Learning Item Attributes and User Interests for Knowledge Graph Enhanced Recommendation

Anonymous Author(s)

ABSTRACT

Knowledge Graphs (KGs) manifest great potential in recommendation. This is ascribable to the rich attribute information contained in KG, such as the price attribute of goods, which is further integrated into item and user representations and improves recommendation performance as side information. However, existing knowledge-aware methods leverage attribute information at a coarse-grained level in two aspects: (1) item representations don't accurately learn the distributional characteristics of different attributes, and (2) user representations don't sufficiently recognize the pattern of user preferences towards attributes. In this paper, we propose a novel *attentive knowledge graph attribute network*(AKGAN) to learn item attributes and user interests via attribute information in KG. Technically, AKGAN adopts a novel graph neural network framework, which has a different design between the first layer and the latter layer. The first layer merges one-hop neighbors' attribute information by concatenation operation to avoid breaking down the independence of different attributes, and the latter layer recursively propagates attribute information without weight decrease of high-order significant neighbors. With one attribute placed in the corresponding range of element-wise positions, AKGAN employs a novel interest-aware attention unit, which releases the limitation that the sum of attention weight is 1, to model the complexity and personality of user interests. Experimental results on three benchmark datasets show that AKGAN achieves significant improvements over the state-of-the-art methods like KGAT [33], KGNN-LS [31], and KGIN [35]. Further analyses show that AKGAN offers interpretable explanations for user preferences towards attributes.

CCS CONCEPTS

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

KEYWORDS

Recommendation, Knowledge Graph, Graph Neural Network

ACM Reference Format:

Anonymous Author(s). 2022. Learning Item Attributes and User Interests for Knowledge Graph Enhanced Recommendation. In *Proceedings of xxx*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

xxx, xxx, xxx

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

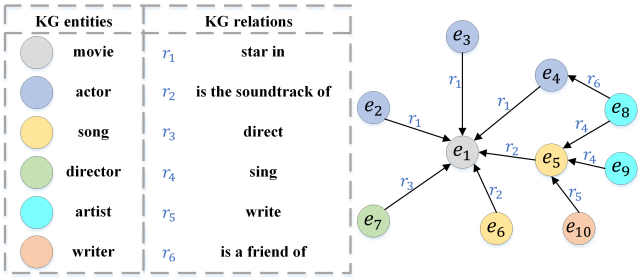

Figure 1: An example of how KG contains multiple attributes information. Best viewed in color.

1 INTRODUCTION

Recommender systems have shown great success in e-commerce, online advertisement, and social medial platforms. The core task of recommender systems is to solve the overload information problem [31, 32, 40] and suggest items that users are potentially interested in. Early methods [7] are content-based and collaborative filtering (CF)-based recommender systems [9, 20, 34]. However, both of them don't introduce much side information.

In recent years, introducing knowledge graphs (KGs) into recommender systems as side information has been effective for improving recommendation performance. KGs represent real-world entities and illustrate the relationship between them with graph data structure, which can reveal multiple attributes of items and explore the potential reason for user-item interactions. Generally, a KG contains item nodes and attribute nodes, and attribute nodes can not only describe items' attributes directly but also represent other attribute nodes' attributes. For example, a movie knowledge graph (as shown in Figure 1) has six types of nodes, where movie node (i.e., e_1) is item node and the others (i.e., e_{2-10}) are attribute nodes. e_{2-4} represent the actor attribute of movie e_1, which means actors e_{2-4} star in movie e_1. And $e_{8,9}$ represent the singer attribute of song e_5, which means artists $e_{8,9}$ perform the song e_5. To integrate attribute information into the recommender system, some works focus on embedding-based methods [1, 3, 12, 45, 46] and path-based methods [19, 27, 41–43]. However, both of them don't capture high-order connectivities and fail to exploit both the rich semantics and topology of KGs. More recently, the propagation-based methods, a.k.a. graph neural network (GNN)-based methods such as KGAT [33], KGNN-LS [31], KNI [23], AKGE [26], KGIN [35], have attracted considerable interest of researchers. With the attribute information iteratively propagating in KGs, GNN-based methods aggregate multi-hop neighbors into representations and have achieved the state-of-the-art recommendation results.

Despite the success of GNN in exploiting multi-hop attribute information, we argue that there are still three shortcomings: (1)**The**

pollution caused by weighted sum operation in merging different attribute information. Different attributes are independent in terms of semantics and user preference. Take the actor attribute e_2 and the singer attribute e_8 of movie e_1 in figure 1 as an example, semantic independence means e_2 doesn't co-occur with e_8 in each movie, since an actor and a singer are invited to work for a movie respectively. Preference independence means whether a user prefers actor e_2 is independent of whether he likes singer e_8. However, existing GNN-based methods pool embeddings from different attribute nodes with weighted sum (i.e., sum, mean, attention) operation, which brings about semantic pollution and the difficulty to distill user-interested attribute information. (2) **The nonlinearity between the distance and importance of attribute node relative to item node**. Usually, high-order neighbors are less relative to the center node in graph data, but this is not absolute. For example, singer attribute e_8 is more valued than director attribute e_7 by some movie viewers who like music, while e_8, a 2-hop neighbor, is further than e_7, a 1-hop neighbor. This problem is caused by inherent graph topology, which means some significant attribute nodes don't connect to item nodes directly, such that it requires multiple passes to aggregate these high-order neighbors into the center node. However, existing GNN-based algorithms neglect this issue and decrease the weight of significant high-order neighbors coupled with the increase of propagation times. (3)**The complexity and personality of user interests towards attributes**. User interests show the following pattern: a user just gets interested in a part of rather than all attributes of an item, and different users prefer different attributes even towards the same item. For example, both users u_1 and u_2 watch movie e_1 because u_1 likes e_1's actor e_2 rather than director e_7 while u_2 prefers the theme song e_5 rather than actor e_2. Therefore, we should integrate e_2's rather than e_7's embedding into u_1's representation and integrate e_5's rather than e_2's embedding into u_2's representation. In other words, the item representation learned after GNN propagation contains noisy signals and we should distill user-interested attributes personally. However, existing GNN-based algorithms recognize this pattern insufficiently, like [35] doesn't consider noisy attributes and [33] doesn't consider personal extraction.

To address the foregoing problems, we propose a novel *attentive knowledge graph attribute network* (AKGAN), which consists of two components: (1) **knowledge graph attribute network (KGAN)**. The core task of KGAN is to learn informative item representations without semantic pollution and weight decrease of significant attribute nodes. Technologically, KGAN has a different design between the first layer and the latter layer under GNN framework. The first layer, called attribute modeling layer, aims to generate initial item representations without semantic pollution. It regards each relation in KG as an attribute and has two key designs: embedding each entity in different attribute spaces and using concatenation operation to merge different attributes. The latter layer, called attribute propagation layer, aims to remain semantics unpolluted and avoid the weight decrease of significant neighbors after GNN propagation. Finally, KGAN generates item representations where one attribute is represented within a specific range of element-wise positions independently. (2) **user interest-aware attention network**. With different attributes placed in corresponding element-wise positions, we design an interest-aware attention layer to distill user-interested attributes. Specifically, for a particular user, we pool the representations of his interacted items and extract each attribute representation according to corresponding element-wise positions. Then each attribute representation will be fed into an attention module to calculate a personal interest score, which describes how much he prefers this attribute. A novel activation unit is introduced to release the limitation that the sum of attention weight is 1, which aims to reserve the intensity of user interests [50]. Finally all interest scores and item representations are further combined to infer user representations.

To summarize, the main contributions of this work are as follows:

- On the item side, we propose a novel knowledge graph attribute network with heterogeneous layers to improve two common shortcomings of current GNN-based methods: (1) prevent semantic pollution and maintain the independence of attributes; (2) avoid weight decrease of significant high-order neighbors.
- On the user side, we develop an interest-aware attention network, which introduces a novel activation unit and releases the limitation that the sum of attention weight is 1 , to model user interests towards attributes personally.
- We conduct extensive experiments on three public benchmarks to demonstrate the superior performance of AKGAN over state-of-the-art baselines. In-depth analyses are provided to illustrate the interpretability of AKGAN for user personal interests.

2 PROBLEM FORMALIZATION

We begin by introducing some related notations and then defining the KG enhanced recommendation problem. Let $\mathcal{U} = \{u\}$ and $\mathcal{I} = \{i\}$ separately denote the user and item sets. A typical recommender system usually has historical user-item interactions, which is defined as $\mathcal{O}^+ = \{(u, i)|u \in \mathcal{U}, i \in \mathcal{I}\}$. Each (u, i) pair indicates user u has interacted with item i before, such as clicking, review, or purchasing. We also have a knowledge graph that stores the structured semantic information of real-world facts. We denote the entity and relation sets in KG as $\mathcal{R} = \{r\}$ and $\mathcal{E} = \{e\}$. KG is presented as $\mathcal{G} = \{(h, r, t)|h \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{E}\}\}$, where each triple means there is a relation r from head entity h to tail entity t. For example, $(James\ Cameron, Direct, Avatar)$ describes the fact that James Cameron is the director of the movie Avatar. Note that $\mathcal{R}$ contains relations in both canonical direction (e.g., Direct) and inverse direction (e.g., DirectedBy). The bridge between KG and the recommender system is that items are contained in entities. Specifically, $\mathcal{E}$ consists of item node $\mathcal{I}(\mathcal{I} \subseteq \mathcal{E})$ as well as attribute node $\mathcal{E} \setminus \mathcal{I}$, which can futher refine user representation $\mathbf{e}_u$ and item representation $\mathbf{e}_i$.

We now formulate the KG enhanced recommendation task in this paper:

- **Input**: a knowledge graph $\mathcal{G}$, which contains rich structure semantic information of items, and the user-item interaction data $\mathcal{O}^+$
- **Output**: a scoring function that denotes the probability that user u will interact with item i.

The key symbols used in this paper are listed in Table 1.

Table 1: List of key symbols

Symbol	Meaning
$\mathcal{U} = \{u\}$	Set of users
$\mathcal{I} = \{i\}$	Set of items
$O^+ = \{(u,i)\|u \in \mathcal{U}, i \in \mathcal{I}\}$	User-item interaction data
$\mathcal{E} = \{e\}$	Set of entities
$\mathcal{R} = \{r\}$	Set of relations
$\mathcal{E} \setminus \mathcal{I}$	Set of attribute entities
$\mathcal{G}$	Knowledge graph
$\mathbf{e}_i^r$	Entity attribute embedding
$\mathbf{e}_{i'}^r$	Entity truncated vector in $\mathbf{e}_i^{(0)}$
$\mathbf{e}_{i''}^r$	Entity truncated vector in $\mathbf{e}_i^*$
$\mathbf{e}_i^{(l)}$	Entity representation in the l−th layer
$\mathbf{e}_i^*$	Final item representation
$\mathbf{e}_u^*$	Final user representation

3 METHODOLOGY

3.1 Model Overview

In this subsection, we introduce the framework of AKGAN. With a widely used two-tower structure in recommender model [8, 21, 47], AKGAN consists of two main modules: (1) knowledge graph attribute network, which is illustrated in Figure 2, and (2) user interest-aware attention module, which is illustrated in Figure 3.

The core task of KGAN is to learn KG enhanced item representations that contain multiple attribute information. KGAN adopts GNN framework and has a heterogeneous design in the first layer and the latter layer, which are named attribute modeling layer and attribute propagation layer respectively.

Attribute modeling layer (AML) is to construct initial entity representations using one-hop neighbors' embeddings $\mathbf{e}_{j\in\mathcal{N}_i}^{atr}$, as follows:

$$\mathbf{e}_i^{(0)} = f_{\text{AML}}(\mathbf{e}_{j\in\mathcal{N}_i}^{atr}, \mathcal{G}) \tag{1}$$

where $\mathcal{N}_i$ denotes all neighbors of node i.

Attribute propagation layer (APL) recursively propagates attribute information to acquire more informative item representations as

$$\mathbf{e}_i^{(l)} = f_{\text{APL}}(\mathbf{e}_{j\in\mathcal{N}_i}^{(l-1)}, \mathcal{G}) \tag{2}$$

After performing L layers, we obtain multiple item representations, namely $\{\mathbf{e}_i^{(0)}, ..., \mathbf{e}_i^{(L)}\}$. As the output of l_{th} layer represents l−hop attribute information, we conduct sum operation to pool them and infer final item representations as

$$\mathbf{e}_i^* = \sum_{l=0}^{L} \mathbf{e}_i^{(l)} \tag{3}$$

When item representations have been learned, the **interest-aware attention** (IAA) module is to learn user representations via interaction data with a novel relation-aware attention mechanism which represents user interests towards relations, a.k.a. attributes. Formally, user representations are obtained by IAA as

$$\mathbf{e}_u^* = f_{\text{IAA}}(\mathbf{e}_i^*, O^+) \tag{4}$$

When user representations and item representations are learned, a scoring function is used to predict their matching score and here we adopt inner product operation as

$$\hat{y}(u,i) = \mathbf{e}_u^{*\top} \mathbf{e}_i^* \tag{5}$$

We illustrate the architecture of AML (cf. Section 3.2), APL (cf. Section 3.3) and IAA (cf. Section 3.4) in detail in the following sections.

3.2 Attribute Modeling Layer

In KGs, one entity has different types of relations with neighbors. For example, in figure 2, e_8 has different relations with e_4 and e_5. To model a pure initial representation without semantic pollution, we regard each relation as an attribute and embed each entity in all attribute embedding spaces, which is similar to the design of FFM [14] that embeds each feature in different fields. Therefore each entity has several embeddings and we denote the embedding set in different attribute spaces as

$$\mathcal{E}^{atr} = \{(\mathbf{e}_i^{r_1}, \mathbf{e}_i^{r_2}, ..., \mathbf{e}_i^{r_M})|i \in \mathcal{E}, M = |\mathcal{R}|\} \tag{6}$$

where $\mathbf{e}_i^{r_m} \in \mathbb{R}^{d^m}$ denotes the embedding of entity i in attribute r_m space and d^m is the embedding dimension of attribute r_m space. We detailly introduce the settings about d^m in the Section 3.5. Here we can set d^m as a fixed constant value which is the same as that all latent vectors of different fields share the same dimension in FFM [14]. However, KGs contain hundreds of relation types, which requires huge computational resources both in memory space and time. Therefore, we design variable dimensions of different attribute spaces. The principle is that more edges belonging to one specific relation type bring in a larger dimension of this attribute space. The detailed settings of variable dimensions are provided in Section 3.5.

Then we construct initial entity representations by aggregating attribute embeddings of one-hop neighbors. We can see that each neighbor has several embeddings and just one relation-aware embedding will be used. Taking figure 2 as an example, there are two links (e_8, r_4, e_5) and (e_8, r_6, e_4), we use embeddings $\mathbf{e}_8^{r_4}$ to represent the singer attribute of song e_5 and embedding $\mathbf{e}_8^{r_6}$ to represent the friend attribute of actor e_4, respectively. After we prepared the relation-aware embeddings, the average operation is adopted to pool the same relation-aware neighbors to acquire the main semantics of this attribute as

$$\mathbf{e}_{i'}^{r_j} = \frac{1}{|\mathcal{N}_i^{r_j}|} \sum_{j\in\mathcal{N}_i^{r_j}} \mathbf{e}_j^{r_j} \tag{7}$$

where $j \in \mathcal{N}_i^{r_j}$ and $\mathcal{N}_i^{r_j} = \{j|j \in (j, r_j, i), (j, r_j, i) \in \mathcal{G}\}$ denotes the set of head entities that belong to the triplet where i is tail entity and r_j is relation. For example, if $e_{2,3}$ are both comedy actors and e_4 is an action actor, such that $mean(\mathbf{e}_{2-4}^{r_1})$ represents movie e_1 is likely to involve many comic scenes. Note that not all attributes occur in one-hop range, like movie e_1 doesn't has friend attribute r_6. To obtain a fixed-length representation, we provide zero vector to the absent attribute. Finally, all attribute embeddings will be integrated into one representation by concatenation operation as

$$\mathbf{e}_i^{(0)} = \mathbf{e}_{i'}^{r_1} \| \mathbf{e}_{i'}^{r_2} \| \cdots \| \mathbf{e}_{i'}^{r_M} \tag{8}$$

Note that we adopt concatenation rather than weighted sum operation (i.e., sum, mean, attention), which aims to solve the problem of semantic pollution. More specifically, equation 8 shows that

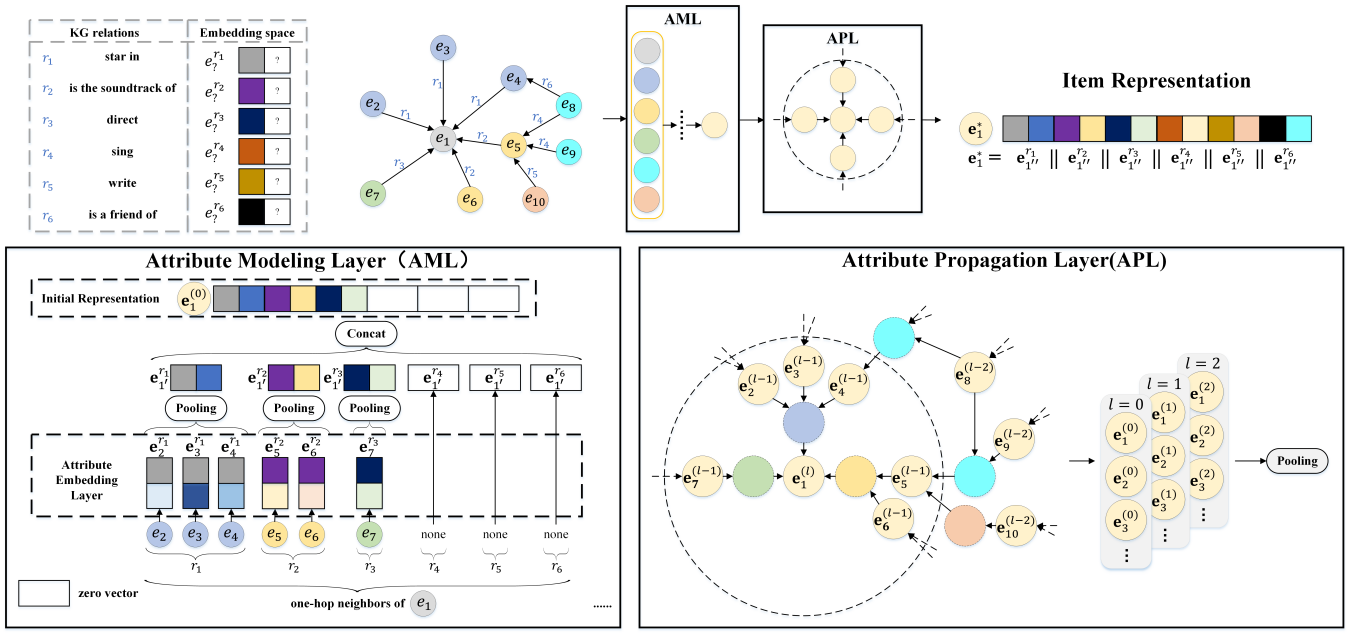

Figure 2: The overall architecture of knowledge graph attribute network. Best viewed in color.

one attribute is represented within a specific range of element-wise positions. In addition, one type of attribute is placed in the same element-wise positions for all entities. For example, two representations $\mathbf{e}_1^{(0)} = \mathbf{e}_{1'}^{r_1} \| \cdots \| \mathbf{e}_7^{r_3} \| \cdots$ and $\mathbf{e}_{11}^{(0)} = \mathbf{e}_{11'}^{r_1} \| \cdots \| \mathbf{e}_7^{r_3} \| \cdots$ mean movie e_1 and movie e_{11} have the same director and different actors. In a word, a concatenated representation avoids the interaction and preserves the semantic independence of different attributes, which will be further advantageous to maintain the weight of significant high-order neighbors in Section 3.3 and distill user-interested attributes in Section 3.4.

3.3 Attribute Propagation Layer

When we obtain initial entity representations via one-hop neighbors, an intuitive idea is to propagate them via GNN framework, so as to aggregate high-order attributes into the center node and acquire more informative item representations.

Here we regard a KG as a heterogeneous graph and adopt a widely used two-step scheme [5, 36, 44] to aggregate the representations of neighbors: (1) same relation-aware neighbors aggregation; (2) relations combination. The pooling methods in the above two steps are average and sum operation, respectively. And we leave the further exploration of other pooling methods like attention as the future work. More formally, in the l−th layer, we recursively formulate the representation of an entity as:

$$\mathbf{e}_i^{(l)} = \sum_{r_j \in \mathcal{R}} \frac{1}{|\mathcal{N}_i^{r_j}|} \sum_{j \in \mathcal{N}_i^{r_j}} \mathbf{e}_j^{(l-1)} \tag{9}$$

Now the final item representation $\mathbf{e}_i^*$ has been learned by equation 3, let's check how KGAN avoids pollution and weight decrease of significant high-order neighbors.

We re-examine $\mathbf{e}_i^*$ from the perspective of element-wise position. Without loss of generality, $\mathbf{e}_i$ is contructed as

$$\mathbf{e}_i^* = \mathbf{e}_{i''}^{r_1} \| \mathbf{e}_{i''}^{r_2} \| \cdots \| \mathbf{e}_{i''}^{r_M} \tag{10}$$

where $\mathbf{e}_{i''}^{r_m} \in \mathbb{R}^{d^m}$ is a truncated vector in $\mathbf{e}_i^* \in \mathbb{R}^D$ and $D = \sum_{r_m \in \mathcal{R}} d^m$. The start and ending index of $\mathbf{e}_{i''}^{r_m}$ in $\mathbf{e}_i^*$ are $\sum_{p=1}^{m-1} d^p$ and $\sum_{p=1}^{m} d^p$, respectively. Reviewing the generation process of $\mathbf{e}_{i''}^{r_m}$, we can see that $\mathbf{e}_{i''}^{r_m}$ is learned by neighbors' embeddings in attribute r_m space and doesn't include any embeddings from other attribute spaces. For example, in figure 2, $\mathbf{e}_{1''}^{r_1}$ is learned by $\mathbf{e}_2^{r_1}$, $\mathbf{e}_3^{r_1}$, and $\mathbf{e}_4^{r_1}$. This means $\mathbf{e}_{i''}^{r_m}$ maintains the independence of attribute r_m and KGAN remains semantics unpolluted after multi-layer propagations. Furthermore, we check specific neighbors which are aggregated into $\mathbf{e}_{i''}^{r_m}$. Take $\mathbf{e}_{1''}^{r_5}$ as an example, after 1-order propagation (one AML), $\mathbf{e}_{1''}^{r_5}$ is a zero vector since attribute r_5 doesn't occur in one-hop neighbors of movie e_1. Then after 2-order propagation (one AML and one APL), $\mathbf{e}_{1''}^{r_5} = \mathbf{e}_{10}^{r_5}$, which seems as if there were a link (e_{10}, r_5, e_1) and e_{10} connected to e_1 directly. In general, when one relation, a.k.a attribute (i.e., r_5), firstly appears in the receptive field of center node (i.e., e_1) at l−hop (i.e., 2−hop) position, the weight of its corresponding node (i.e., e_{10}) will not decrease, which proves that KGAN maintains the weight of significant high-order neighbors.

3.4 Interest-aware Attention Layer

After item representations have been obtained by KGAN, a typical idea in recommender system is to enhance user representations by clicked items, like [16, 30, 39]. We take avearage pooling as an

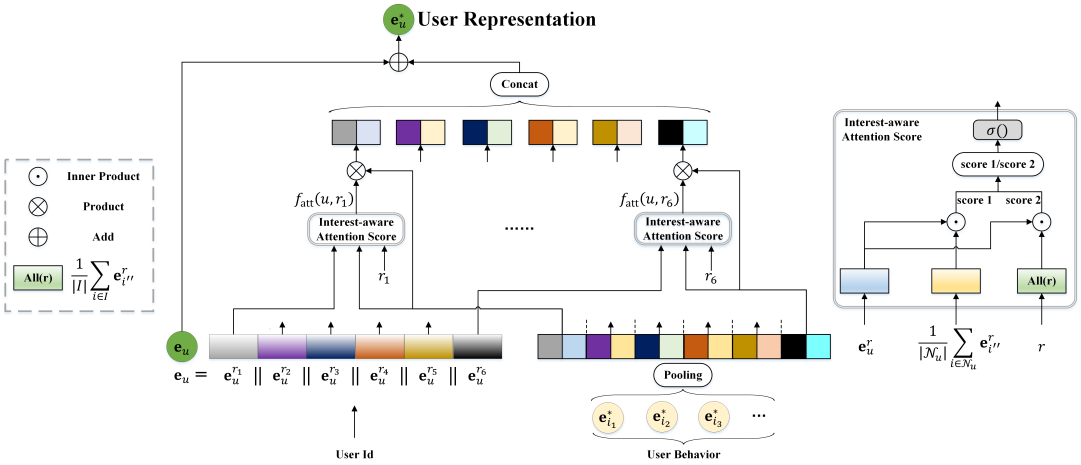

Figure 3: The structure of interest-aware attention module. Best viewed in color.

example, as follows:

$$\mathbf{e}_u^* = \mathbf{e}_u + \frac{1}{\mid \mathcal{N}_u \mid} \sum_{i \in \mathcal{N}_u} \mathbf{e}_i^* \tag{11}$$

where $\mathcal{N}_u = \{i | (u,i) \in O^+\}$ and $\mathbf{e}_u$ represents user embedding for collaborative filtering. However, average pooling doesn't consider user preferences personally, thereby recent works aims to model user interests via attention mechanism, such as [17, 22, 33, 35, 49].

Here we propose a novel attention module to model user interests towards different attributes. We assign an interest score $f_{\text{att}}(u, r_m)$ to each pair of attribute r_m and user u, and personally generate user representation by combining interest scores and interacted items. We firstly introduce how to calculate $f_{\text{att}}(u, r_m)$ and then illustrate how to combine $f_{\text{att}}(u, r_m)$ with interacted items.

With different attributes placed in corresponding element-wise positions as shown in equation 10, we truncate user vector $\mathbf{e}_u$ with the same strategy as

$$\mathbf{e}_u = \mathbf{e}_u^{r_1} \parallel \mathbf{e}_u^{r_2} \parallel \cdots \parallel \mathbf{e}_u^{r_M} \tag{12}$$

where $\mathbf{e}_u^{r_m} \in \mathbb{R}^{d^m}$ is a truncated vector in $\mathbf{e}_u \in \mathbb{R}^D$. The start and ending index of $\mathbf{e}_u^{r_m}$ in $\mathbf{e}_u$ are $\sum_{p=1}^{m-1} d^p$ and $\sum_{p=1}^{m} d^p$, respectively. Then the interest score of user u towards attribute r_m is calculated by

$$f_{\text{att}}(u, r_m) = \sigma\left(\tau \frac{\frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} \mathbf{e}_u^{r_m \top} \mathbf{e}_{i''}^{r_m}}{\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mathbf{e}_u^{r_m \top} \mathbf{e}_{i''}^{r_m}}\right) \tag{13}$$

where τ is the temperature coefficient as a hyperparameter. In equation 13, $\frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} \mathbf{e}_u^{r_m \top} \mathbf{e}_{i''}^{r_m}$ denotes the interest-degree of user u for his interacted items in attribute r_m, while $\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mathbf{e}_u^{r_m \top} \mathbf{e}_{i''}^{r_m}$ denotes the interest-degree of user u for all items in attribute r_m. If user u attaches great importance to attribute r_m when choosing items, the corresponding truncated representation $\mathbf{e}_{i''}^{r_m}$ of his interacted items will be radically different from that of other uninterested items, such that $\frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} \mathbf{e}_u^{r_m \top} \mathbf{e}_{i''}^{r_m}$ will be much greater than $\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mathbf{e}_u^{r_m \top} \mathbf{e}_{i''}^{r_m}$, and vice versa. Therefore, the ratio between the above two expressions denotes the interest-degree of user u for attribute r_m. Since a negative value of ratio is meaningless, we first feed this ratio to Relu and then select tanh as the nonlinear activation function, therefore $\sigma(x) = \tanh(\text{Relu}(x))$.

After $f_{\text{att}}(u, r_m)$ has been prepared, we need to combine it with interacted items to learn user representations. We firstly re-express equation 11 as

$$\mathbf{e}_u^* = \mathbf{e}_u + \mathop{\Vert}_{r_m \in \mathcal{R}} \left(\frac{1}{\mid \mathcal{N}_u \mid} \sum_{i \in \mathcal{N}_u} \mathbf{e}_{i''}^{r_m} \right) \tag{14}$$

where $\Vert$ is the concatenation operation. Therefore, an intuitive idea is to use $f_{\text{att}}(u, r_m)$ to control how much attribute information, a.k.a. $\frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} \mathbf{e}_{i''}^{r_m}$, will be passed to user. Consider the limit case, when $f_{\text{att}}(u, r_m) = 0$ which means user u doesn't pay attention to attribute r_m at all, any information of attribute r_m should not be contained in user representation. Formally, user representation is obtained by

$$\mathbf{e}_u^* = \mathbf{e}_u + \mathop{\Vert}_{r_m \in \mathcal{R}} \left(f_{\text{att}}(u, r_m) \sum_{i \in \mathcal{N}_u} \mathbf{e}_{i''}^{r_m} \right) \tag{15}$$

Note that here we relax the constraint that the sum of attention weights towards all attributes is 1, a.k.a. $\sum_{r_m \in \mathcal{R}} f_{\text{att}}(u, r_m) \neq 1$. The reason is as follows: when the number of members participating in the attention calculation is large and the limitation that the sum of attention weights is 1 is still reserved at this time, it will cause the attention weight of each member to be dispersed, making it difficult to learn the coefficients of important nodes. This problem has also appeared in Graphair [11], which shows estimating $O(\mid \mathcal{N} \mid^2)$ coefficients exposes the risk of overfitting. Therefore, we introduce the above novel activation unit to release the limitation that the sum of attention weights is 1, and the same idea is adopted by DIN [50].

3.5 Embedding Dimension

In KG, the number of edges varies according to relation type. We can set d^m as a fixed constant value which is the same as that all latent vectors of different fields share the same dimension in

Table 2: Edge number of Alibaba-iFashion dataset.

# r_0 - r_9	260,477 469 2,518 482 1177 343 429 618 143 865
# r_{10} - r_{19}	2,443 1,572 155 939 221 934 244 187 162 93
# r_{20} - r_{29}	833 161 129 736 22 61 318 142 435 36
# r_{30} - r_{39}	296 459 19 102 73 23 53 321 63 45
# r_{40} - r_{50}	132 7 10 40 42 15 78 11 11 5 6

FFM [14]. However, KGs contain hundreds of relation types, which requires huge computational resources both in memory space and time. Table 2 shows an example on the Alibaba-iFashion dataset. Therefore, we design variable dimensions of different attribute spaces. The principle is that more edges belonging to one specific relation type bring in a larger dimension of this attribute space. Formally, we make the dimension increase linearly with the edge number as below:

$$d^m = \begin{cases} d_{min} + \dfrac{d_{max} - d_{min}}{c}|r_m|, & |r_m| \leq \dfrac{c}{d_{max} - d_{min}} \\ d_{max}, & |r_m| > \dfrac{c}{d_{max} - d_{min}} \end{cases} \tag{16}$$

where d_{max}, d_{min}, c, c are hyperparameters and $|r_m|$ is the number of edges belonging to relation r_m, like $|r_0| = 260477$ in Alibaba-iFashion.

3.6 Model Optimization

With the user representation $\mathbf{e}_u^*$ and the item representation $\mathbf{e}_i^*$ ready, equation 5 is adopted to calculate the prediction score of each pair of user and item. Then we employ the BPR loss [24] to encourage that the observed interactions should be assigned higher prediction values than unobserved ones. The objective function is formulated as

$$\mathcal{L} = \sum_{(u,i^+,i^-)\in O} -\ln \sigma\Big(\hat{y}(u,i^+) - \hat{y}(u,i^-)\Big) + \lambda\|\Theta\|_2^2 \tag{17}$$

where $O = \{(u,i^+,i^-)|(u,i^+) \in O^+, (u,i^-) \in O^-\}$ denotes the training set, which contains the observed interactions O^+ and the unobserved interactions O^-; $\sigma(\cdot)$ is the sigmoid function. $\Theta = \{\mathbf{e}_i^{r_m}, \mathbf{e}_u | i \in \mathcal{I}, u \in \mathcal{U}, r_m \in \mathcal{R}\}$ is the model parameter set. L_2 regularization parameterized by λ on Θ is conducted to prevent overfitting. We employ the Adam [15] optimizer and use it in a mini-batch manner.

4 EXPERIMENTS

We evaluate our proposed AKGAN method on three benchmark datasets to answer the following research questions:

- **RQ1**: Does our proposed AKGAN outperform the state-of-the-art recommendation methods?
- **RQ2**: How do different components (i.e., attribute modeling layer, attribute propagation layer, and interest-aware attention layer) affect AKGAN?
- **RQ3**: Can AKGAN provide potential explanations about user preferences towards attributes?

4.1 Experimental Settings

Table 3: Statistics of the datasets. 'ave./int.' indicates the average number of interactions per user.

Datasets	Amazon-book	Last-FM	Alibaba-iFashion
# users	70,679	23,566	114,737
# items	24,915	48,123	30,040
# interactions	847,733	3,034,796	1,781,093
# ave./int.	11.99	128.78	15.52
# entities	88,572	58,266	59,156
# relations	39	9	51
# triples	2,557,746	464,567	279,155

Dataset Description. We choose three benchmark datasets to evaluate our method: Amazon-Book[1], Last-FM[2], and Alibaba-iFashion[3]. The former two datasets are released in [33] and the last one is released in [35]. All of them are publicly available. Each dataset consists of two parts: user-item interactions and a corresponding knowledge graph. The basic statistics of the three datasets are presented in Table 3. We follow the same data partition used in [33, 37] to split the datasets into training and testing sets. For each observed user-item interaction, we randomly sample one negative item that the user has not interacted with before, and pair it with the user as a negative instance.

Evaluation Metrics. We evaluate our method in the task of top-K recommendation. For each user, we treat all the items that the user has not interacted with as negative and the observed items in the testing set as positive. Then we rank all these items and adopt two widely-used evaluation protocols: Recall@K and NDCG@K, where K is set as 20 by default. We report the average metrics for all users in the testing set.

Baselines. To demonstrate the effectiveness, we compare AKGAN with KG-free (MF [24]), attribute-based (FM [25] and FFM [14]), knowledge-aware embedding-based (CKE [45]), knowledge-aware path-based (RippleNet [29]), and knowledge-aware GNN-based (KGAT [33], KGNN-LS [31], KGIN [35]) methods.

- **MF** [24]: This is matrix factorization optimized by the Bayesian personalized ranking (BPR) loss, which only considers the user-item interactions and leaves KG untouched.
- **FM** [25]: FM uses factorization machines to explore high-order feature interactions.
- **FFM** [14]: FFM is the variant of FM and embeds each feature in different fields.
- **CKE** [45]: This method uses TransR [18], a typical knowledge graph embedding algorithm, to regularize the representations of items, which are fed into MF framework for recommendation.
- **RippleNet** [29]: RippleNet first assigns entities in the KG with initial embeddings using TransE [2] and propagates users' preferences on the KG via entities related to his historically clicking items.

[1] http://jmcauley.ucsd.edu/data/amazon
[2] https://grouplens.org/datasets/hetrec-2011/
[3] https://drive.google.com/drive/folders/1xFdx5xuNXHGsUVG2VIohFTXf9S7G5veq

- **KGAT** [33]: This method encodes user behaviors and item knowledge as an unified knowledge graph to exploit high-order connectivity using an attentive neighborhood aggregation mechanism.
- **KGNN-LS** [31]: It transforms a heterogeneous KG into a user-personalized weighted graph and employs label smoothness regularization to avoid overfitting of edge weights.
- **KGIN** [35]: KGIN is the state-of-the-art GNN-based recommender. It models each intent as an attentive combination of KG relations to explore intents behind user-item interactions and adopts a novel relational path-aware aggregation scheme.

Parameter Settings. We employ Pytorch to implement all baselines and the proposed AGKAN. See Appendix **??** for more details about parameter settings. We implement our AGKAN model in Pytorch and Deep Graph Library (DGL)[4], which is a Python package for deep learning on graphs. We released all implementations (code, datasets, parameter settings, and training logs) to facilitate reproducibility[5]. The embedding size of one attribute space in AGKAN is designed by equation 16. For a fair comparison, we fix the size of ID embeddings of all baselines as the sum of all attribute dimensions in AKGAN. We adopt Adam [15] as the optimizer and the batch size is fixed at 1024 for all methods. We use the Xavier initializer [6] to initialize model parameters and apply a grid search for hyper-parameters: the learning rate is searched in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, the coefficient of L2 normalization is tuned in $\{10^{-5}, 10^{-4}, \cdots, 10^{-1}\}$, the number of GNN layers L is searched in $\{1, 2, 3\}$ for GNN-based methods, and the dropout ratio is tuned in $\{0.0, 0.1, \cdots, 0.9\}$. Besides, we use the node dropout technique for KGAT, KGIN, and AKGAN, where the ratio is searched in $\{0.0, 0.1, \cdots, 0.9\}$. For FM and FFM, we take attribute nodes within 3-hop (2-hop in Alibaba-iFashion dataset for best performance) range as the features. For RippleNet, we set the number of hops as 2, and the memory size as 5, 15, 8 for Alibaba-iFashion, Last-FM, and Amazon-book respectively to obtain the best performance. Since FM, FFM and RippleNet is a model for CTR prediction, we generate top-K items with top-K scores in all items, which are compared with the test set to compute Recall@K and NDCG@K. For KGAT, we use the pre-trained ID embeddings of MF as the initialization, which is also adopted by KGIN. Moreover, early stopping strategy is performed, i.e., premature stopping if Recall@20 on the test set does not increase for 10 successive epochs.

We list the parameter settings of AKGAN on three datasets in Table 4 for better reproducibility, where the hyperparameters include the learning rate l_r, the coefficient λ of L_2 regularization, the temperature coefficient τ, and three coefficients about dimension d_{max}, d_{min}, c.

Table 4: Hyperparameter settings of AKGAN.

	l_r	λ	τ	d_{max}	d_{min}	c
Amazon-Book	10^{-4}	10^{-5}	0.25	32	4	5000
Last-FM	10^{-4}	10^{-5}	0.5	64	16	5000
Alibaba-iFashion	10^{-4}	10^{-5}	0.1	64	4	5000

[4]https://github.com/dmlc/dgl
[5]https://github.com/huaizepeng2020/AKGAN

4.2 Performance Comparison(RQ1)

We report the empirical results in Table 5 and use %Imp. to denote the percentage of relative improvement on each metric. The observations are as follows:

- AKGAN consistently achieves the best performance on three datasets in terms of all measures. Specifically, it achieves significant improvements over the strongest baselines w.r.t. NDCG@20 by 8.63%, 26.42%, and 11.87% in Amazon-Book, Last-FM, and Alibaba-iFashion, respectively. These improvements are attributed to the following reasons: (1) By combining all attributes using concatenation operation, AKGAN avoids semantic pollution caused by weighted sum operation and learns more high-quality item representations for recommendation. (2) Compared to GNN-based baselines (i.e., KGAT, KGNN-LS, KGIN), AKGAN maintains the weight of significant high-order neighbors by placing different attributes in corresponding element-wise positions. (3) Benefiting from our novel interest-aware attention module that assigns an interest score to each pair of user and attribute, AKGAN can recognize the pattern of user interest at a fine-grained level to conduct a better personal recommendation.
- Jointly analyzing AKGAN across the three datasets, we find that the improvement on Last-FM is more significant than that on Alibaba-iFashion and Amazon-Book. The main reason is that the average number of interactions per user of Last-FM (128.78) is much larger than that of the other two datasets (11.99, 15.52). Therefore, there exists richer interaction information on the Last-FM dataset for AKGAN to refine user and item representations by collaborative signals. This indicates that AKGAN will fully realize its potential in recommendation scenarios with dense interaction data.
- KG-free method (MF) underperforms knowledge-aware methods (i.e., CKE, RippleNet, AKGAN). A clear reason is MF doesn't leverage the rich attribute information in KG.
- GNN-based methods (i.e., KGAT, KGIN, AKGAN) achieve better performances than path-based (RippleNet) and embedding-based (CKE) methods in Amazon-Book and Last-FM, suggesting the superiority of GNN in leveraging high-order attribute information. However, in Alibaba-iFashion, RippleNet and CKE outperform part GNN-based methods (i.e., KGAT, KGCN-LS). A possible reason is that KG embedding algorithm is more suitable for modeling the major first-order connectivity in Alibaba-iFashion, which is similar with the results in [35].
- In four GNN-based methods, AKGAN performs best, KGIN is the second-best, while KGAT and KGNN-LS are at the same level and achieve the worst results. The decreasing performance is because that the level of how a recommender learns item attributes and user interests is in descending order, from fine-grained to coarse-grained. AKGAN uses concatenation operation to avoid attribute interaction while other models adopt weighted sum (attentive combination) operation. AKGAN maintains the weight of significant high-order neighbors while other models neglect this issue. AKGAN, KGIN, and KGNN-LS all learn user interests towards attributes explicitly while KGAT doesn't.
- FFM slightly underperforms KGIN but outperforms other baselines, which proves the necessity of embedding attributes in different latent spaces.

Table 5: Overall Performance Comparison. The best performance is boldfaced; the runner up is labeled with '*'. '%Imp.' indicates the relative improvements.

Dataset	Metrics	KG-free	Attribute-based		Embedding-based	Path-based	GNN-based				Imp.
		MF	FM	FFM	CKE	RippleNet	KGAT	KGNN-LS	KGIN	AKGAN	
Amazon-Book	Recall	0.1241	0.1366	0.1601	0.1287	0.1355	0.1473	0.1389	0.1687*	**0.1783**	5.69%
	NDCG	0.0650	0.0755	0.0912	0.0674	0.0763	0.0782	0.0614	0.0915*	**0.0994**	8.63%
Last-FM	Recall	0.0774	0.0771	0.0935	0.0780	0.0842	0.0876	0.0877	0.0978*	**0.1209**	23.62%
	NDCG	0.0669	0.0648	0.0810	0.0659	0.0766	0.0745	0.0653	0.0848*	**0.1072**	26.42%
Alibaba-iFashion	Recall	0.0921	0.1003	0.1089	0.1068	0.1121	0.1015	0.1046	0.1147*	**0.1253**	9.24%
	NDCG	0.0562	0.0577	0.0681	0.0633	0.0695	0.0616	0.0582	0.0716*	**0.0801**	11.87%

Table 6: Impact of concatenation operation and interest score.

	Amazon-Book		Last-FM		Alibaba-iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
AKGAN-mean	0.1504	0.0799	0.0849	0.0713	0.1064	0.0660
AKGAN-sum	0.1489	0.0784	0.0886	0.0736	0.1045	0.0643
AKGAN-w/o att	0.1758	0.0969	0.1141	0.1012	0.1213	0.0774

Table 7: Impact of the number of layers L.

	Amazon-Book		Last-FM		Alibaba-iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
AKGAN-1	0.1737	0.0974	0.1192	0.1060	0.1245	0.0791
AKGAN-2	0.1768	0.0987	0.1205	0.1069	0.1253	0.0801
AKGAN-3	0.1783	0.0994	0.1209	0.1066	0.1221	0.0776

4.3 Study of AKGAN(RQ2)

In this section, we first conduct an ablation study to investigate the effect of concatenation operation and interest-aware attention layer. Towards further analysis, we study the influence of layer numbers. In the following, we explore how the temperature coefficient affects the performance.

Impact of concatenation operation & interest score. To demonstrate the necessity of concatenation operation and interest score, we compare the performance of AKGAN with the following three variants: (1) combing different attributes with the average operation and discarding interest score, termed AKGAN-mean, (2) combing different attributes with the sum operation and discarding interest score, termed AKGAN-sum, (3) only discarding interest score, termed AKGAN-w/o att. Discarding interest score means we use equation 11 to learn user representations. Note that weighted sum operation and interest-aware attention layer are not retained simultaneously. The reason is that concatenation operation is the prerequisite of interest-aware attention layer, therefore we have to discard the interest score when testing the weighted sum operation. The results are shown in Table 6 and the major findings are as below:

- Comparing AKGAN-mean/AKGAN-sum with AKGAN-w/o att, we can clearly see that replacing concatenation operation with weighted sum operation dramatically degrades performance of recommendation, which indicates the superiority of concatenation operation.
- The improvement from AKGAN-w/o att to AKGAN verifies the necessity of interest score.
- The performance decrease caused by removing concatenation operation is larger than that caused by discarding interest score in three datasets. This helps validate the great benefits brought by embedding entity in different attribute spaces and using cancatenation operation to combine them.
- Jointly comparing AKGAN-w/o att and AKGAN across the three datasets, we find that the improvement on Last-FM is more significant than that on Alibaba-iFashion and Amazon-Book, which is consistent with the aforesaid conclusion in Section 4.2. The main reason is that Last-FM has more dense interaction data than the other two datasets, such that AKGAN can better learn user interests in Last-FM.

Impact of model depth. We investigate the influence of depth of receptive field in AKGAN by searching L in the range of $\{1, 2, 3\}$. Particularly, $L = 1$ means AKGAN has only one attribute modeling layer. The results are reported in Table 7. In Amazon-Book and Last-FM, increasing propagation times of attributes can boost the performance, because more attribute information is aggregated into the center node to learn more informative user and item representations. While in Alibaba-iFashion, AKGAN-2 is the best and AKGAN-3 is the worst. The reason is that all attribute information of an item node is stored within its 2-hop range, leading to the best performance of AKGAN-2. See the detailed topology of Alibaba-iFashion KG in Section 4.4 for a better understanding.

Impact of temperature coefficient. We vary the temperature coefficient τ in the range of $\{0.01, 0.1, 0.2, ..., 0.9, 1\}$ to study its influence on the performance of AKGAN and summarize the results on partial datasets in Figure 4. We can see that the best performance occurs when τ is set to the optimal value. The reason is that τ describes the sensitivity of the interest score relative to a given divisor in equation 13. A large τ will make the output of equation 13 saturate too early, while a small τ will cause underfitting. The result on Last-FM dataset is omitted since it has a similar conclusion.

4.4 Case Study(RQ3)

In this section, we visualize the interest score to show how AKGAN learns user preferences towards attributes and use Alibaba-iFashion as the example dataset since it helps to provide an intuitive

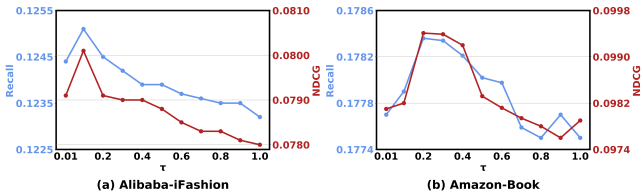

Figure 4: Impact of temperature coefficient τ. Best viewed in color.

Table 8: Part numbers of relations appeared in user0's history behaviors.

Staff	Category(number)	Staff	Category(number)
Tops	T-shirt(6)	Shoes	Women's shoes(10)
	Trench coat(40)		Boots(15)
	Women's Shirt(7)	Handbag	Women's Handbag(2)
	Sweater(1)	Accessories	Ear studs(16)
	Wool coat(28)		Earrings(23)
	Overcoat(3)		Hat(13)
Bottoms	Women's pants(9)		Necklace(17)
	Men's pants(38)		
	Jeans(4)		
	Skirt(20)		

explanation. For better comprehension, we firstly give a detailed introduction about this dataset.

Alibaba-iFashion is an E-commerce dataset, which contains user-outfit click history for recommendation. Each outfit consists of several fashion staffs (e.g., tops, bottoms, shoes) and each staff is assigned with different categories. For example, trench coat, T-shirt and sweater are three kinds of tops, pants and long skirt are two kinds of bottoms. Alibaba-iFashion regards these categories as relations in KG and there is one more relation between outfit and staff, called *including*. Therefore, Alibaba-iFashion KG has two kinds of triples: *(outfit, including, staff)* and *(staff, having-category-?, ?)*, where ? is the specific category such as *(Air Jordan, having-category-shoes, shoes)*. Therefore, all attribute information has been captured in the 2-hop range. According to Table 3, Alibaba-iFashion KG has 51 relations, where we number relation *including* as 0 and number 50 staff categories from 1 to 50.

We randomly select a user called *User0* as the example user. Figure 5 introduces his historical behaviors, including both training set and testing set, and Table 8 lists part numbers of relations appeared in user0's history behaviors. Figure 6 displays the interest score learned from the training set and we use the testing set to validate the effectiveness of these scores. We have the following observations:

- The interest score accurately models user0's preferences towards attributes. Among 11 outfits clicked by user0 (in Figure 5), the tops contain four main kinds: T-shirt, shirt, trench coat, and sweater, which means user0 prefers the above four types of tops. The corresponding interest score are 0.3254, 0.4379, 0.6793, 0.7256, respectively. The most clicked bottoms are women's pants which are assigned with a high score 0.8223, while the jeans and skirt occur less frequently and their scores, 0.0601 and 0.0671, are particularly small. As for shoes, user0 prefers women's shoes and boots only appear once, which is reflected in their corresponding scores, 0.7623 and 0.0431. The earrings and hat are two main accessories and both of them are assigned with high scores, 0.5506 and 0.3506.
- All outfits clicked by user0 have only one kind of handbag and the interest score $f_{\text{att}}(u_0, r_2) = 0.1506$, which is a small value. A possible reason is that outfits are manually created by Taobao's fashion experts who prefer to include handbags for the completion of the outfit [4]. Therefore user0 judges whether to click an outfit based on other staffs rather than the handbag. Another explanation is that handbag is not as frequently changed as other staffs like tops in daily dressing, such that user0 pays less attention to the handbag when browsing through outfits.
- Testing set (in Figure 5) proves the effectiveness of interest score. For example, $f_{\text{att}}(u_0, r_{40}) = 0.6793$ is proved to be reansonable since outfit12 which contains a trench coat appears in the testing

Figure 5: The training set and testing set of user0. Best viewed in color.

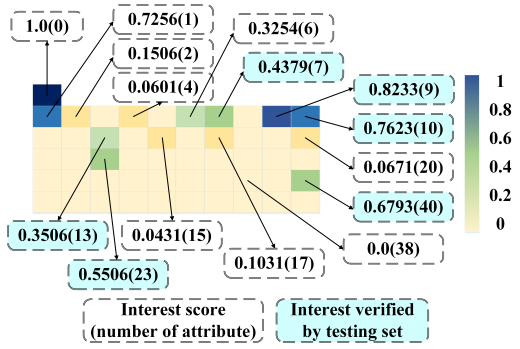

Figure 6: An example of the interest score of user0 to verify the interpretability of AKGAN. Each dotted box includes an interest score and corresponding number of attributes within parentheses. The dotted box with a background color means the effectiveness of this interest score has been validated by testing set in Figure 5. Best viewed in color.

set. And $f_{\text{att}}(u_0, r_9) = 0.8233$ is validated by outfit13 and outfit14 with same reason.

- In one staff, the interest degree varies according to category. Take accessories as an example, user0 prefers earrings than hat and $f_{\text{att}}(u_0, r_{23}) = 0.5506 > f_{\text{att}}(u_0, r_{13}) = 0.3506$. This result is also proved by the testing set, where two outfits contain earrings while only one outfit has a hat.
- Unconcerned attribute is assigned with a zero score, like $f_{rmatt}(u_0, r_{38}) = 0$. The reason is that user0 perhaps is female since her click histories are all women's clothing and she isn't interested in men's pants.
- Attribute *including* has the highest score 1. The reason is that this relation *(outfit, including, staff)* is the necessary bridge to acquire what staffs an outfit consists of.

5 RELATED WORK

Our work is highly related with the knowledge-aware recommendation, which can be grouped into three categories.

Embedding-based methods [1, 3, 12, 30, 45, 46] hire KG embedding algorithms (i.e., TransE [2] and TransH [38]) to model prior representations of item, which are used to guide the recommender model. For example, DKN [30] learns knowledge-level embedding of entities in news content via TransD [13] for news recommendation. KSR [12] utilizes knowledge base information learned with TransE as attribute-level preference to enhance the sequential recommendation. CFKG [46] first defines a user-item knowledge-graph structure specialized for recommender systems, then conducts collaborative filtering on this graph to provide personalized recommendations. KTUP [3] reveals the preferences of users on consuming items via the KG embedding algorithm and improves the preformances of both tasks via joint learning. However, they ignore higher-order connectivity in a knowledge graph and fail to explore multi-hop information, resulting in missing the reasons why users interact with items. Although these methods prove the effectiveness of knowledge-aware embeddings, they ignore the higherorder connectivity in a knowledge graph and fail to explore multi-hop information, resulting in missing the reasons why users interact with items.

Path-based methods [10, 19, 27–29, 41–43] usually predefine a path scheme (i.e., meta-path) and leverages path-level semantic similarities of entities to refine the representations of users and items. HeteRec [42] introduces metapath-based latent features to represent the connectivity between users and items along different types of paths. Hete-CF [19] effectively utilizes multiple types of relations in a heterogeneous social network, which can be extended to arbitrary social-based recommendation scenarios, including event based social networks, location based social networks, and etc.. MCRec [10] learns the explicit representations of meta-paths to depict the interaction context of user-item pairs. RKGE [28] mines the path relation between user and item automatically and encodes the entire path using a recurrent network to predict user preference towards this item. RippleNet [29] memorizes the item embeddings with paths rooted at each user, and adopts them to enhance user representations. However, they rely on expert knowledge and are limited by path quality, resulting in poor universality and robustness across different recommendation scenarios.

GNN-based methods [23, 26, 31, 33, 35, 48] utilize the message-passing mechanism in graph to aggregate high-order attribute information into item representation for enhanced and explainable recommendation. KGAT [33] regards user-item interaction as a new relation added to KG and then employs attentive mechanism to propagate attribute information. IntentGC [48] reconstructs user-to-user relationships and item-to-item relationships based on KG and proposes a novel graph convolutional network to aggregate the attribute information from neighbors. HAKG [26] extracts the expressive subgraphs that link user-item pairs to characterize their connectivities, which are encoded via a hierarchical attentive subgraph encoding to generate effective subgraph embeddings for enhanced user preference prediction. KNI [23] proposes neighborhood interaction (NI) to explore the neighborhood structures of users and items and captures more complicated structural patterns behind user-item interactions. KGIN [35] learn user interest via an attentive combination of attributes and integrates relational information from multi-hop paths to refine the representations. However, they all ignore the semantic pollution problem in KG and the weight decrease of significant high-order nodes in the aggregating process.

6 CONCLUSION

In this paper, we study the attribute information in knowledge graphs to improve recommendation performance. On the item side, the proposed AKGAN can learn more high-quality item representation by maintaining the independency of attributes and maintaining the weight of high-order significant attribute nodes. On the user side, AKGAN mines user interests towards attributes and provides a personal recommendation. Extensive experiments demonstrate the effectiveness and explainability of AKGAN. For future work, we plan to investigate how to model the evolving process of user interests toward attributes based on the long-term behavior sequence. Another direction is to explore whether attribute interaction will benefit recommendation since feature interaction has shown great success in click-through rate prediction.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *NIPS* 26 (2013).

[3] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*. 151–161.

[4] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: personalized outfit generation for fashion recommendation at Alibaba iFashion. In *KDD*. 2662–2670.

[5] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*. 2331–2341.

[6] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *JMLR*. JMLR Workshop and Conference Proceedings, 249–256.

[7] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *TKDE* (2020).

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[10] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *KDD*. 1531–1540.

[11] Fenyu Hu, Yanqiao Zhu, Shu Wu, Weiran Huang, Liang Wang, and Tieniu Tan. 2021. Graphair: Graph representation learning with neighborhood aggregation and interaction. *Pattern Recognition* 112 (2021), 107745.

[12] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*. 505–514.

[13] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*. 687–696.

[14] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *RecSys*. 43–50.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.

[17] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *ICKM*. 2615–2623.

[18] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

[19] Chen Luo, Wei Pang, Zhe Wang, and Chenghua Lin. 2014. Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *ICDM*. IEEE, 917–922.

[20] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*. 931–940.

[21] Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, and Zhenzhong Chen. 2020. A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce. In *KDD*. 3405–3415.

[22] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User behavior retrieval for click-through rate prediction. In *SIGIR*. 2347–2356.

[23] Yanru Qu, Ting Bai, Weinan Zhang, Jianyun Nie, and Jian Tang. 2019. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–9.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[25] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 635–644.

[26] Xiao Sha, Zhu Sun, and Jie Zhang. 2019. Attentive knowledge graph embedding for personalized recommendation. *arXiv preprint arXiv:1910.08288* (2019).

[27] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. 2015. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *CIKM*. 453–462.

[28] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*. 297–305.

[29] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *ICKM*. 417–426.

[30] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *WWW*. 1835–1844.

[31] 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 *KDD*. 968–977.

[32] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*. 839–848.

[33] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *KDD*. 950–958.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[35] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *WWW*. 878–887.

[36] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[37] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *WWW*. 99–109.

[38] Zhigang Wang, Juanzi Li, Zhiyuan Liu, and Jie Tang. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*. 4–17.

[39] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *SIGIR*. 235–244.

[40] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.

[41] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI* 27 (2013).

[42] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*. 283–292.

[43] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *RecSys*. 347–350.

[44] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.

[45] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*. 353–362.

[46] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540* (2018).

[47] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. In *WWW*. 2220–2231.

[48] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In *KDD*. 2347–2357.

[49] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.

[50] 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 *KDD*. 1059–1068.