Neural Graph Matching based Collaborative Filtering

Yixin Su
University of Melbourne, Australia
yixins1@student.unimelb.edu.au

Rui Zhang*†
www.ruizhang.info, China
rayteam@yeah.net

Sarah M. Erfani*
University of Melbourne, Australia
sarah.erfani@unimelb.edu.au

Junhao Gan
University of Melbourne, Australia
junhao.gan@unimelb.edu.au

ABSTRACT

User and item attributes are essential side-information; their interactions (i.e., their co-occurrence in the sample data) can significantly enhance prediction accuracy in various recommender systems. We identify two different types of attribute interactions, *inner interactions* and *cross interactions*: inner interactions are those between *only* user attributes or those between *only* item attributes; cross interactions are those between user attributes and item attributes. Existing models do not distinguish these two types of attribute interactions, which may not be the most effective way to exploit the information carried by the interactions. To address this drawback, we propose a neural Graph Matching based Collaborative Filtering model (GMCF), which effectively captures the two types of attribute interactions through modeling and aggregating attribute interactions in a graph matching structure for recommendation. In our model, the two essential recommendation procedures, characteristic learning and preference matching, are explicitly conducted through graph learning (based on inner interactions) and node matching (based on cross interactions), respectively. Experimental results show that our model outperforms state-of-the-art models. Further studies verify the effectiveness of GMCF in improving the accuracy of recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender Systems; Attribute Interactions; Neural Graph Matching; Graph Neural Networks; Collaborative Filtering

ACM Reference Format:
Yixin Su, Rui Zhang, Sarah M. Erfani, and Junhao Gan. 2021. Neural Graph Matching based Collaborative Filtering. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462833

*Corresponding authors.
†Rui Zhang did this work while working at the University of Melbourne.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462833

1 INTRODUCTION

Collaborative Filtering (CF) is one of the most frequently used algorithms in recommender systems. It performs the predictions based on an assumption that similar users will have common preferences on similar items. For example, matrix factorization based recommender systems [14, 19, 36] learn user and item embeddings from user-item interactions (e.g., click, purchase) to exhibit similarities between similar users and between similar items. One of the most important challenges faced by CF is how to consider user and item attributes (e.g., user genders, item colors) to enhance the prediction performance. To this end, recent CF models conduct attribute embedding to capture fine-grained collaborative information and reveal the similarities between attributes [1, 29]. While learning attribute embeddings, considering the co-occurrence of attributes inside each data sample, i.e., *attribute interactions*, have been proven essential in providing useful information for more accurate predictions [25, 28]. For example, in a movie recommendation system, the director *Nolan* is a master of producing *sci-fi* movies. In this scenario, considering the attribute interaction <*Nolan*, *sci-fi*> is more effective than considering the two attributes separately. Generally, an attribute-interaction-aware CF model takes the attribute interactions into account (by modeling and aggregating them) to jointly decide the final predictions. Factorization Machine (FM) [25] models each attribute interaction as a dot product of two embedded vectors and aggregates all the modeling results linearly. With the development of Graph Neural Networks (GNNs), Li et al. [21] and Su et al. [30] leverage the relation modeling ability of GNNs to capture more sophisticated attribute interaction information and aggregate the information through graph learning.

While these models capture the co-occurrence between attributes by modeling attribute interactions, we argue that they may not be sufficient to yield a satisfactory joint decision. The key reason is that these models simply treat all the attribute interactions equally, and hence, model and aggregate them in the same way. However, different attribute interactions should have different impacts on the final prediction. Specifically, when the attribute interactions are exploited to perform joint decisions, they could play different roles for recommendation. For example, without loss of generality, the goal of movie recommendation is to predict the preference of a user (represented by the user's attributes) on a movie (represented by the item's attributes). Therefore, the attribute interactions are naturally

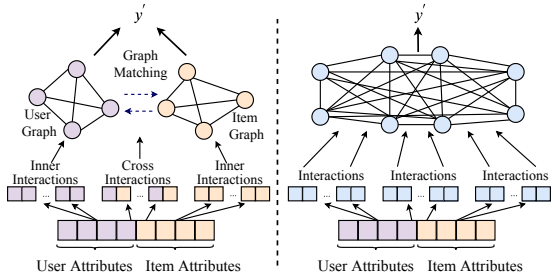

Figure 1: Illustration of the differences between GMCF (left) and existing work (right). GMCF treats attribute interactions differently in a structure of graph matching, while existing work treats all attribute interactions equally.

divided into two categories based on their roles for recommendation. First, the interactions between user attributes *only* (or between item attributes *only*), e.g., <*male*, *18-24*>, <*Nolan*, *sci-fi*>, serve as user (item) characteristic learning. These interactions are called *inner interactions*. Second, the interactions between user attributes and item attributes, e.g., <*18-24*, *sci-fi*>, serve as preference matching, such as "whether users in 18-24 years old like sci-fi movies". These interactions are called *cross interactions*. Existing work do not distinguish these two types of interactions. Such unawareness of attribute interaction types inevitably limits the capability of the existing work of leveraging attribute interactions for producing satisfactory joint decisions.

In this work, we explicitly model and aggregate inner interactions and cross interactions in different ways in a graph matching structure. Specifically, we use a graph composed of a user's (an item's) attributes to represent the user (the item). Each attribute is a node, and each pairwise attribute interaction is an edge. Then, we propose a neural Graph Matching based Collaborative Filtering model, called GMCF. Our GMCF uses a GNN to model a user attribute graph and an item attribute graph based on their inner interactions, respectively. Meanwhile, it matches the two attribute graphs at node levGel based on cross interactions, and makes the final prediction result. Figure 1 illustrates the differences between GMCF and existing work (GNN-based work as an example) in modeling and aggregating attribute interactions. As a result, GMCF consider attribute interactions in a way that fits what a recommender system aims to do: learn user and item characteristics (through graph learning), and match users' preference to items based on their characteristics (through graph matching).

We summarize the main contributions of our work as follows:

- We highlight the importance of considering the different impacts of attribute interactions on recommendation predictions, and categorize them into inner interactions and cross interactions based on their roles for recommendation, which deserve to be treated differently when they jointly decide the predictions.
- We propose a novel neural graph matching-based model, GMCF, that effectively leverages inner interactions for user and item characteristic learning (through graph learning), and exploits cross interactions for preference matching (through graph matching). [1] To our best knowledge, we are the first to represent each user and each item as an attribute graph, and model and aggregate attribute interactions in a graph matching structure.
- We conduct extensive experiments. Experimental results show that (i) our model outperforms state-of-the-art baselines in terms of accuracy; (ii) GMCF is able to effectively model the two types of attribute interactions to produce accurate predictions.

[1] Our implementation of the GMCF model is available at: https://github.com/ruizhang-ai/GMCF_Neural_Graph_Matching_based_Collaborative_Filtering.

2 RELATED WORK

In this section, we discuss more related work on attribute-aware CF models, graph neural networks, and graph matching methods.

2.1 Attribute-aware CF models

Factorization machine (FM) [25] is one of the most frequently used collaborative filtering algorithms that takes attribute interactions into account. FM performs interaction modeling between each pair of attributes and sums up all the modeling results to make the final prediction. Some extensions of FM further calculates an attention score [39] or a gate [22] for each interaction modeling result to differentiate each interaction's importance using the attention mechanism. However, these models consider the structural information linearly, which is ineffective in leveraging attribute interaction to make a joint decision. NFM [13] and DeepFM [11] add a multilayer perceptron (MLP) on top of attributes or attribute interactions, aiming to implicitly capture the structural information in a non-linear way. However, the use of MLP to model the interactions between input variables has been proven less effective [4, 28]. GMCF models and aggregates attribute interactions explicitly in a structure of graph matching, which is more effective in attribute interaction modeling and structural information capturing.

2.2 Graph Neural Networks

Graph neural networks (GNNs) facilitate learning about entities and their relations [18, 23, 40, 43]. Existing work leverages GNNs in various domains, such as molecular property prediction [9, 10] and object relation learning physical systems [3, 6]. Recently, GNNs attract attention in the domain of recommender systems. Some work considers the user-item interactions as a bipartite graph, where an edge between a user and an item indicates an interaction (e.g., click or rate) [15, 31, 35]. These models only consider user-item interactions in GNNs. Other work leverages GNNs to model knowledge graphs for recommendation [33, 34, 38, 42]. These models regard edges as predefined relations between attributes and items (users) instead of between attributes. Therefore, they do not consider the attribute interactions. Li et al. [21] and Su et al. [30] leverage GNNs to perform attribute interaction modeling and aggregation as a graph learning procedure. However, these models analyze all attribute interactions equally, which are ineffective in capturing useful structural information of attribute interactions to make a joint decision. Our model differentiates inner interactions and cross interactions, and models and aggregates these interactions in a graph matching structure that is considered as more suitable for recommendation.

2.3 Graph Matching

Graph matching is a long-standing research topic in computer science such as database and data mining domains [8, 41]. The goal of graph matching is to discover the similarity between two

graph form representations. Traditional graph matching algorithms are based on heuristic rules, such as minimal graph edit distance [24, 37], or based on graph kernel methods, such as random walks inside graphs [16, 32] and graph sub-structures [26, 27]. With the development of GNNs, recent work explores neural-based graph matching. Bai et al. [2] fuses the graph-level embeddings learned by GNNs and a node-level matching embedding learned by a pairwise node comparison method. Li et al. [20] uses GNNs to learn two graphs' embeddings in a Siamese network framework [5], with a node-level matching performed in the fusing procedure. However, the neural graph matching methods have not been fully explored in recommender systems yet. We are the first to represent each user and each item in a graph form and leverage the framework of neural graph matching for preference matching.

3 PROBLEM STATEMENT & BACKGROUND

In this section, we introduce the problem definition, the representations of attribute interactions, and the basic idea of GNNs.

3.1 Problem Statement

Denote by $\mathcal{J}^U$ the *universe* of the user attributes and by $\mathcal{J}^I$ the *universe* of the item attributes. An *attribute-value pair* is a name-value pair, denoted by (att, val), where att is the name of the attribute and val is the value on this attribute. For example, $(Male, 1)$ and $(Female, 1)$ mean the user's gender is Male and Female, respectively, where $Male \in \mathcal{J}^U$ and $Female \in \mathcal{J}^U$ are considered as two user attributes. Let D be a set of N training data pairs, denoted by $D = \{(\boldsymbol{x}_n, y_n)\}_{1 \le n \le N}$. In each training data $(\boldsymbol{x}_n, y_n) \in D$,

- $\boldsymbol{x}_n$ is called a *data sample*, which consists of:
  - a set $C_n^U = \{c^U = (att, val)\}_{att \in J_n^U}$ of attribute-value pairs with respect to a set of user attributes $J_n^U \subseteq \mathcal{J}^U$, and
  - a set $C_n^I = \{c^I = (att, val)\}_{att \in J_n^I}$ of attribute-value pairs with respect to a set of item attributes $J_n^I \subseteq \mathcal{J}^I$.

  Moreover, C_n^U and C_n^I are respectively called the *characteristic* of the user and the item specified in this data sample.
- $y_n \in \mathbb{R}$ is the *implicit feedback* (e.g., watched, liked) of the user on the item.

It should be noted that the number of attribute-value pairs in each data sample may be different, as the information of some attributes may be missing, or may contain *multiple* attributes of the same type, e.g., a movie may belong to multiple genres.

The Recommendation Problem. The goal of the *Recommendation Problem* studied in this paper is to design a *predictive model* $F(\boldsymbol{x}_n)$ such that for an input data sample $\boldsymbol{x}_n$ (specifying the characteristics of a user and an item), the output of the model, $F(\boldsymbol{x}_n)$, is a prediction on the *true* feedback, y_n, of the user on the item.

Our Solution. In this paper, we propose a *Neural Graph Matching based Collaborative Filtering* (GMCF) model $F_{GMCF}(\boldsymbol{x}_n)$. Our GMCF models the user and the item characteristics as two graphs, respectively, and leverages these user and item graph representations to predict y_n by graph matching techniques.

3.2 Representing Attributes and Interactions

Each attribute $att \in \mathcal{J}^U \cup \mathcal{J}^I$ is embedded as a vector $\boldsymbol{v}$ in d-dimensional space $\mathbb{R}^d$. This process can be seen as building a parameter matrix as an embedding lookup table. All the data samples share the same vector embedding $\boldsymbol{v}_{att}$ of the same attribute att, but they may have different *scalar* on the vector due to the potentially different values val's. More specifically, for an attribute-value pair (att, val), the corresponding vector, $\boldsymbol{u}_{att}$, is computed as $\boldsymbol{u}_{att} = val \cdot \boldsymbol{v}_{att}$; and such $\boldsymbol{u}_{att}$ is called the *representation* of the attribute-value pair. Initially, the $\boldsymbol{v}_{att}$ of each attribute is set as a random vector. In the following, when the context of the attribute att is clear, we omit the subscript from $\boldsymbol{v}$ and $\boldsymbol{u}$ for simplicity.

The co-occurence of two attributes att_1 and att_2 in a data sample is defined as an *interaction* between att_1 and att_2. Such an interaction is modeled by a function $f(\boldsymbol{u}_1, \boldsymbol{u}_2) : \mathbb{R}^{2\times d} \to \mathbb{R}^\ell$, where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are the representations of the attribute-value pairs of att_1 and att_2 (in the same data sample), and ℓ is the dimensionality of the output. Since the set of attributes, J_n^U and J_n^I, in each data sample may be different, the interactions specified in different data samples could be different. And because each attribute may appear (with different values) in multiple data samples, the collaborative information of the interactions in different data samples would further help discover the interactions between attributes that have never co-occurred. Therefore, $f(\cdot, \cdot)$ actually learns attribute embeddings that capture the collaborative information between the attributes (i.e., similar attributes would have similar embeddings) [25].

Our proposed model GMCF categorizes attribute interactions in a data sample into two types, *inner interactions* and *cross interactions*. More specifically, the interactions between user attributes only and between item attributes only are defined as inner interactions. However, on the other hand, those between one user attribute and one item attribute are cross interactions. As we will see shortly in Section 4, our GMCF model deploys different functions, $f(\cdot, \cdot)$, to model these two kinds of attribute interactions, and respectively uses them for different purposes: (i) user and item characteristic learning, and (ii) recommendation.

3.3 Graph Neural Networks

Consider a graph $G = \langle V, E \rangle$, where $V = \{\boldsymbol{v}_i\}_{1 \le i \le k}$ is the set of k nodes, each of which is represented by a vector representation $\boldsymbol{v}_i$, and E is the set of edges which indicate the neighborhood information between nodes: two nodes are neighbors if they are linked by an edge. A Graph Neural Network (GNN) learns the vector representation of each node by message passing, a procedure of aggregating neighborhood information. Specifically, the message passing procedure for node i first aggregates the vector representations of all its neighbors. Then, it gets the fused representation of node i by fusing $\boldsymbol{v}_i$ and the aggregated vector representation. Formally, the fused vector representation of node i, $\boldsymbol{v}_i^{'}$, through the graph modeling is:

$$\boldsymbol{v}_i^{'} = f_{fuse}(\boldsymbol{v}_i, Aggregate_v(\{\boldsymbol{v}_j\}_{j \in N(i)})), \tag{1}$$

where f_{fuse} is the fusing function, $Aggregate_v(\cdot)$ is an aggregation function that aggregates the neighborhood embeddings into a fixed dimension representation (e.g., element-wise sum) and $N(i)$ is the set of all the neighbors of node i.

If it is necessary, the graph representation can be computed as the aggregation of the vector embeddings of all the nodes: $\boldsymbol{v}_G = Aggregate_G(\{\boldsymbol{v}_i^{'}\}_{i \in V})$, where $\boldsymbol{v}_G$ is the graph representation and $Aggregate_G(\cdot)$ is a node aggregation function that is similar to the one for aggregating nodes' neighbors.

4 OUR APPROACH

In this section, we describe our model in detail. First, we give an overview of our GMCF model. Then, we unveil the details of each part of the model. Finally, we discuss the relations of our model to existing work and applicable situations.

4.1 GMCF Overview

Our proposed model GMCF mainly consists of three modules: (i) User and Item Graph Construction Module, (ii) Node Matching based GNN Module, and (iii) Graph Representation Matching Module. Figure 2 shows an overview of the GMCF model. In particular, for an input data sample $\boldsymbol{x} = \{c_i^U = (att, val)\}_{1 \le i \le p} \cup \{c_j^I = (att, val)\}_{1 \le j \le q}$, each module works as follows.

The User and Item Graph Construction Module. GMCF constructs a *user attribute graph* and an *item attribute graph* respectively based on the user and item characteristic specified in the data sample $\boldsymbol{x}$. More specifically, the user attribute graph (resp., item attribute graph) is a *complete graph*, where each node corresponds to an attribute-value pair (att, val) in the user (resp., item) characteristic and is represented by the representation $\boldsymbol{u} = val \cdot \boldsymbol{v}$ of the pair.

The Node Matching based GNN Module. For each node i with representation $\boldsymbol{u}_i$ in the user (resp., item) attribute graph, this module first computes the *message passing information* $\boldsymbol{z}_i$ and the *node matching information* $\boldsymbol{s}_i$ with the item (resp., user) attribute graph. It then fuses $\boldsymbol{u}_i$, $\boldsymbol{z}_i$ and $\boldsymbol{s}_i$ to obtain a fused node representation $\boldsymbol{u}_i'$ for the node i. Finally, a graph representation $\boldsymbol{v}_G$ is obtained by aggregating the fused node representation $\boldsymbol{u}_i'$ for all nodes.

The Graph Representation Matching Module. Our GMCF performs a graph matching between the user and item attribute graph representations. The final prediction is obtained from matching result.

Next, we introduce the details of the three modules.

4.2 User & Item Graph Construction

We represent each user and each item as an *attribute graph*, with their attributes as nodes and their inner interactions as edges. Specifically, for each data sample $\boldsymbol{x}$, the user attributes are the nodes of the user attribute graph. Each node i that represents an attribute takes the attribute representation $\boldsymbol{u}_i^U$ as the node representation. Therefore, we represent the node set of the user attribute graph as $V^U = \{\boldsymbol{u}_i^U\}_{i \in \mathcal{J}^U}$. Then, each pair of nodes in the graph are connected with an edge to indicate the pairwise interaction between the two attributes. In summary, each user attribute graph is represented as $G^U = \langle V^U, E^U \rangle$, where E^U is the edge set that contains all edges in the graph. Note that since we consider all pairwise attribute interactions, the user attribute graph is a complete graph. We perform the same transformation for item attributes and get the item graph $G^I = \langle V^I, E^I \rangle$. In practice, we perform the graph construction by simply dividing attribute interactions as edges and matching pairs for different modules, which does not take additional effort than other explicit pairwise interaction modeling methods (e.g., AutoInt, Fi-GNN).

4.3 Node Matching based GNN

We propose a node matching based GNN, f_G that considers both inner interactions through message passing and cross interactions through node matching. In this section, we describe how f_G outputs the graph representation through modeling the two types of interactions. Note that since the modeling on user attribute graphs and item attribute graphs are symmetric, the notations for f_G in the following subsections is generic (i.e., we omit the superscript U and I) that can apply to both user and item attribute graphs.

4.3.1 Neural Interaction based Message Passing. In our model, the message passing method models the inner interactions for characteristic learning. Inspired by [3, 30], we use an MLP to model each inner interactions. Then, we aggregate the interaction modeling results as the message passing information. Specifically, an MLP function $f_{neural} \in \mathbb{R}^{2 \times d} \rightarrow \mathbb{R}^d$, takes the embeddings of the two nodes as input and output the interaction modeling results:

$$\boldsymbol{z}_{ij} = f_{neural}(\boldsymbol{u}_i, \boldsymbol{u}_j), \tag{2}$$

where $\boldsymbol{z}_{ij}$ is the interaction modeling results of the node pair (i, j).

Then, all the interaction modeling results corresponding to each node will be aggregated as the message passing information. In GMCF, we use the element-wise sum to aggregate the interaction modeling results $\boldsymbol{z}_i = \sum_{j \in N_i} \boldsymbol{z}_{ij}$, where $\boldsymbol{z}_i \in \mathbb{R}^d$ is the message passing results of node i and N_i is a set of node i's neighbors.

f_{neural} explicitly models the interaction information between two attributes, which effectively unifies the interaction modeling for recommendation and the message passing in graph learning. In addition, inner interactions are used for capturing user (item) characteristics and are inherently complex. A high inner interaction result does not mean that the two attributes should be similar when determining the characteristics of a user (an item), which are different from cross interaction model results that reveal similarity (will be discussed in the next section). Therefore, a neural method (e.g., MLP) that non-linearly models the two attributes are desired.

4.3.2 Bi-Interaction based Node Matching. GMCF conducts node matching between two graphs through modeling cross interactions. Intuitively, we expect that an attribute c_i^U will have a high matching score with an attribute c_j^I if c_i^U shows a high preference on c_j^I. For example, if male users prefer sci-fi movies, the node matching score of the node pairs <*male*, *sci-fi*> should be high. In collaborative filtering, if a user attribute has a high preference for an item attribute, their embeddings should be similar after training. To achieve this, we use the Bi-interaction [13] for node matching, which keeps the monotonically increasing correlation between interaction modeling results and the attribute similarities. As a result, if a user attribute has a high matching score on an item attribute, they have similar attribute representations. Specifically, the Bi-interaction algorithm models the attribute interactions as:

$$\boldsymbol{s}_{ij} = \boldsymbol{u}_i \odot \hat{\boldsymbol{u}}_j, \tag{3}$$

where $\boldsymbol{u}_i$ is the embedding of node i in one graph, $\hat{\boldsymbol{u}}_j$ is the embedding of node j in the other graph, $\boldsymbol{s}_{ij}$ is the node matching result of two nodes from different graphs, and $\odot$ is the element-wise product.

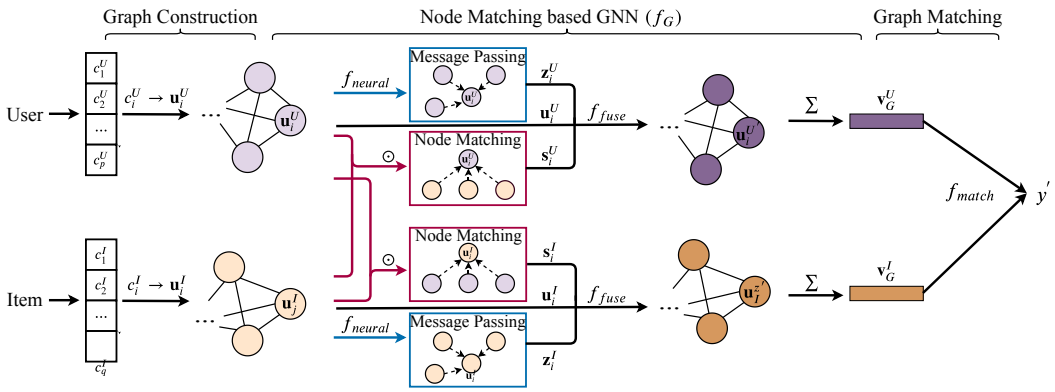

Figure 2: An Overview of the GMCF Model.

Similar to the message passing, we use an element-wise sum to aggregate the node matching result between a node in one graph and all nodes in the other graph. The aggregated node matching result is $\boldsymbol{s}_i = \sum_{j \in \hat{V}} \boldsymbol{s}_{ij}$, where $\hat{V}$ is the node set of the other attribute graph and $\boldsymbol{s}_i$ is the aggregated node matching result of node i.

4.3.3 Information Fusing. Besides message passing results, GMCF further considers the node matching results to capture the node-level matching information while generating the fused node representations. Specifically, the node fusing function $f_{fuse} \in \mathbb{R}^{3\times d} \rightarrow \mathbb{R}^d$ takes the initial node representation $\boldsymbol{u}_i$, the message passing results $\boldsymbol{z}_i$ and the node matching results $\boldsymbol{s}_i$ as input, and get the fused node representation. Formally, we have $\boldsymbol{u}_i^{'} = f_{fuse}(\boldsymbol{u}_i, \boldsymbol{z}_i, \boldsymbol{s}_i)$, where $\boldsymbol{u}_i^{'}$ is the fused node representation of node i.

We can use any method for f_{fuse}, e.g., element-wise addition or recurrent neural network. Through testing, we find that the recurrent neural networks perform the best. We use GRU, an effective recurrent neural network model, as the function f_{fuse}. Therefore, f_{fuse} considers $[\boldsymbol{u}_i, \boldsymbol{z}_i, \boldsymbol{s}_i]$ as the input sequence and the final output hidden layer of GRU is the fused node representation.

4.3.4 Node Representation Aggregation. The fused node representations of each graph are aggregated as the graph representation. We use the element-wise sum to aggregate the node representations. In summary, the f_G in GMCF is:

$$f_G(G, \hat{V}) = \sum_{i \in V} \boldsymbol{u}_i^{'}, \tag{4}$$

4.4 Graph Matching

While performing graph matching, we get the vector representations of the user attribute graph and the item attribute graph through f_G, respectively:

$$\boldsymbol{v}_G^U = f_G(G^U, V^I), \quad \boldsymbol{v}_G^I = f_G(G^I, V^U). \tag{5}$$

We match the two graphs by using dot product as f_{match} on the two graph representations to get the predicted output $y^{'} = \boldsymbol{v}_G^{U\top} \boldsymbol{v}_G^I$.

4.5 Model Training

While training, we use the L_2 norm to regularize all the parameters of GMCF. Therefore, the empirical risk minimization function of GMCF minimizes a loss function and an L_2 norm:

$$\begin{aligned} \mathcal{R}(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(F_{GMCF}(\boldsymbol{x_n}; \boldsymbol{\theta}), y_n) + \lambda(\|\boldsymbol{\theta}\|_2), \\ \boldsymbol{\theta}^* &= arg\, min_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta}), \end{aligned} \tag{6}$$

where F_{GMCF} is the prediction function of GMCF that outputs $y^{'}$, $\mathcal{L}(\cdot)$ corresponds to a loss function (e.g., binary cross-entropy loss), $\boldsymbol{\theta}$ are all parameters in GMCF, and $\boldsymbol{\theta}^*$ are the final parameters.

4.6 Relation to Existing Work and Discussion

Our model has close relations to attribute graph-based recommender systems and FM. In this section, we discuss the relations to the two types of recommender systems. Then, we show that our model can apply to situations when the user or item attributes are not available, which happens in practice.

4.6.1 Relation to Attribute Graph-based Recommender Systems. Fi-GNN [21] and L_0-SIGN [30] are two attribute graph-based models that treat all attribute interactions equally in one graph. Our model can be considered an extension of this framework. Specifically, if GMCF considers all interactions equally, e.g., all are modeled as cross interactions, and puts the match function f_{match} linear, e.g., $f_{match}(\boldsymbol{v}_G^U, \boldsymbol{v}_G^I) = sum(\boldsymbol{v}_G^U) + sum(\boldsymbol{v}_G^I)$, where $sum(\cdot)$ sums up all the elements of the input vector. As a result, GMCF has a similar framework to Fi-GNN and L_0-SIGN.

4.6.2 Relation to FM. Different from graph-based models, FM aggregates all interaction modeling results linearly (i.e., sum up). GMCF can also be regarded as an extension of FM. Besides the modifications that reduce to the framework of Fi-GNN and L_0-SIGN (every interaction is modeled by element-wise product), if the fusing function f_{fuse} is linear, e.g., $f_{fuse}(\boldsymbol{u}_i, \boldsymbol{z}_i, \boldsymbol{s}_i) = \boldsymbol{u}_i + \frac{1}{2} \sum_{j \in V^a/\{i\}} \boldsymbol{s}_{ij}$, where $V^a = V^U \cup V^I$, $\boldsymbol{s}_{ij} = \boldsymbol{u}_i \odot \boldsymbol{u}_j$, the prediction function becomes:

$$\begin{aligned} y^{'} &= sum(\sum_{i \in V^U} (\boldsymbol{u}_i + \frac{1}{2} \sum_{j \in V^a/\{i\}} \boldsymbol{s}_{ij})) + sum(\sum_{i \in V^I} (\boldsymbol{u}_i + \frac{1}{2} \sum_{j \in V^a/\{i\}} \boldsymbol{s}_{ij})) \\ &= \sum_{i \in V^a} sum(\boldsymbol{v}_i) \cdot val_i + \sum_{i \in V^a} \sum_{j \in V^a, j > i} \boldsymbol{v}_i^\top \boldsymbol{v}_j \cdot val_i \cdot val_j. \end{aligned} \tag{7}$$

Equation 7 is similar to the prediction function of FM in its original paper (Equation 1 in [25]) with the weight parameter ω_i of

each attribute i being $sum(\boldsymbol{v}_i)$ and the bias term ω_0 being omitted (Note that $sum(\boldsymbol{u}_i \odot \boldsymbol{u}_j) = \boldsymbol{v}_i^\top \boldsymbol{v}_j \cdot val_i \cdot val_j$).

4.6.3 When the User or Item Attributes are Unavailable. In some situations, the user or item attributes are not available. GMCF is also applicable in these situations. Specifically, if one type of attribute is unavailable, e.g., user attributes, the user attribute graph becomes a single node (user ID). Then, no inner interaction modeling is in the user attribute graph, i.e., $\boldsymbol{v}_G^U = \boldsymbol{u}_u^{'} = f_{fuse}(\boldsymbol{u}_u, \boldsymbol{s}_u)$, where $\boldsymbol{u}_u$ is the node representation of user ID. If both user and item attributes are unavailable, both the two attribute graphs are represented by a single node. Then, there is no inner interaction modeling but only the cross interaction modeling between user ID and item ID. We will evaluate our model in these scenarios in Section 5.3.3.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate GMCF. We focus on three questions: (i) what is the ability of GMCF in providing accurate recommendations comparing to other baselines that can consider user attributes and item attributes; (ii) what is the effectiveness of each component in GMCF on providing accurate predictions; (iii) whether the learned attribute embeddings reveal collaborative relations and semantic meaning in their latent space.

5.1 Experimental Protocol

We first describe the datasets and the baselines used in our experiments. Then, we illustrate the experimental set-up in detail.

5.1.1 Datasets. We run GMCF and baselines on three datasets, that contain both user and item attributes. Table 1 shows their statistic information. Below are the descriptions of the datasets:

MovieLens 1M [12] contains users' ratings on movies. Each data sample contains a user and a movie with their corresponding attributes. We further collect movies' other attributes, such as directors and casts from IMDB to enrich the datasets. **Book-crossing** [45] contains users' implicit and explicit ratings of books. Each data sample contains a user and a book with their corresponding attributes. The reprocessed words in the book titles are also regarded as attributes of the book. **Taobao** [44] is a dataset that collects the log of click on display advertisement displayed on the website of Taobao. Each log contains a user with corresponding attributes such as gender and age level, a displayed advertisement with attributes such as category and brand of the advertised item.

MovieLens 1M and Book-crossing contain explicit ratings. We transfer the explicit ratings to implicit feedback. We regard the ratings greater than 3 as positive ratings for MovieLens 1M and regard all rated explicit ratings as positive ratings for Book-crossing due to its sparsity. Then, we randomly select the same number of negative samples equal to the number of positive samples for each user. To ensure the datasets' quality, we select the users with more than 10 positive ratings for MovieLens 1M and have more than 20 positive ratings for Book-crossing and Taobao.

5.1.2 Baselines. We compare our model with competitive baselines that can take user attributes and item attributes into account:

FM [25] models every feature interaction by dot product and sums up all the modeling results as the final prediction result. **AFM** [39] additionally calculates an attention value for each interaction in FM as the weight when performing the aggregation. **NFM** [13] leverages an MLP on top of the aggregated interaction modeling results of FM so that as to non-linearly analyze the feature interactions. **W&D** [7] combines a linear model that with a deep neural network for recommendation. The input of the deep neural network is the concatenation of all feature embeddings. **DeepFM** [11] combines interaction analysis results from using MLP and FM for prediction. Specifically, in the MLP part, all attribute embeddings are concatenated together and then fed into an MLP to learn their interactions. **AutoInt** [28] explicitly models all feature interactions using a multi-head self-attentive neural network and then aggregates all the modeling results as the final prediction result. **Fi-GNN** [21] represents each data sample as a feature graph that each node is a feature field. Then, it models the interactions of the field in a GNN using the multi-head self-attention method. L_0**-SIGN** [30] considers each data sample as a graph, with the user, the item, and all corresponding features as nodes. L_0-SIGN simultaneously detects beneficial interactions and leverages the detected ones as edges for graph classification, and the classification results are the prediction results. For all baselines, we set the MLP structure (if used) for interaction modeling to be the same as the MLP for inner interaction modeling in our model for a fair comparison.

Table 1: Dataset statistics. The *attr.* refers to "attributes".

Dataset	#Data	#User	#Item	#User *attr.*	#Item *attr.*
MovieLens 1M	1,149,238	5,950	3,514	30	6,944
Book-Crossing	1,050,834	4,873	53,168	87	43,157
Taobao	2,599,463	4,532	371,760	36	434,254

5.1.3 Experimental Set-Up. We randomly split each dataset into training, validation, and test set for each user with a ratio of 6:2:2. The validation set is only used to decide the best parameter setting, and the test set is only used to evaluate the models. Unless otherwise specified, we use the following hyper-parameter settings: the node representation dimension is 64 (i.e., $d = 64$); the number of hidden layers for MLP is 1 and the number of units for the hidden layer is $4d$; the learning rate is 1×10^{-3}; the λ for the regularization is 1×10^{-5}. We use the binary cross entropy as the loss function and use Adam [17] as the optimization algorithm. We use the area under the curve (AUC), Logloss, and NDCG@k as evaluation metrics to evaluate the performance of our model and baseline models. AUC and Logloss are frequently used in the tasks with implicit feedback and NDCG@k is a frequently used metric to evaluate the top-k recommendation. We set k to 5 and 10. All the experiments are repeated 5 times and the average results are recorded.

5.2 Overall Performance

We compare the performance of GMCF with the baselines. Table 2 shows the prediction performance of each model. The best results for each dataset are in bold, and the best baseline results are in underline. The rows *Improv* and *p-value* show the improvement and statistical significance test results (through Wilcoxon signed-rank test) of GMCF and the best baseline results, respectively. From Table 2, we observe that:

- GMCF outperforms all the baselines significantly, with the p-value of all metrics rejecting the null hypothesis with a level of

Table 2: Summary of the performance in comparison with baselines.

	MovieLens 1M				Book-Crossing				Taobao			
	AUC	Logloss	NDCG@5	NDCG@10	AUC	Logloss	NDCG@5	NDCG@10	AUC	Logloss	NDCG@5	NDCG@10
FM	0.8761	0.4409	0.8143	0.8431	0.7417	0.5771	0.7616	0.8029	0.6171	0.2375	0.0812	0.1120
AFM	0.8837	0.4323	0.8270	0.8676	0.7541	0.5686	0.7820	0.8258	0.6282	0.2205	0.0872	0.1240
NFM	0.8985	0.3996	0.8486	0.8832	0.7988	0.5432	0.7989	0.8326	0.6550	0.2122	0.0997	0.1251
W&D	0.9043	0.3878	0.8538	0.8869	0.8105	0.5366	0.8048	0.8381	0.6531	0.2124	0.0959	0.1242
DeepFM	0.9049	0.3856	0.8510	0.8848	0.8127	0.5379	0.8088	0.8400	0.6550	0.2115	0.0974	0.1243
AutoInt	0.9034	0.3883	0.8619	0.8931	0.8130	0.5355	0.8127	0.8472	0.6434	0.2146	0.0924	0.1206
Fi-GNN	0.9063	0.3871	0.8705	0.9029	0.8136	0.5338	0.8094	0.8522	0.6462	0.2131	0.0986	0.1241
L_0-SIGN	0.9072	0.3846	0.8849	0.9094	0.8163	0.5274	0.8148	0.8629	0.6547	0.2124	0.1006	0.1259
GMCF	**0.9179**	**0.3726**	**0.9372**	**0.9484**	**0.8271**	**0.5219**	**0.8671**	**0.8951**	**0.6679**	**0.1960**	**0.1112**	**0.1467**
Improv	1.18%	3.12%	5.91%	4.28%	1.32%	1.04%	6.42%	3.73%	1.96%	7.32%	10.53%	16.52%
p-value	1.09%	0.25%	0.25%	0.25%	0.46%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%

significance of $\alpha = 5\%$. These results prove the ability of GMCF in effectively analyzing the structural information of user and item attributes for accurate predictions.

- The models that explicitly model attribute interactions (GMCF, AutoInt, Fi-GNN, L_0-SIGN) gain better prediction accuracy than other models. This indicates that the explicit interaction modeling is promising to extract useful information from attributes and attribute interactions for accurate predictions.
- FM and AFM perform worse than the other models. This is because FM and AFM do not use neural methods, but solely rely on the dot product to extract information from attribute interactions. Hence, to model more complicated interactions, sophisticated methods (e.g., MLP) are required. Accordingly, our model leverages MLP to model the inner interactions inside the user attribute graph and the item attribute graph, which provides powerful interaction modeling ability for accurate predictions.
- The GNN-based models (GMCF, Fi-GNN, L_0-SIGN) gain better prediction accuracies than other models. It shows the ability of GNNs in modeling attribute interactions for recommendation. GMCF further models attribute interactions in a structure of graph matching, which is more suitable for recommendation and gains better performance than Fi-GNN and L_0-SIGN.

5.3 Study of Neural Graph Matching

In this section, We evaluate the neural graph matching of GMCF. We focus on (i) the effectiveness of modeling inner and cross interactions using different methods; (ii) the comparison of using different algorithms as the fusing function; (iii) the performance of GMCF when the user and item attributes are not available. Due to space limitation, we omit the results of Logloss and NDCG@5, which show similar trends with AUC and NDCG@10, respectively.

5.3.1 Evaluation of Inner and Cross Interaction modeling. We evaluate the inner and cross interaction modeling methods. Specifically, we focus on three questions: 1) what is the effectiveness of node-level graph matching (i.e., modeling cross interactions)? 2) whether we should use different methods to model the two types of interactions? 3) whether sophisticated nonlinear algorithms are always suitable for modeling the two types of interactions?

To answer the three questions, we run GMCF with different interaction modeling methods. For clear demonstration, we use the pair <*inner interaction model, cross interaction model*> to indicate the method combinations in each variation. For example, we represent the original GMCF as <*MLP, Bi*> to indicate an MLP-based message passing and a Bi-interaction-based node matching.

We run GMCF using the combination <*MLP, None*>, where *None* indicates that we do not perform node-level matching (i.e., no cross interaction modeling). Then, we run two variations that use the same method to model all interactions: <*Bi, Bi*>, <*MLP, MLP*>. Note that the second variation uses the same MLP (i.e., the same neural architecture with shared parameters). Finally, we further run two variations that use different methods: <*MLP1, MLP2*> (the same architecture with different parameters) and <*Bi, MLP*>. Figure 3 shows the results of using different variations. We omit the results of Book-crossing due to the space limitation, which shows similar trend with MovieLens 1M (the same as the remaining figures).

For question 1), <*MLP, None*> gains the worst performance. The reason is that the attribute graphs are inherently complex. Fusing this information only in the last step makes it difficult to match users' preferences on items through attributes. The node matching method provides the explicit attribute communication through the cross interactions between the two graphs, which results in a fine-grained analysis for a more accurate preference matching. For question 2), the combinations (<*Bi, Bi*> and <*MLP, MLP*>) gain relatively worse results than the original setting (<*MLP, Bi*>). This is because the inner interactions and cross interactions are inherently different that should be modeled differently that fit their roles. The inner interactions are used for characteristic learning and do not indicate the similarity information (i.e., two attributes having strong interaction does not mean that they are similar). Meanwhile, in collaborative filtering, two attributes are expected to show similarity information if their cross interaction is strong (high preference). The Bi-interaction algorithm fits for the requirement of cross interactions. For question 3), even using different methods, <*MLP1, MLP2*> and <*Bi, MLP*> still gain worse results than the original GMCF. Although an MLP seems more powerful than the Bi-interaction algorithm, MLP cannot to effectively learn the measurement for the cross interactions. The results show that sophisticated algorithms such as MLP are expected to model the inner interactions, and similarity measurements such as the Bi-interaction algorithm are expected to model the cross interactions.

5.3.2 Evaluation of the Fusing Method. The fusing method in GMCF aggregates the message passing results and the node matching

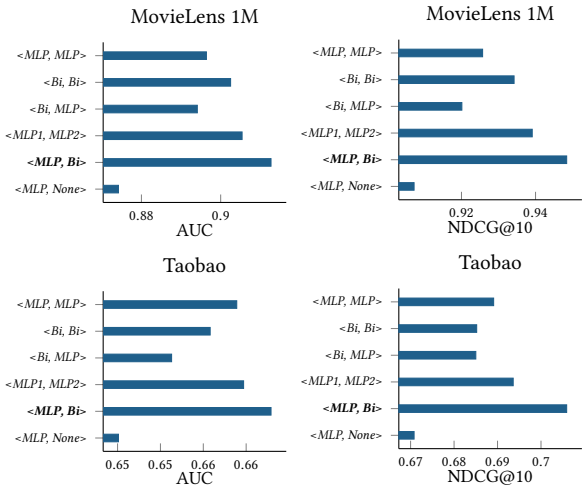

Figure 3: The comparison of using different inner interaction and cross interaction modeling combinations.

Table 3: The performance of using different fusing algorithms.

	MovieLens 1M		Book-Crossing		Taobao	
	AUC	NDCG@10	AUC	NDCG@10	AUC	NDCG@10
SUM	0.9042	0.9404	0.8190	0.5231	0.6583	0.1383
MLP	0.9022	0.9389	0.8156	0.5207	0.6636	0.1421
GRU	**0.9127**	**0.9484**	**0.8211**	**0.5269**	**0.6679**	**0.1467**

results to get the fused node representation. We evaluate the effectiveness of using different algorithms as the fusing algorithm. Except the GRU, we further use the element-wise sum on the three vectors as the fused node representation (SUM) and the MLP that concatenates the vectors as input and outputs the fused node representation (MLP). We use the MLP that has one hidden layer, with the number of neurons being $4d$.

Table 3 shows the experimental results of using the three algorithms as the fusing method. From the table, we can see that using GRU results in the best performance on all datasets. It shows the ability of GRU to effectively aggregate the message passing information and the node matching information into the fused node representation for accurate predictions. Summing up the results (*SUM*) gains the worst results in most situations, which indicates that the message passing and node matching information are complex. Powerful algorithms are required to fuse them.

5.3.3 Evaluation of the User and Item Attributes. In some situations, user or item attributes are not available. We evaluate how GMCF performs in these situations. GMCF is a flexible framework that is applicable when the user or item attributes are not available. In these situations, the user (or item) graphs are reduced into a single node indicating user (or the item) ID. We run GMCF and the best baseline L_0-SIGN on the situations that user or item attributes are not available. Figure 4 shows the results. Specifically, in the x-axis, "*None*" indicates neither user or item attributes are available, "*User*" indicates only user attributes are available, "*Item*" indicates only item attributes are available, and "*Both*" indicates both user and item attributes are available.

From the figure, we observe that: 1) Both models perform better when user and item attributes become available (from left to right).

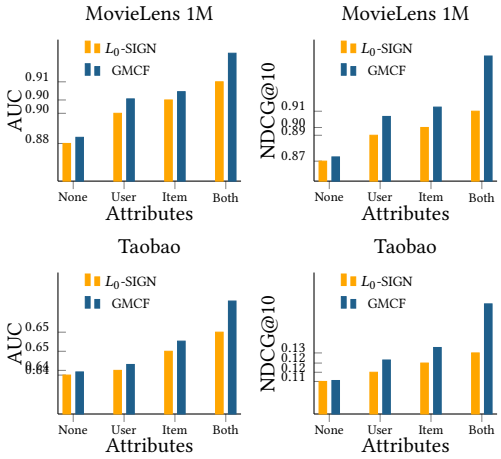

Figure 4: The comparison when using different attributes.

It shows that both user and item attributes are useful for performance gain. Although the performance gain from user attributes (*User*) seems not that significant compared to item attributes (*Item*), the user attributes provide useful information for potential explanations of the prediction results, which will be discussed in Section 5.5. 2) The two models have similar performance when no attributes are available (*None*). However, GMCF performs much better when user and item attributes are available. The performance gain compared to L_0-SIGN when the user and item attributes are available resulted from the generated graph matching structure, which captures more useful structural information for accurate predictions.

5.4 Parameter study

In this section, we evaluate GMCF with different hyper-parameter settings. Specifically, we evaluate the node representations' dimension and the depth of the MLP used in our model.

Figure 6 shows the results of our model and best-performed baselines on different node representation dimensions (d). From the figure, we observe that GMCF constantly outperforms baselines on different node representation dimensions, which shows the robustness of our model in delivering superior prediction accuracy. Then, when the dimension is 64, our model and most of the baselines gain the best performance. This indicates that a highger dimension does not necessarily result in better prediction accuracy. This is because that a larger dimension means more parameters to fit, and thus is prone to cause the overfitting problem.

Then, we evaluate how the different number of hidden layers in the MLP affects our model's performance. In our model, we use an MLP with 1 hidden layer to analyze the inner interactions while message passing. Now we evaluate our model with different number of layers. Specifically, we run our model with 0, 1, 2, 3, 4 hidden layers respectively (GMCF-0,...,GMCF-4). Note that GMCF-0 means that the MLP only performs a linear transformation from the node representations to the interaction modeling results. We use the same number of units ($4d$) for each hidden layer.

Table 4 shows the results of using different MLP layers in GMCF. We can see that using the MLP with 0 hidden layer (GMCF-0) gains much worse results than other settings. It shows that a powerful non-linear algorithm helps extract useful information from inner

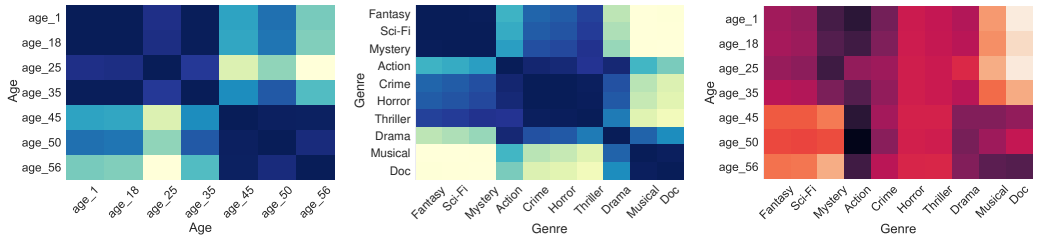

Figure 5: The case studies on the MovieLens 1M dataset. *Left*: the embedding similarities between user age groups. *Central*: the embedding similarities between movie genres. *Right*: the node matching (preference) results between the ages groups and the genres.

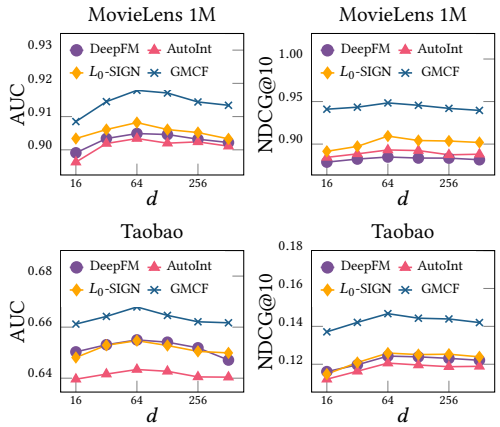

Figure 6: The performance of GMCF and selected baselines in different node (attribute) embedding dimensions.

Table 4: The performance of GMCF on using different MLP depths.

	MovieLens 1M		Book-Crossing		Taobao	
	AUC	NDCG@10	AUC	NDCG@10	AUC	NDCG@10
GMCF-0	0.9025	0.9342	0.8051	0.5117	0.6608	0.1341
GMCF-1	**0.9127**	**0.9484**	**0.8211**	**0.5269**	**0.6679**	**0.1467**
GMCF-2	0.9109	0.9452	0.8182	0.5235	0.6662	0.1429
GMCF-3	0.9080	0.9419	0.8178	0.5223	0.6645	0.1395
GMCF-4	0.9071	0.9379	0.8153	0.5197	0.6650	0.1393

interactions than a linear transformation. This result is consistent with the results in section 5.3.1. When the number of the hidden layer is 1, GMCF gains the best performance. This illustrates that deeper MLP does not necessarily increase the performance due to the overfitting [11, 13], and one hidden layer is enough in our models to analyze the inner interactions.

5.5 Case Study

In this section, we conduct case studies to evaluate whether our model learns collaborative information between attributes and whether the attributes show semantic meaning that provide potential explanations of the predictions. Specifically, we use the learned attribute embeddings from the MovieLens 1M dataset. We first calculate the embeddings' cosine similarities between user age groups (e.g., 1-18, 19-24) and between movie genres. Then, we calculate the node matching results between the the two types of attributes. Figure 5 shows the similarity results (the left and central figures) and the node matching results (the right figure). Note that the darker the color, the higher the similarity or the node matching value. The age labels indicate the age groups (e.g., *age_18* means age 18-24).

From the left and central figures, we can see that similar user attributes and similar item attributes have similar embeddings after training. For example, in the left figure, the age group attributes are clearly divided into two groups at the age of 45, which means that the similar age attributes are grouped. It indicates that the same age group users may have a similar preference for movies in terms of age. In the central figure, Fantasy, Sci-fi, and Mystery have similar embeddings in the latent space. Semantically, these genres are similar. Similar results are also shown for Crime, Horror, and Thriller. The above observations show that similar attributes are successfully learned to have similar embeddings in GMCF.

The right figure shows the node matching between age groups and genres. We observe that different age groups have different preferences for movie genres. For example, younger users (under 45) have higher node matching values (preference) on fantasy, sci-fi, and mystery movies more than older users (over 45). In comparison, the older users seem to like musical and Documentary (*Doc*) movies more. These observations show our model's ability to provide potential explanations about the prediction results at attribute level, e.g., we recommend *Interstellar* (a sci-fi movie) to a 16-year-old user because younger people have a high chance to prefer a sci-fi movie.

6 CONCLUSION

User and item attribute interactions provide useful information in recommender systems for accurate predictions. While existing work treat all the attribute interactions equally, in this work, we identify two types of attribute interactions: inner interactions and cross interactions. We propose a neural Graph Matching based Collaborative Filtering (GMCF) model. The GMCF models and exploits the two types of interactions for different purpose in a graph matching structure. Specifically, GMCF explicitly performs user and item characteristic learning with the inner interactions, and performs preference matching for recommendation based on the cross interactions. Experimental results show that our GMCF model is effective and outperforms all state-of-the-art baselines in terms of accuracy on three widely used datasets. In future work, we will consider higher-order interactions in our model architecture, which may contain edge matching and sub-graph exploration.

ACKNOWLEDGMENTS

This work is supported by the China Scholarship Council (CSC). In this research, Junhao Gan was in part supported by Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA) DE190101118.

REFERENCES

[1] Ryan Prescott Adams, George E Dahl, and Iain Murray. 2010. Incorporating Side Information in Probabilistic Matrix Factorization with Gaussian Processes. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*. 1–9.

[2] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. 2019. SimGNN: A Neural Network Approach to Fast Graph Similarity Computation. In *Proceedings of the 35th International Conference on Web Search and Data Mining (WSDM)*. 384–392.

[3] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. 2016. Interaction Networks for Learning about Objects, Relations and Physics. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. 4502–4510.

[4] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*. 46–54.

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. 737–744.

[6] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. 2016. A Compositional Object-based Approach to Learning Physical Dynamics. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 1–15.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (RecSys)*. 7–10.

[8] Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. 2009. Graph Matching Algorithms for Business Process Model Similarity Search. In *Proceedings of the 7th International Conference on Business Process Management (BPM)*. Springer, 48–63.

[9] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. 2224–2232.

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 1263–1272.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a Factorization-Machine based Neural Network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 1725–1731.

[12] F Maxwell Harper and Joseph A Konstan. 2015. The Movielens Datasets: History and Context. *Transactions on Interactive Intelligent Systems (TIIS)* (2015), 1–19.

[13] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th international conference on world wide web (WWW)*. 173–182.

[15] Xinting Huang, Jianzhong Qi, Yu Sun, Rui Zhang, and Hai-Tao Zheng. 2019. CARL: Aggregated Search with Context-Aware Module Embedding Learning. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 101–108.

[16] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*. 321–328.

[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*. 1–15.

[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 1–14.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[20] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 3835–3845.

[21] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *Proceedings of the 28th International Conference on Information and Knowledge Management (CIKM)*. 539–548.

[22] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *Proceedings of the 26th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2636–2645.

[23] Yunsheng Pang, Yunxiang Zhao, and Dongsheng Li. 2021. Graph Pooling via Coarsened Graph Infomax. *arXiv preprint arXiv:2105.01275* (2021).

[24] John W Raymond, Eleanor J Gardiner, and Peter Willett. 2002. Rascal: Calculation of Graph Similarity Using Maximum Common Edge Subgraphs. *Comput. J.* 45, 6 (2002), 631–644.

[25] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 10th International IEEE Conference on Data Mining (ICDM)*. 995–1000.

[26] Nino Shervashidze and Karsten Borgwardt. 2009. Fast Subtree Kernels on Graphs. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. 1660–1668.

[27] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 488–495.

[28] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic Feature Interaction Learning via Self-attentive Neural Networks. In *Proceedings of the 28th International Conference on Information and Knowledge Management (CIKM)*. 1161–1170.

[29] Yixin Su, Sarah Monazam Erfani, and Rui Zhang. 2019. MMF: Attribute Interpretable Collaborative Filtering. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[30] Yixin Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. 2021. Detecting Beneficial Feature Interactions for Recommender Systems. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

[31] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. arXiv:1706.02263 [stat.ML]

[32] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph Kernels. *The Journal of Machine Learning Research (JMLR)* 11 (2010), 1201–1242.

[33] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-Aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 968–977.

[34] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 950–958.

[35] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International conference on Research and Development in Information Retrieval (SIGIR)*. 165–174.

[36] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2021. 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)*. 427–435.

[37] Peter Willett, John M Barnard, and Geoffrey M Downs. 1998. Chemical Similarity Searching. *Chemical Information and Computer Sciences* (1998), 983–996.

[38] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *Proceedings of the 42nd International Conference on Research and Development in Information Retrieval (SIGIR)*. 285–294.

[39] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 3119–3125.

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. 1–17.

[41] Xifeng Yan, Philip S Yu, and Jiawei Han. 2005. Substructure Similarity Search in Graph Databases. In *Proceedings of the 2005 International Conference on Management of Data (SIGMOD)*. 766–777.

[42] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge base Embedding for Recommender Systems. In *Proceedings of the 22nd international conference on knowledge discovery and data mining (SIGKDD)*. 353–362.

[43] Yunxiang Zhao, Jianzhong Qi, Qingwei Liu, and Rui Zhang. 2021. WGCN: Graph Convolutional Networks with Weighted Structural Features. *arXiv preprint arXiv:2104.14060* (2021).

[44] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-through Rate Prediction. In *Proceedings of the 24th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 1059–1068.

[45] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International Conference on World Wide Web (WWW)*. 22–32.