods

DG-ENN [6]: The DG-ENN model, based on the proposed user attribute graph and item attribute graph, leverages a GCN to independently learn the embedding representations of user and item attribute features. These learned attribute embeddings are then utilized to enhance the quality of the user and item embeddings derived from the user–item interaction graph, thereby improving the expressive power of the embedding representations.

AF-GCN [3]: The AF-GCN model introduces an attention-based attribute fusion module that integrates multiple attribute nodes for both users and items into composite attribute nodes. It then performs graph convolutional operations on a heterogeneous graph constructed from <user, item, attribute> triplets. Ultimately, the learned embeddings of users and items at different layers are utilized to accomplish recommendations.

AGNN [16]: The AGNN model, grounded in attribute graphs, proposes an eVAE architecture that infers the preference embeddings from the attribute distributions. Through a gated GNN structure, it effectively aggregates different types of attribute nodes in the target node's neighborhood, enhancing the embedding representation capabilities of attributes.

AGMR [17]: The AGMR model acknowledges that the influence of attributes and behaviors on the entity nodes varies, thus devising a fine-grained preference fusion strategy that integrates attribute-based group preferences with individual behavioral preferences. This approach enhances the accuracy, comprehensiveness, and personalization of the embedding representations.

4.4. Parameter Settings

In all models, the embedding dimensions for the user IDs and item IDs were fixed at 64, while those for user attributes and item attributes were set to 64. The number of graph neural network layers was configured as 2, with a regularization parameter of 0.001 and a batch size of 1024. All model parameters were optimized using the Adam optimizer. Regarding hyperparameter tuning, a grid search method was employed, specifically adjusting the learning rate within the range of $\{0.0001, 0.0005, 0.001, 0.005\}$ and initializing the model parameters with the Xavier initializer. As the majority of mainstream models adopt the Top-20 recommendation task for evaluating model performance, this comparative experiment followed suit. Furthermore, an early stopping strategy was implemented during model training, whereby training would cease prematurely if the Recall@20 on the validation dataset failed to improve over 50 consecutive epochs. A comparative analysis of the proposed model against relevant baseline models was conducted on the Top-20 recommendation task, with Recall@20 and NDCG@20 serving as quantitative metrics for assessing the model performance.

4.5. Experimental Results

An analysis of the experimental comparison results presented in Table 2 reveals that the AAGCNR model outperforms ten baseline methods on both datasets in the Top-20 recommendation task.

Compared with the best model, NGCF, based on the user–item interaction features in graph neural networks, our model achieves an improvement of 5.57% in Recall@20 and 4.33% in NDCG@20 on the MovieLens-100K dataset; 15.00% in Recall@20 and 16.88% in NDCG@20 on the MovieLens-1M dataset; and 31.87% in Recall@20 and 29.88% in NDCG@20 on the DoubanBook dataset. Compared with the best results from the feature

interaction models SAIN and AFM, which incorporate attention mechanisms, our model achieves an improvement of 4.28% in Recall@20 and 1.61% in NDCG@20 on the MovieLens-100K dataset; 5.34% in Recall@20 and 2.80% in NDCG@20 on the MovieLens-1M dataset; and 8.85% in Recall@20 and 7.32% in NDCG@20 on the DoubanBook dataset. This demonstrates the effectiveness of the AAGCNR model in combining user–item interaction features with attribute interaction features.

Compared with the models DG-EN and AGMR, which combine attribute feature interactions and behavioral features, our model achieves an improvement of 2.10% in Recall@20 and 1.35% in NDCG@20 on the MovieLens-100K dataset; 6.28% in Recall@20 and 3.20% in NDCG@20 on the MovieLens-1M dataset; and 11.89% in Recall@20 and 9.45% in NDCG@20 on the DoubanBook dataset. This further confirms the effectiveness of the AAGCNR model in integrating user–item interaction features with attribute interaction features.

The comparative analysis of the experimental results with baseline models from different fields demonstrates the effectiveness of the AAGCNR model, which combines user–item interaction features with attribute interaction features. It more effectively integrates interaction features and user–item attribute interaction features, learning richer and more accurate embedding representations, thereby improving the performance of the recommendation system.

Table 2. Comparative results of experiments for the Top-20 recommendation task.

Methods	Movie-Lens 100K		Movie-Lens 1M		DoubanBook	
	Recall@20	**NDCG@20**	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**
NGCF	0.323	0.485	0.12	0.314	0.1324	0.1422
LightGCN	0.31	0.469	0.095	0.294	0.1071	0.1153
NFM	0.293	0.385	0.086	0.204	0.0983	0.1068
SAIN	0.326	0.498	0.131	0.357	0.1604	0.1721
AFM	0.327	0.49	0.108	0.246	0.0747	0.0750
KGAT	0.313	0.478	0.117	0.301	0.0846	0.1024
DG-EN	0.333	0.499	0.134	0.326	0.1702	0.1788
AF-GCN	0.329	0.489	0.126	0.318	0.143	0.1494
AGNN	0.308	0.454	0.093	0.304	0.1384	0.1499
AGMR	0.334	0.497	0.129	0.335	0.1566	0.1798
AAGCNR	0.341	0.506	0.138	0.367	0.1746	0.1847

4.6. Ablation Experiments and Parametric Analysis

4.6.1. Ablation Experiments

To further verify the effectiveness of combining attribute interaction feature information with graph convolution feature information compared to using only one type of feature and to assess the role of the attention fusion layer in enhancing the user and item embeddings, we designed the following variants based on the AAGCNR model:

AGCNR_DI: This variant removes the bilinear interaction layer compared to AAGCNR and does not consider the relationships between user and item attributes.

AAGCNR_DG: This variant removes the graph convolution model compared to AAGCNR and directly uses the output of the user preference fusion module as the input to the prediction layer.

AAGCNR_DA: This variant removes the self-attention layer compared to AAGCNR and does not capture the relationships between attributes in different semantic spaces.

The effectiveness of each module in the model is validated through the Top-20 recommendation task. The results of the ablation experiments are shown in Table 3.

As shown in Table 3, the models AAGCNR-DI, which removes the interaction feature information between attributes, and AAGCNR-DG, which removes the graph convolution module, perform the worst. Both are inferior to the AAGCNR-DA model, which lacks the attention mechanism. This indicates that the embedding representations of users and

items are significantly influenced by the characteristics of low-frequency items. During the graph convolution process, distinguishing the categories of neighboring nodes is crucial, as different types of nodes affect the target node. Therefore, integrating the interaction features between attributes with the behavioral features between users and items and incorporating them through graph convolution can embed more information into the representations of users and items, thereby improving the recommendation performance of the model. For the AAGCNR-DA model, which removes the attention mechanism, the adaptive attention coefficients a and b were set to fixed values of 0.5 during the experiment, eliminating the influence of the attention mechanism on the embedding representations of users and items. The experimental results of the AAGCNR-DA model were significantly lower than those of the AAGCNR model, indicating that the adaptive attention fusion mechanism is crucial in balancing the embeddings of user attributes and user IDs, as well as item attributes and item IDs, for the final embedding representations of users and items.

Table 3. Ablation experiment results for the AAGCNR model.

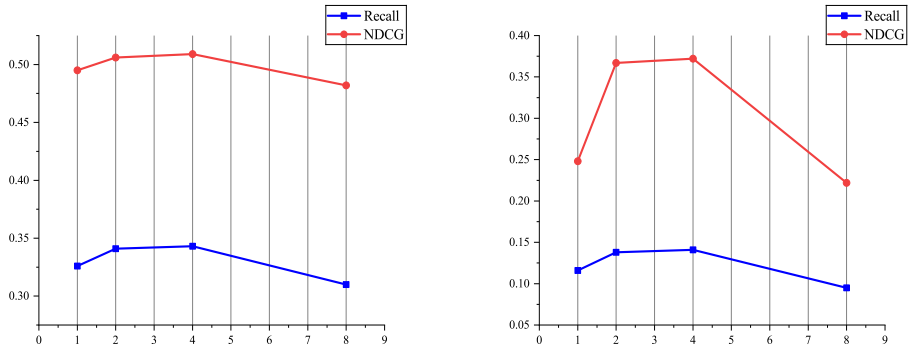
Methods	Movie-Lens 100K		Movie-Lens 1M		DoubanBook	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
AAGCNR-DI	0.325	0.489	0.123	0.315	0.1688	0.1702
AAGCNR-DG	0.307	0.384	0.088	0.210	0.1603	0.1657
AAGCNR-DA	0.328	0.418	0.126	0.322	0.1720	0.1796
AAGCNR	0.341	0.506	0.138	0.367	0.1746	0.1847

4.6.2. Parametric Analysis

Experiments were conducted to investigate the impact of varying the hyperparameters on the performance of the AAGCNR model. These hyperparameters included the embedding dimension d for user and item IDs, the embedding dimension d_a for user and item attributes, and the number of multi-head self-attention heads h. The results were compared to assess the influence of different parameter settings on the model's overall performance.

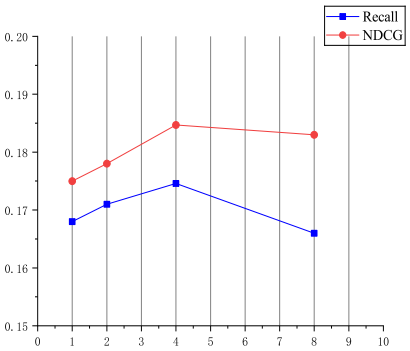
The impact of varying the number of multi-head self-attention heads h on the experimental results is compared in Figure 2 when using the Top-20 recommendation task on the Movielens-100K and Movielens-1M datasets and the DoubanBook dataset.

The comparative analysis of the experimental results in Figure 2 reveals that with all the other model parameters held optimal and constant, increasing the number of attention heads h from 1, 2, 4, to 8 initially leads to an improvement in model performance. However, when h reaches 8, a decline in performance is observed on both datasets, indicating that an excessive number of attention heads introduces excessive noise into the interaction features between attributes, thereby negatively affecting the model performance.



(**a**) (**b**)

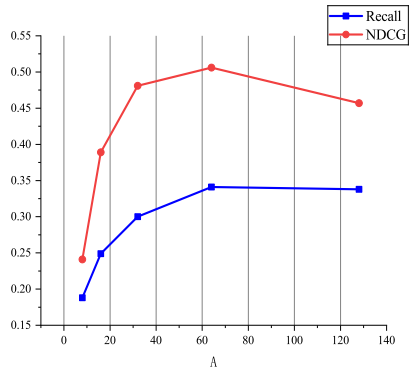
Figure 2. *Cont.*



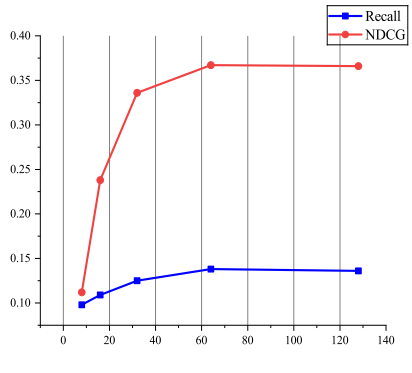
(**c**)

Figure 2. Impact of varying the number of attention heads h on experimental results. (**a**) Movielens-100K. (**b**) Movielens-1M. (**c**) DoubanBook.

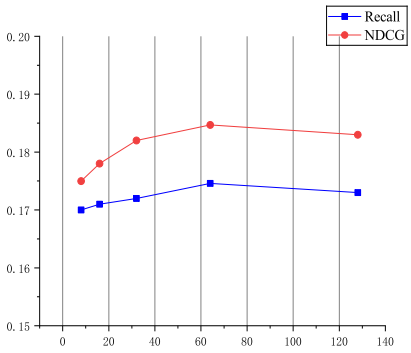
Figure 3 illustrates the impact of varying the embedding dimension d for the user and item IDs on the experimental results, as assessed through the Top-20 recommendation task on the Movielens-100K and Movielens-1M datasets and the DoubanBook dataset.



(**a**)



(**b**)



(**c**)

Figure 3. Impact of varying the embedding dimension d for user and item IDs on experimental results. (**a**) Movielens-100K. (**b**) Movielens-1M. (**c**) DoubanBook.

The comparative analysis of the experimental results in Figure 3 indicates that with all the other model parameters held optimal and constant, increasing the embedding dimension d for user and item IDs from 8, 16, 32, 64, to 128 leads to a consistent upward trend in the performance on both datasets. Specifically, on the Movielens-100K and DoubanBook

datasets, the optimal performance is achieved at $d = 64$, followed by a decline in model performance for $d > 64$, suggesting that excessive noise in the embedding representations hinders further performance gains. On the Movielens-1M dataset, the optimal performance is also observed at $d = 64$, with fluctuations noted for $d > 64$, characterized by a slight decrease in Recall@20 and a marginal increase in NDCG@20.

Figure 4 illustrates the comparative impact of varying the embedding dimension d_a for the user and item attributes on the experimental results, as evaluated through the Top-20 recommendation task on the Movielens-100K, Movielens-1M, and DoubanBook datasets.

Through a comparative analysis of the experimental results in Figure 4, it is evident that with the other model parameters held optimal and unchanged, by adjusting the embedding dimension $d_a = 8, 16, 32, 64, 128$ for the user and item attributes, the performance improves on both datasets as d_a increases, reaching the optimal value at $d_a = 64$.

Through a comparative analysis of the impact of varying the embedding dimension d for the user and item IDs and the embedding dimension d_a for attributes on the experimental results, it is evident that the interactive features between attributes, as well as user–item interaction features, are both effective in enhancing the recommendation performance. Furthermore, the magnitude of the change in Recall and NDCG for the ID embedding dimension d across the range of 8 to 128 is lower than that observed for the attribute embedding dimension d_a within the same range, indicating that attribute interactions play a role in alleviating the sparsity issue inherent in user–item interaction behavior features.

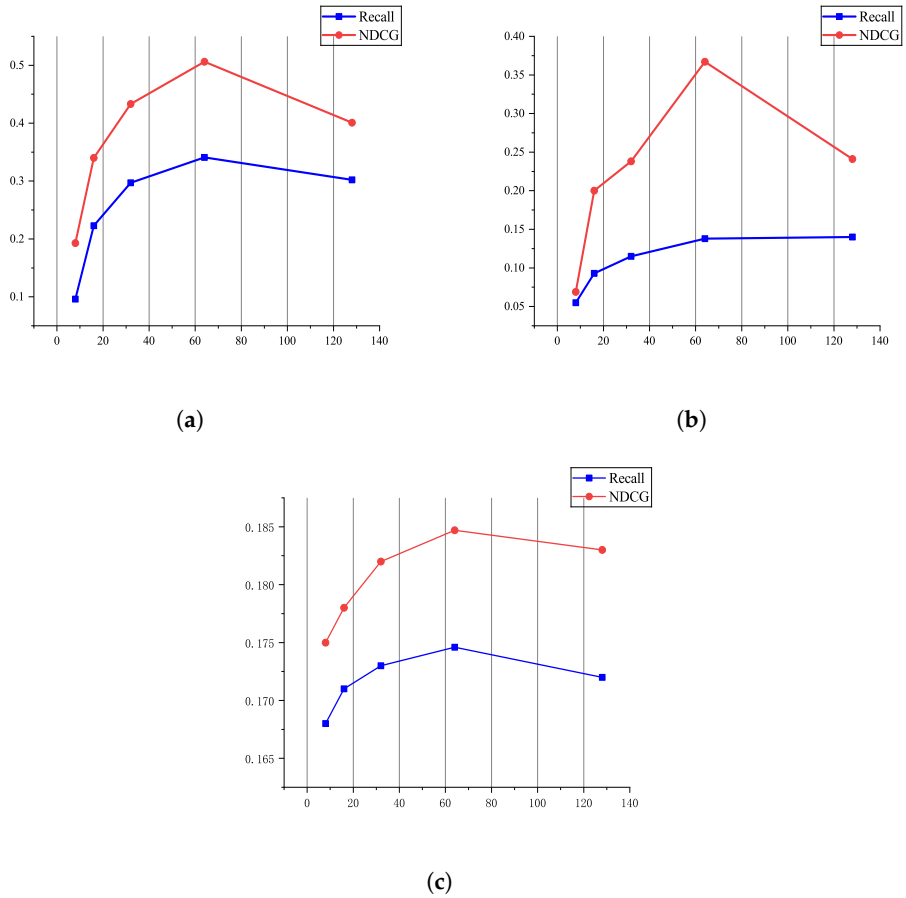


Figure 4. Impact of varying user–item attribute embedding dimension d_a on experimental results. (**a**) Movielens-100K. (**b**) Movielens-1M. (**c**) DoubanBook.

5. Conclusions

This paper introduces an attribute-aware graph convolutional recommendation method, AAGCNR. The model first incorporates both user and item attribute features into the graph.

By constructing attribute interaction features alongside user behavioral features, it addresses the limitations of relying solely on user–item interaction data, effectively mitigating the issue of sparse interaction data. Additionally, the model refines the aggregation process of neighboring nodes around the target node by constructing a collaborative convolutional graph. The interactions during the graph convolution process are informed by correlations between the composite attribute embeddings, thereby enhancing the interpretability of the graph convolution operation.

While existing graph recommendation algorithms, which integrate user and item attribute interaction features with user–item behavioral features, have shown some success, certain areas still require improvement and optimization. Based on the findings of this study, future research could explore the following directions:

(1) Most attribute-based recommendation algorithms are relatively slow to detect changes in user preferences. By integrating variables such as time factors and item popularity, models could become more responsive to newly introduced popular items.

(2) The current recommendation algorithms predominantly focus on analyzing and modeling users' historical behaviors and the attributes of items they have previously engaged with. Introducing interactive recommendation mechanisms could increase the exposure of items that users have not yet interacted with.

Author Contributions: Conceptualization, N.W., J.D. and J.G.; Methodology, N.W. and J.G.; Software, N.W. and Y.L.; Validation, N.W.; Formal analysis, N.W.; Investigation, N.W.; Resources, J.D.; Data curation, Y.L. and J.D.; Writing—original draft, N.W., Y.L. and J.D.; Writing—review & editing, J.G.; Visualization, J.D.; Supervision, X.C. and J.G.; Project administration, X.C. and J.G.; Funding acquisition, X.C. and J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the *S&T* Program of Hebei (No. 226Z0102G and No. 21310101D); the National Natural Science Foundation of China (No. 42306218 and No. 62172352); and Hebei Natural Science Foundation (F2023407003).

Data Availability Statement: All the data generated or analyzed during this study are included in this published article.

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

References

1. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T.S. Neural graph collaborative filtering. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 165–174.
2. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T.-S. KGAT: Knowledge Graph Attention Network for Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 950–958.
3. Yue, G.; Xiao, R.; Zhao, Z.; Li, C. AF-GCN: Attribute-fusing graph convolution network for recommendation. *IEEE Trans. Big Data* **2022**, *9*, 597–607. [CrossRef]
4. Yun, S.; Kim, R.; Ko, M.; Kang, J. Sain: Self-attentive integration network for recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 1205–1208.
5. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 417–426.
6. Guo, W.; Su, R.; Tan, R.; Guo, H.; Zhang, Y.; Liu, Z.; Tang, R.; He, X. Dual Graph enhanced Embedding Neural Network for CTR Prediction. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 496–504.
7. Bianchi, F.M.; Grattarola, D.; Alippi, C. Spectral clustering with graph neural networks for graph pooling. In Proceedings of the 37th International Conference on Machine Learning, Virtual Event, 13–18 July 2020; pp. 874–883.
8. Hamilton, W.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1025–1035
9. 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; pp. 974–983.

10. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 3307–3313.
11. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020; pp. 639–648.
12. Mao, K.; Zhu, J.; Xiao, X.; Lu, B.; Wang, Z.; He, X. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Queensland, Australia, 1–5 November 2021; pp. 1253–1262.
13. Wang, X.; Huang, T.; Wang, D.; Yuan, Y.; Liu, Z.; He, X. Learning intents behind interactions with knowledge graph for recommendation. In Proceedings of the Web Conference, Ljubljana, Slovenia, 19–23 April 2021; pp. 878–887.
14. Gong, N.Z.; Talwalkar, A.; Mackey, L.; Huang, L.; Shin, E.C.R.; Stefanov, E.; Shi, E. Joint link prediction and attribute inference using a social-attribute network. *ACM Trans. Intell. Syst. Technol.* **2014**, *5*, 1–20. [CrossRef]
15. Yang, C.; Zhong, L.; Li, L.-J.; Jie, L. Bi-directional joint inference for user links and attributes on large social graphs. In Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, 3–7 April 2017; pp. 564–573.
16. Qian, T.; Liang, Y.; Li, Q.; Xiong, H. Attribute graph neural networks for strict cold start recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 3597–3610. [CrossRef]
17. Sun, K.; Liu, S.; Du, Y. Movie Recommendation Model Based on Attribute Graph Attention Network. *Comput. Sci.* **2022**, *49*, 294–301. (In Chinese)
18. Wu, L.; Yang, Y.; Zhang, K.; Hong, R.; Fu, Y.; Wang, M. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020; pp. 679–688.
19. Liu, F.; Cheng, Z.; Zhu, L.; Liu, C.; Nie, L. An attribute-aware attentive GCN model for attribute missing in recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 4077–4088. [CrossRef]
20. 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, Shinjuku, Tokyo, Japan, 7–11 August 2017; pp. 355–364.
21. 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.

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.