Pattern Recognition 128 (2022) 108628

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Personalized knowledge-aware recommendation with collaborative and attentive graph convolutional networks

Quanyu Dai [a,1], Xiao-Ming Wu [b,*], Lu Fan [b], Qimai Li [b], Han Liu [c], Xiaotong Zhang [c], Dan Wang [b,*], Guli Lin [d], Keping Yang [d]

[a] *Noah's Ark Lab, Huawei, Shenzhen 518129, China*
[b] *Department of Computing, The Hong Kong Polytechnic University, Hung Hom 999077, Hong Kong*
[c] *School of Software, Dalian University of Technology, Dalian 116024, China*
[d] *Alibaba Group, Hangzhou 311121, China*

ARTICLE INFO

Article history:
Received 18 October 2020
Revised 27 November 2021
Accepted 5 March 2022
Available online 6 March 2022

Keywords:
Recommender system
Graph convolutional network
Attention mechanism
Knowledge graph

ABSTRACT

Knowledge graphs (KGs) are increasingly used to solve the data sparsity and cold start problems of collaborative filtering. Recently, graph neural networks (GNNs) have been applied to build KG-based recommender systems and achieved competitive performance. However, existing GNN-based methods are either limited in their ability to capture fine-grained semantics in a KG, or insufficient in effectively modeling user-item interactions. To address these issues, we propose a novel framework with collaborative and attentive graph convolutional networks for personalized knowledge-aware recommendation. Particularly, we model the user-item graph and the KG separately and simultaneously with an efficient graph convolutional network and a personalized knowledge graph attention network, where the former aims to extract informative collaborative signals, while the latter is designed to capture fine-grained semantics. Collectively, they are able to learn meaningful node representations for predicting user-item interactions. Extensive experiments on benchmark datasets demonstrate the effectiveness of the proposed method compared with state-of-the-arts.

1. Introduction

The information explosion is creating serious challenges in various aspects of modern society, especially in online environments such as search engines, E-commerce, multimedia platforms, and news portals. People are overwhelmed by the huge number of choices and options before they can benefit from the abundance of information. To deal with these challenges, recommender systems have become a necessity for web applications.

Collaborative filtering has been widely adopted in recommender systems to predict user interests [1–3]. However, it suffers from the sparsity issue of user-item interaction data and the cold-start problem, which may greatly affect the quality of recommendation. To address these issues, many studies seek to leverage the rich side information of items such as properties of items.

A variety of existing methods use item attributes independently [5], which may not be sufficient to capture item relations. To improve performance, some recent studies [6–8] propose to model items and their attributes using the readily available knowledge graphs (KGs) such as YAGO, NELL, and DBpedia. KGs explicitly encode relations among entities (e.g., items and their attributes) in the form of triples: (*head entity, relation, tail entity*), and each triple represents a fact or some concept relation, e.g., (*Interstellar, DirectedBy, Nolan*). A large number of triples then form a heterogeneous graph containing rich semantic relations and complex connectivity patterns, which can be used to extract item relations and improve recommendation quality.

For KG-based recommendation, it is important to explicitly model high-order relations in *both* the bipartite graph of user-item interactions and the KG. For example, consider a movie recommendation task illustrated in Fig. 1. User u_1 may want to watch movie i_3 ("Interstellar") for either one of the following reasons: 1) User u_2, who has watched the same movie i_2 ("Contact") as u_1, also watched movie i_3 (path $u_1 - i_2 - u_2 - i_3$); 2) Both movies i_2 and i_3 are science fiction and share a common leading actor Matthew (paths $u_1 - i_2 - e_5/e_7 - i_3$). Modeling high-order relations allows to capture these collaborative signals and informative

* Corresponding authors.
E-mail addresses: csxmwu@comp.polyu.edu.hk (X.-M. Wu), csdwang@comp.polyu.edu.hk (D. Wang).
[1] This work was done while the author was a student in the Department of Computing, The Hong Kong Polytechnic University.

https://doi.org/10.1016/j.patcog.2022.108628
0031-3203/

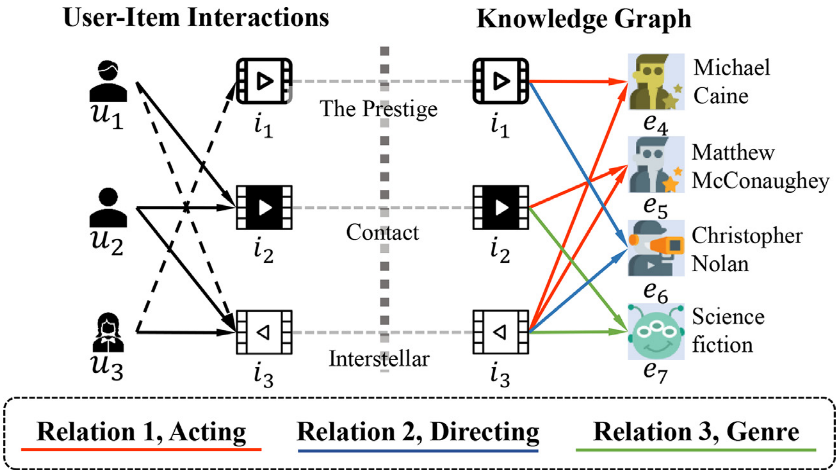

Fig. 1. Illustration of KG-based recommendation with a movie recommendation example.

KG relations for recommendation. On the other hand, it is necessary to design mechanisms to capture user-specific fine-grained semantics of the KG for recommendation. For example, user u_3 might be more interested in movie i_1 (path $u_3 - i_3 - e_6 - i_1$) than movie i_2 (path $u_3 - i_3 - e_5 - i_2$) because both i_1 and i_3 are directed by his/her favorite director Nolan and he/she cares more about the director (Nolan) than the leading actor (Matthew) when selecting a movie. Modeling user-specific relation preference is therefore critical to achieve personalized recommendation. Furthermore, entities might have a different importance weight in representing their common neighbor. For instance, entity e_5 (Matthew) as the leading actor should weigh in more than the supporting actor e_4 (Michael) in characterizing movie i_3 ("Interstellar") in general. However, how to design effective mechanisms to capture these fine-grained semantics remains a challenging problem.

In recent years, substantial efforts have been devoted to designing knowledge-aware recommender systems [6,8,9]. Some studies [10,11] adopt path-based approaches to explicitly model high-order relations among users and items. However, they require manually designed meta-paths for each specific heterogeneous graph and thus are heavily dependent on domain knowledge. Some regularization-based methods [6,12] enhance collaborative filtering models with KG embedding techniques so as to capture the complex relationships among items. For example, CKE [6] uses TransR [13] to encode the translational property of KG embeddings. Albeit being flexible, these methods only implicitly refine item embeddings with a KG embedding loss, which are insufficient to deal with KG-based recommendation tasks.

Most recently, recommendation methods based on graph neural networks have achieved state-of-the-art performance and attracted significant attention. KGCN [7] and KGNN-LS [14] obtain the embedding of an entity in KG by uniformly aggregating the embeddings of its same-relation neighbors. Although they aggregate neighbors of different relations with different weights, they completely ignore intra-relation semantics. Besides, they do not explicitly model the user-item graph and hence might be inadequate in extracting collaborative signals. KGAT [8] incorporates user-item interactions into a KG by representing each interaction as a triple (*user, Interact, item*). It then models high-order relations with graph attention networks on the extended KG. However, its attention mechanism does not model user-specific preferences on entities and relations. Also, the collaborative signals from user-item interactions may be overwhelmed by irrelevant attributes from the KG, especially in scenarios where user-item interactions are sparse (as demonstrated in Sections 5.2 and 5.3 empirically).

To address these issues, in this paper, we propose a novel framework with **co**llaborative and **at**tentive graph convolutional networks (COAT) for personalized knowledge-aware recommendation. To effectively capture both the collaborative signals and the attribute-based relations, COAT models the user-item graph and the KG separately and simultaneously with two different graph neural networks (GNNs). Since the user-item graph and the KG have distinct structural properties and connectivity patterns (e.g., community structure and degree distribution), they need to be modeled by GNNs with different design features.

In this regard, our COAT framework offers greater flexibility than existing models, especially KGAT that forms an extended KG by integrating user-item interactions and models it with a single graph neural network. It also has clear advantages over KGCN and KGNN-LS, which fail to model high-order user-item interactions. In particular, on the user-item graph, an efficient graph convolutional network is leveraged to encode high-order collaborative signals for learning effective user and item embeddings and tackling the sparsity issue.

This gives another advantage of our COAT framework in handling very sparse user-item graphs (as shown in Section 5.3). On the KG, a knowledge graph attention network is proposed for learning entity embeddings, where each layer is equipped with a novel personalized KG-aware attention mechanism for capturing user-specific fine-grained semantics.

The proposed attention mechanism and variants jointly consider user features, target entity, entity relation, and neighbor entity to achieve personalization. Compared with the attention mechanisms used in KGCN and KGAT, it has the advantage in capturing more fine-grained semantic information for modeling user preferences. By making same entities (items) in the user-item graph and the KG share the same embeddings during training (Section 4.2), our model is capable to effectively capture high-order relations across the user-item graph and the KG with the cooperation of the two GNNs.

The main contributions of this paper include the following:

- We propose a new framework COAT based on collaborative and attentive graph convolutional networks for knowledge-aware recommendation, which effectively addresses the limitations of existing GNN-based methods and combines their advantages in a holistic system.

- We design novel personalized knowledge-aware attention mechanisms to capture user-specific fine-grained semantics in the KG to achieve more personalized recommendation.
- We conduct extensive experiments to evaluate our model COAT on four benchmark datasets for top-K recommendation and click-through rate prediction. The results demonstrate its competitive advantages over a variety of state-of-the-art methods.

The organization of this paper is as follows. In Section 2, we provide a review of current literature. In Section 3, we formulate the research problem. We present a detailed description of the proposed method in Section 4. The experimental results and analysis are provided in Section 5. Finally, a short summary is included in Section 6.

2. Related works

2.1. Graph neural networks

Graph neural networks, particularly graph convolutional neural networks generalize the convolutional operator defined on Euclidean data such as image (2D grid) and text (1D sequence) to non-regular graph-structured data [15]. They can be mainly categorized as spectral-based methods [16,17] and spatial-based methods [18,19]. Spectral-based methods define graph convolution in the Fourier domain based on graph signal processing theory [20,21]. Spectral CNN [16] exploits the spectrum of graph-Laplacian to define graph convolution filter, but suffers from the expensive computational complexity. ChebNet [22] approximates the graph filter with Chebyshev polynomials of the diagonal matrix of eigenvalues of graph-Laplacian. GCN [17] further simplifies it by leveraging a first-order approximation of ChebNet. Spatial-based methods define graph convolution based on a node's spatial relations. Representative models include GraphSAGE [18], GAT [19] and FastGCN [23]. More comprehensive analysis of GNNs can refer to the survey [15]. The above methods are mainly proposed for homogeneous graphs. Substantial efforts have also been devoted to designing GNN models for more complex graph-structured data, such as heterogeneous graphs [24,25] and knowledge graphs [26,27].

Recently, there are many practices in applying graph neural networks to model user-item interactions in recommender systems [28–30]. Both GCMC [28] and NGCF [30] apply graph convolutions for feature learning on the user-item graph, with the former having only one layer while the latter having multiple layers to explore high-order relations. PinSage [29] is an efficient method based on random walks and graph convolutions to generate item embeddings from the item-item graph. However, these methods can only model homogeneous graphs, whereas in this paper we consider enriching user-item interactions with a heterogeneous KG.

2.2. Knowledge-aware recommendation

We focus on leveraging KGs as item side information to address the data sparsity and cold start problems of collaborative filtering. KG-aware methods can be divided into four categories including regularization-based, path-based, hybrid, and GNN-based methods. Regularization-based methods [6,12] leverage KG embedding techniques to capture the semantic meanings among entities. They can flexibly incorporate different existing KG embedding losses for regularization, but such implicit way of modeling KG structure is less effective for improving recommendation performance. Path-based methods [10,11] are proposed to leverage heterogeneous information networks for recommendation. They require manually designed meta-paths to explore graph structures which are heavily dependent on domain knowledge. Hybrid methods such as PKGE [31], RippleNet [9] and AKUPM [32] combine the regularization-based methods and the path-based methods. They automatically learn paths connecting users and items to improve the prediction of user preferences.

Table 1
Comparison of the attention mechanisms used in different methods. (√means meeting the description; ✗means not meeting the description).

	Entity-aware	Relation-aware	User-aware
GAT	√	✗	✗
KGCN	✗	√	√
KGAT	√	√	✗
COAT (ours)	√	√	√

Most recently, graph neural networks have been applied to fully explore KG structures [7,8,14] and achieved promising performance in KG-based recommendation. KGCN [7] and KGNN-LS [14] convert the KG into a weighted graph by learning user-specific relation scores.

However, they fail to capture intra-relation semantics since they aggregate neighbors of the same relation with the same weight. Moreover, they do not explicitly model user-item connections and cannot catch high-order collaborative signals. In our proposed COAT framework, we explicitly model the user-item graph with an efficient graph convolutional network, which enables to effectively model high-order collaborative signals and tackle the sparsity issue. KGAT [8] incorporates user-item interactions into the KG and then models high-order connections with graph attention networks.

However, the collaborative signals in user-item interactions can be easily overwhelmed by irrelevant attributes from the KG. Moreover, it lacks user-specific personalization and suffers from high computational complexity. In our COAT framework, the UI graph and the KG are modeled separately to fully explore their different connectivity patterns and semantics, and jointly utilized for user preference prediction.

2.3. Attention mechanism

Attention is one of the most powerful techniques in deep learning. It allows the model to focus on the most informative parts of the inputs dynamically and offers excellent explainability in handling image and text data [33,34]. It has also been successfully used in graph neural networks [7,8,19]. Table 1 presents a comparison of the attention mechanisms proposed in our model COAT to those in other GNN-based methods. Specifically, the attention mechanisms are categorized based on their capability in learning attention scores for triples in the KG. Accordingly, entity-aware, relation-aware and user-aware means the attention scores are related to entity, relation and user embeddings, respectively. As the pioneering work, GAT [19] proposes an attention mechanism to learn node-aware attention scores on homogeneous graphs. When applied in the KG, it can learn entity-aware attention scores. Research efforts have also been made in designing attention mechanisms for multi-relational data such as KGs [7,8,35] and general heterogeneous graphs [24]. Here, we focus on methods for KGs. KGCN [7] proposes to learn user-specific relation preference by considering user embeddings and relation embeddings in learning the attention scores. KGAT [42] combines the user-item graph and the KG into an extended KG by representing each user-item interaction as a triple (u, *Interact*, i). It uses an attention mechanism that jointly considers head, tail and relation embeddings. However, it can only learn user-specific scores for some triples (user-item interactions) in the extended KG, but not for other triples from the original KG. Thus, KGAT is not a user-aware model. In comparison, our proposed COAT can learn user-specific triple-aware attentions for feature aggregation on the KG.

3. Problem definition

3.1. User-item graph (UI)

We consider implicit feedback data as in many recommendation methods [6,8]. Specifically, there are a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ and a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_M\}$. The interactions between users and items are represented as a rating matrix $\mathbf{Y} \in \mathbb{R}^{N\times M}$, where $y_{nm} = 1$ if user u_n has engaged with item i_m such as listening, reading or watching, otherwise $y_{nm} = 0$. $\mathbf{Y}$ can be considered as the biadjacency matrix of a bipartite graph between users and items. Based on this, we can construct a graph termed user-item graph with the adjacency matrix as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{Y} \\ \mathbf{Y}^\top & \mathbf{0} \end{bmatrix}. \tag{1}$$

The complex user-item connectivity patterns in the UI encode rich collaborative signals which are essential to capture user preferences.

3.2. Knowledge graph (KG)

A knowledge graph consists of a set of triples $\mathcal{G} = \{(e_h, r, e_t)|e_h, e_t \in \mathcal{E}, r \in \mathcal{R}\}$, where e_h, r and e_t represent the head entity, relation and tail entity, respectively. $\mathcal{E}$ represents the entity set and $\mathcal{R}$ represents the relation set. The KG can help to capture rich semantic relations and high-order structural connections of items.

There is a set of item-entity alignments $\mathcal{A} = \{(i, e)|i \in \mathcal{I}, e \in \mathcal{E}\}$ between items and entities, which bridges the user-item interactions and entity semantic relations in the KG. Here, without loss of generality, we assume that each item corresponds to an entity in the KG as in many existing works [6–8]. Based on the alignments, we denote the entity set of the KG as

$$\mathcal{E} = \{i_1, \ldots, i_M, e_{M+1}, \ldots, e_{|\mathcal{E}|}\}, \tag{2}$$

which means $\mathcal{I} \subset \mathcal{E}$.

Given a UI and the associated KG, the research problem of this paper is to predict the probability of a user u engaging with an item i in the future.

4. The proposed model

4.1. Model overview

In this paper, we propose a new framework with **co**llaborative and **at**tentive graph convolutional networks (COAT) to jointly model the user-item graph and the knowledge graph for recommendation. The model framework is shown in Fig. 2. It consists of an embedding learning component and a prediction component. In the embedding learning component, an efficient graph convolutional network preserves collaborative signals in the refined user and item embeddings from the UI, while a personalized knowledge graph attention network encodes rich semantic meanings of the KG into the refined item embeddings in a personalized manner. The prediction component takes the learned embeddings as inputs to predict the probability of user-item interaction.

4.2. Embedding layer

We transform the one-hop representations of users, items, entities, and relations into low-dimensional dense vectors by setting up an embedding lookup layer as in some existing works [6,11]. Specifically, the embedding matrix for users and items is denoted as $\mathbf{X_1} \in \mathbb{R}^{(N+M)\times d}$, where d is the dimension of embedding vectors, and the first N rows of $\mathbf{X_1}$ are the embeddings of users and the last M rows are the embeddings of items. The entity embedding matrix is denoted as $\mathbf{X_2} \in \mathbb{R}^{|\mathcal{E}|\times \mathbf{d}}$, where the first M rows are item embeddings.

By the assumption that each item corresponds to an entity in the KG, i.e., $\mathcal{I} \subset \mathcal{E}$, $\mathbf{X_1}$ and $\mathbf{X_2}$ share the same embedding vector for the same item/entity. Similarly, the relation embedding matrix is represented as $\mathbf{X}_3 \in \mathbb{R}^{|\mathcal{R}|\times d}$. Formally, given a one-hot representation $\mathbf{o_u} \in \mathbb{R}^{(N+M)\times 1}$ of user u, the initial embedding is obtained with

$$\mathbf{x}_u = \mathbf{X_1}^\top \cdot \mathbf{o_u}. \tag{3}$$

Similar operations can be applied to items, entities, and relations to obtain their initial embeddings.

4.3. Efficient graph convolutional layer

To capture the behavior information of users, we propose to aggregate the embeddings of user-interacted items for feature smoothing. Specifically, with one graph convolutional layer, the user embedding is obtained as follows:

$$\mathbf{h}_u = \sigma\left(\mathbf{W}^\top \sum_{i\in\mathcal{N}_u} \frac{1}{|\mathcal{N}_u|}\mathbf{x}_i\right), \tag{4}$$

where $\mathcal{N}_u$ represents the neighbor set of user u in the UI, $\mathbf{x}_i$ is the embedding of item i from the embedding layer, $\mathbf{W}$ is the weight matrix, and $\sigma(\cdot)$ is the activation function and set to LeakyReLU [36]. Since the embedding of the user itself is also very important, a self-loop is added on each node in the graph, i.e., $\mathcal{N}_u = \mathcal{N}_u \cup \{u\}$. Similarly, item embeddings can be learned in the same way.

To fully explore the collaborative signals in user-item interactions, it is important to model high-order connections in the UI explicitly as suggested in Wang et al. [30]. Therefore, we stack L_1 graph convolutional layers to learn user and item embeddings, which allows to aggregate the embeddings of L_1-hop neighbors. The embedding learning function in the lth layer can be written in the matrix form as follows:

$$\mathbf{H}^{(l)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{5}$$

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$, $\mathbf{H}^{(0)} = \mathbf{X}_1$, and $\mathbf{W}^{(l)}$ is the weight matrix in the lth layer. However, with the graph convolutional layer defined as in Eq. (4), one has to stack multiple layers to explore high-order connections explicitly, which will also increase model complexity because of the accompanied trainable parameters in each layer. This can easily cause overfitting in scenarios where user-item interactions are sparse. To address this issue, we propose to use an efficient graph convolutional layer [37] to control model capacity while still aggregating embeddings via high-order connections. Then, the hidden embeddings in the lth layer are obtained by:

$$\mathbf{H}^{(l)} = \sigma(\hat{\mathbf{A}}^s\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{6}$$

where s is the exponent of $\hat{\mathbf{A}}$. By setting an appropriate s, we can easily control the degree of utilizing high-order connections while avoiding overfitting. Normally, a large s should be used for sparse data. We denote the output embeddings from the final layer as $\mathbf{H} = \mathbf{H}^{(L_1)}$.

4.4. Personalized knowledge graph attention layer

To model the KG for recommendation, a plausible approach is to design graph neural networks that can automatically learn the importance weights of KG links and then perform embedding learning with graph convolution [7,8]. The challenges mainly lie in two aspects. Firstly, it is difficult to capture the semantic relations among entities due to the diversity of relation and entity

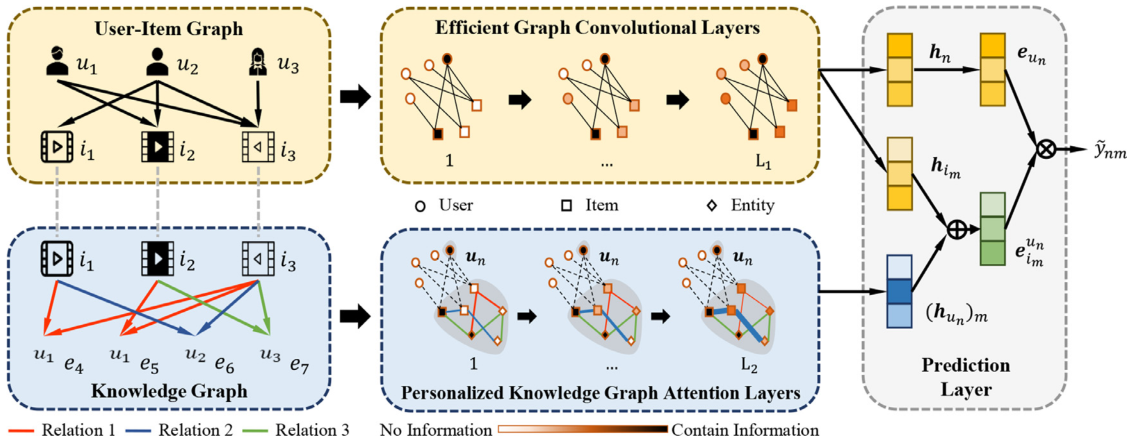

Fig. 2. The proposed framework for personalized knowledge-aware recommendation. The graph convolutional layers and graph attention layers aggregate and propagate information on the UI and the KG respectively, as conceptually illustrated by the change of node color.

types. Secondly, it is important yet challenging to take into account personalized user preferences when characterizing the relations among entities for embedding learning.

In the KG, one entity may connect to others through different relations. For example, as shown in Fig. 1, the movie Interstellar is connected to Matthew via “acting”, to Nolan via “directing”, and to science fiction via “genre”. Intuitively, different relations have different influences on characterizing an entity, and each user has his/her own preferences over different relations. Further, one entity may connect to multiple entities through the same relation, e.g., both Matthew (leading actor) and Michael (supporting actor) have played roles in the movie Interstellar. These neighboring entities should have different importance in characterizing the target entity Interstellar. Also, such importance might be different for different users due to their personal interests. Thus, it is necessary to learn personalized attention scores for embedding aggregation. Existing methods [7,8,19] cannot fully capture the above observations due to their flawed design. Specifically, both GAT [19] and KGAT [8] ignore user features, and thus are incapable of learning personalized attention scores. KGCN [7] does not consider head and tail features, and thus fails to characterize the importance of different neighbors especially those with the same relations. Therefore, it is necessary to consider features of user, relation, target entity and neighbor entity simultaneously in designing attention mechanisms so as to fully capture the above mentioned fine-grained semantics of KG for recommendation.

4.4.1. Personalized knowledge-aware attention

Based on the above observations, we design a novel personalized knowledge graph attention layer to learn item embeddings from the KG concerning a given user to provide more accurate, explainable and diverse recommendations. The key component is the personalized knowledge-aware attention model as illustrated in Fig. 3. Consider a tetrad (u, e_i, r_{ij}, e_j) of user, target entity, relation and neighbor entity. The model takes the embeddings of the tetrad (u, e_i, r_{ij}, e_j) as inputs to learn the attention score:

$$(a_u)'_{ij} = f(\mathbf{x}_u, \mathbf{x}_{e_i}, \mathbf{x}_{r_{ij}}, \mathbf{x}_{e_j}), \tag{7}$$

where $f(\cdot)$ is the attention function, and $\mathbf{x}_u$, $\mathbf{x}_{e_i}$, $\mathbf{x}_{r_{ij}}$ and $\mathbf{x}_{e_j}$ are the initial embeddings of user, target entity, relation and neighbor entity, respectively. By designing an attention mechanism to jointly model the tetrad (u_n, i_1, r_{14}, e_4), our method could capture more fine-grained semantics in KG than other methods. In this paper, we explore different ways to implement $f(\cdot)$ for learning the attention score:

- *Inner-Product Attention* extends the attention model of KGCN by also considering the features of the target and neighbor entities with the inner-product operator so as to learn personalized relation-aware and entity-aware attention scores, which is implemented as follows:

$$(a_u)'_{ij} = \mathbf{x}_u \cdot \mathbf{x}_{r_{ij}} + \mathbf{x}_{e_i} \cdot \mathbf{x}_{e_j}. \tag{8}$$

- *Sum Attention* first sums the embeddings of the tetrad (u, e_i, r_{ij}, e_j), and then obtains the attention score with a single-layer perceptron as follows:

$$(a_u)'_{ij} = \sigma\left(\mathbf{w}^\top(\mathbf{x}_u + \mathbf{x}_{e_i} + \mathbf{x}_{r_{ij}} + \mathbf{x}_{e_j}) + b\right), \tag{9}$$

  where $\mathbf{w}$ is the weight vector, b is the bias, and $\sigma(\cdot)$ is the activation function and set to LeakReLU.
- *Concat Attention* concatenates the embeddings of the tetrad (u, e_i, r_{ij}, e_j) first, and then leverages a single-layer perceptron to compute the attention score as follows:

$$(a_u)'_{ij} = \sigma\left(\mathbf{w}^\top(\mathbf{x}_u || \mathbf{x}_{e_i} || \mathbf{x}_{r_{ij}} || \mathbf{x}_{e_j}) + b\right). \tag{10}$$

- *Bi-Interaction Attention* extends the Sum attention by introducing a new component to explore feature interactions among the tetrad (u, e_i, r_{ij}, e_j) as follows:

$$\begin{aligned}(a_u)'_{ij} = {} & \sigma\left({\mathbf{w}_1}^\top(\mathbf{x}_u + \mathbf{x}_{e_i} + \mathbf{x}_{r_{ij}} + \mathbf{x}_{e_j}) + b_1\right) \\ & + \sigma\left({\mathbf{w}_2}^\top(\mathbf{x}_u \odot \mathbf{x}_{e_i} \odot \mathbf{x}_{r_{ij}} \odot \mathbf{x}_{e_j}) + b_2\right),\end{aligned} \tag{11}$$

  where $\mathbf{w}_1$ and $\mathbf{w}_2$ are the weight vectors, b_1 and b_2 are the biases, and $\odot$ is the element-wise product.
  The second term allows to sufficiently model interactions among features of the tetrad through element-wise product, which increases the flexibility and capability of the attention mechanism. As will be shown by the experiments in Section 5.3, Bi-Interaction attention can significantly improve the recommendation performance on sparse user-item interactions by modeling KG more effectively.

Finally, the attention score on triple (e_i, r_{ij}, e_j) with respect to user u is normalized with a softmax function as follows:

$$(a_u)_{ij} = \frac{\exp\left((a_u)'_{ij}\right)}{\sum_{e_k \in \mathcal{N}_{e_i}} \exp\left((a_u)'_{ik}\right)}, \tag{12}$$

where $\mathcal{N}_{e_i}$ denotes the neighbors of entity e_i.

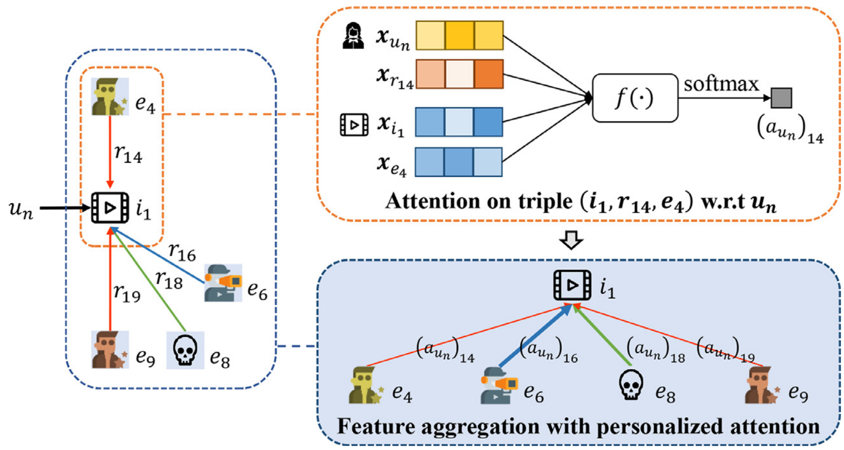

Fig. 3. Personalized knowledge-aware attention. To predict the interaction probability between user u_n and item i_1, we need to learn the personalized item embedding of i_1. The attention score between i_1 and e_4 is obtained from a personalized knowledge-aware attention model that jointly considers the embeddings of the tetrad (u_n, i_1, r_{14}, e_4).

4.4.2. Embedding aggregation with attention.

With the learned attention scores, the embedding of entity e_i with respect to user u in the first aggregation layer is computed as follows:

$$(\mathbf{h}_u)_{e_i} = \sigma\left(\mathbf{W}_{\mathcal{E}}{}^{\top} \sum_{e_j \in \mathcal{N}_{e_i} \cup \{e_i\}} (a_u)_{ij} \mathbf{x}_{e_j}\right), \tag{13}$$

where $(a_u)_{ii}$ is set to 1 to include the embedding of the entity itself, and $\sigma(\cdot)$ is the activation function and set to LeakyReLU.

We stack L_2 embedding aggregation layers to explore high-order connectivity patterns in KG. For a particular user u, the entity embeddings in the lth layer is computed as

$$\mathbf{H}_u^{(l)} = \sigma(\mathbf{A}_u \mathbf{H}_u^{(l-1)} \mathbf{W}_{\mathcal{E}}^{(l)}), \tag{14}$$

where $\mathbf{A}_u$ is a weighted adjacency matrix learned with the proposed personalized knowledge-aware attention model, and $(a_u)_{ij}$ is an entry of $\mathbf{A}_u$ with index (i, j). We denote $\mathbf{H}_u = \mathbf{H}_u^{(L_2)}$ as the output entity representations with respect to user u. To improve model efficiency, we fix the neighborhood size as T for each entity and sample from the original neighbor set uniformly for embedding aggregation as in Wang et al. [7], Hamilton et al. [18].

4.5. Interaction prediction

The refined user embeddings are obtained from the UI, with the embedding of user u_n as

$$\mathbf{e}_{u_n} = \mathbf{h}_n, \tag{15}$$

where $\mathbf{h}_n^{\top}$ is the nth row of $\mathbf{H}$. The refined item embeddings are a combination of those from the UI and the KG. Specifically, the refined embedding of item i_m with respect to user u_n is:

$$\mathbf{e}_{i_m}^{u_n} = \alpha \mathbf{h}_{N+m} + (1-\alpha)(\mathbf{h}_{u_n})_m, \tag{16}$$

where $\alpha \in [0, 1]$ is a balancing parameter and set to 0.5 in our experiments, $\mathbf{h}_{N+m}^{\top}$ is the $(N+m)$th row of $\mathbf{H}$ and $(\mathbf{h}_{u_n})_m$ is the mth row of $\mathbf{H}_{u_n}$. The item embedding thus contains both collaborative signals from the UI and attribute-based information from the KG.

With the learned user and item embeddings, the interaction probability between user u_n and item i_m is computed as:

$$\tilde{y}_{nm} = g(\mathbf{e}_{u_n}, \mathbf{e}_{i_m}^{u_n}), \tag{17}$$

where $g(\mathbf{e}_1, \mathbf{e}_2) = \sigma(\mathbf{e}_1^{\top}\mathbf{e}_2)$ with $\sigma(\cdot)$ set to the sigmoid function. Note that neural network based models [38] can be utilized to explore more complicated interactions of the embeddings.

4.6. Optimization and computational complexity

We train our model with the cross entropy loss as follows:

$$\mathcal{L} = \sum_{u_n \in \mathcal{U}} \sum_{i_m : y_{nm}=1} -\log(\tilde{y}_{nm}) - \sum_{k=1}^{K} \mathbb{E}_{i_k \sim P_k(i)} \log(1 - \tilde{y}_{nk}), \tag{18}$$

where $P_k(i)$ is a negative sampling distribution and K is the number of negative items sampled for each observed user-item pair and set to 1 in the experiments.

Graph neural networks suffer from the overfitting issue just like other deep learning models [39]. To improve the generalization ability of COAT, we employ two regularization methods. Firstly, L2 norm loss is enforced on the model parameters. Secondly, node dropout and message dropout are leveraged in the GNN for the UI. Specifically, the output messages from Eqs. (5), (6) are set to 0 with a probability p_1 [40]. Besides, each edge in the UI is removed with a probability p_2 at each training step as suggested in Rong et al. [39]. This operation actually augments the randomness and the diversity of the input graph, thus improving model robustness against noisy connections and alleviating the overfitting issue.

The computation mainly comes from the two GNNs for the UI and the KG, respectively. Suppose that $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$, and $\|\mathbf{Y}\|_0$ is the number of user-item interactions. It takes $\mathcal{O}(\|\mathbf{Y}\|_0 d_{l-2} d_{l-1} d_l)$ to compute the hidden representations with a single graph convolutional layer on the UI through Eq. (5), which is linear to the number of user-item interactions. The runtime complexity of the efficient graph convolutional layer (Eq. (6)) is similar to the normal one, but with an additional constant scale factor s. To improve time and space complexity, for the GNN on the KG, we fix the neighborhood size of each entity as T and sample neighbors through uniform sampling as in Wang et al. [7], Hamilton et al. [18]. Suppose that $\mathbf{W}_{\mathcal{E}}^{(l)} \in \mathbb{R}^{d_{\mathcal{E}}^{(l-1)} \times d_{\mathcal{E}}^{(l)}}$. It takes $\mathcal{O}(\|\mathbf{Y}\|_0 T^{L_2} d_{\mathcal{E}}^{(l-2)} d_{\mathcal{E}}^{(l-1)} d_{\mathcal{E}}^{(l)})$ to obtain entity embeddings with Eq. (14) given $\mathbf{A}_u$, which is linear to the number of triples in the KG. The time complexity of computing $\mathbf{A}_u$ is $\mathcal{O}(\|\mathbf{Y}\|_0 T^{L_2} d^2)$ with d as the dimension of the initial embeddings and L_2 as the number of layers in the GNN. Therefore, the overall complexity of our COAT model is linear to the size of the UI and the KG.

5. Experiments

In this section, we empirically evaluate our proposed method COAT in two scenarios including click-through rate (CTR) prediction and top-K recommendation on benchmark datasets. We aim to answer the following research questions through experiments:

Table 2
Statistics of the datasets and settings of the hyper-parameters used in our mothod COAT (d: embedding size, $|\mathcal{B}|$: batch size, λ: L2 regularization weight, η: learning rate, s: coefficient of the efficient GCN layer, and T: neighborhood size for each entity in training.).

	Movie	Book	Music	Restaurant
# users	138,159	17,860	1872	2,298,698
# items	16,954	14,967	3846	1362
# interactions	13,501,622	69,873	42,346	23,416,418
# entities	102,569	77,903	9366	28,115
# relations	32	18	60	7
# KG triples	499,474	60,787	15,518	160,519
d	32	64	32	8
$\|\mathcal{B}\|$	65,536	256	128	65,536
λ	10^{-7}	2×10^{-5}	10^{-4}	10^{-7}
η	2×10^{-2}	5×10^{-5}	$5 \times 10{-}3$	2×10^{-2}
s	1	3	8	1
T	4	8	8	4

RQ1 How does COAT perform compared with state-of-the-art methods?
RQ2 How do different variants of COAT perform under scenarios with very sparse user-item interactions?
RQ3 How does the performance contribution of the proposed personalized knowledge-aware attention mechanisms?
RQ4 How does the effectiveness and efficiency of the efficient graph convolutional layer?
RQ5 How do the hyper-parameters affect the performance of COAT?

5.1. Experimental setup

5.1.1. Datasets

We conduct experiments with benchmark datasets including MovieLens-20M (Movie), Last.FM (Music), Dianping-Food (Restaurant), and Book-Crossing (Book), with the first three from [14] and the last one from [9]. The descriptive statistics of the datasets are presented in Table 2.

5.1.2. Baselines

We select state-of-the-art baselines from five categories including supervised learning methods (FM and NFM), regularization-based methods (CKE), path-based methods (PER), hybrid methods (RippleNet), and GNN-based methods (GCMC, NGCF, KGAT, KGCN and KGNN-LS). The descriptions are listed as follows:

- **FM, NFM** [5]: FM builds upon factorization machines to model second-order feature interactions. NFM improves upon FM by leveraging neural networks to model higher-order feature interactions.
- **CKE** [6]: This model regularizes matrix factorization based collaborative filtering with a KG embedding loss from TransR [13].
- **PER** [10]: It is a representative path-based method that manually designs meta-paths that connect users and items in a heterogeneous graph and extracts them to enrich user-item connections.
- **RippleNet** [9]: It is a hybrid model that combines the regularization-based and path-based methods, which extends the user's interests iteratively by utilizing his/her historically interacted items and KG links.
- **GCMC** [28], **NGCF** [30]: Both methods use graph convolutional networks to model the user-item graph. GCMC only uses a single convolutional layer, while NGCF stacks multiple layers to explore high-order connectivity.
- **KGAT** [8], **KGCN** [7], **KGNN-LS** [14]: These are the most recent state-of-the-art knoweledge-aware methods based on graph neural networks.

Table 3
Classification of possible results when recommending an item to a user in CTR prediction.

	Recommended	Not recommended
Clicked	True-Positive (TP)	False-Negative (FN)
Not Clicked	False-Positive (FP)	True-Negative (TN)

We denote COAT with the proposed Inner-Product, Sum, Concat and Bi-Interaction attentions as COAT-I, COAT-S, COAT-C and COAT-B, respectively.

5.1.3. Settings

We implement COAT using Tensorflow [41] with Adam optimizer [42]. Both the node dropout ratio p_1 and the message dropout ratio p_2 are set to 0.1. The settings of other hyper-parameters are shown in Table 2. For the baselines, we conduct grid search to find optimal settings as suggested in the corresponding papers. In experiments, the observed user-item interactions are randomly split into three sets with the ratio 6:2:2 for training, validation and testing, respectively, as in Wang et al. [7,14]. To reduce randomness and ensure fairness, we use the same random seed during sampling for all the methods, which makes sure they all use the same data splits. The validation set is used to find the optimal hyper-parameter settings.

In CTR prediction, one negative item is sampled for each user-item pair in both the validation and testing sets for evaluation. AUC and F1 scores are used as performance measures. CTR prediction can be considered as a binary classification problem, and recommendation results can be classified as in Table 3. Based on the counting of different types of results, the evaluation metrics are computed as follows:

$$\text{Precision} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}},$$
$$\text{Recall (True Positive Rate)} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}},$$
$$\text{False Positive Rate} = \frac{\#\text{FP}}{\#\text{FP} + \#\text{TN}}. \tag{19}$$

Then, F1 score can be computed as follows:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{20}$$

AUC is the area under the receiver operating characteristic (ROC) curve which is obtained by plotting the true positive rate against the false positive rate at various threshold settings [43]. It can be calculated as follows:

$$\text{AUC} = \frac{\sum_{x \in \mathcal{P}} \text{rank}_x - \frac{|\mathcal{P}| \cdot (|\mathcal{P}|+1)}{2}}{|\mathcal{X}|}, \tag{21}$$

where $\mathcal{X}$ is the test set, $\mathcal{P} \subset \mathcal{X}$ is the positive user-item pairs, and rank_x is the ranking position of the predicted score of sample x among all samples in ascending order.

In top-K recommendation, Recall@K is selected as the evaluation metric for recommendation performance. For a given user, Recall@K is computed as follows:

$$\text{Recall@}K = \frac{\#\text{TP@}K}{\#\text{TP@}K + \#\text{FN@}K}, \tag{22}$$

where #TP@K and #FN@K represent the number of true-positive results and false-negative results respectively in the top-K items with the highest prediction scores among all items with respect to the given user. The final Recall@K score is obtained by taking the average of the Recall@K scores of all users.

All the experiments are run on a 128 G RAM computer with CPU Inter Xeon Silver 4110 2.10 GHz and GPU NVIDIA RTX 2080Ti.

Table 4
The results of Recall@K in top-K recommendation on movie and book.

Model	Movie				Book			
	R@2	R@10	R@50	R@100	R@2	R@10	R@50	R@100
FM	0.0364	0.1250	0.3183	0.4470	0.0029	0.0389	0.1421	0.1949
NFM	0.0299	0.1284	0.3019	0.4258	0.0031	0.0622	0.1418	0.1982
CKE	0.0351	0.1284	0.3333	0.4668	0.0041	0.0208	0.0528	0.0686
PER	0.0220	0.0770	0.1600	0.2430	0.0220	0.0410	0.0640	0.0700
RippleNet	0.0112	0.0867	0.2898	0.4216	0.0279	0.0449	0.0836	0.1154
GCMC	**0.0554**	**0.1657**	**0.4164**	**0.5540**	0.0000	0.0428	0.0673	0.0756
NGCF	0.0336	0.0962	0.2365	0.3064	0.0109	0.0174	0.0478	0.0656
KGAT	0.0371	0.1479	0.3544	0.4790	0.0276	0.0529	0.1056	0.1327
KGCN	0.0306	0.1271	0.3034	0.4428	0.0328	0.0528	0.1007	0.1329
KGNN-LS	0.0351	0.1336	0.3127	0.4056	0.0317	0.0584	0.1083	0.1255
COAT-I	**0.0452**	0.1461	**0.4004**	**0.5513**	**0.0547**	**0.0831**	**0.1480**	**0.2068**
COAT-S	**0.0465**	**0.1656**	**0.4015**	**0.5471**	**0.0547**	**0.0814**	**0.1528**	**0.2095**
COAT-C	0.0418	**0.1590**	0.3853	0.5350	**0.0550**	0.0780	0.1449	**0.2089**
COAT-B	0.0400	0.1557	0.3972	0.5387	**0.0581**	**0.0814**	**0.1596**	0.1928

* The top 3 results for each task are highlighted in bold.

Table 5
The results of Recall@K in top-K recommendation on music and restaurant.

Model	Music				Restaurant			
	R@2	R@10	R@50	R@100	R@2	R@10	R@50	R@100
FM	0.0061	0.0604	0.1616	0.2180	0.0276	0.1082	0.3737	**0.5488**
NFM	0.0175	0.0691	0.1640	0.2509	0.0235	0.1096	0.3799	0.5231
CKE	**0.0457**	**0.1556**	**0.3281**	**0.3969**	**0.0344**	0.1002	0.3399	0.4775
PER	0.0140	0.0520	0.1160	0.1760	0.0230	0.1020	0.2560	0.3540
RippleNet	0.0208	0.0596	0.1782	0.2875	0.0309	0.1025	0.3178	0.4960
GCMC	0.0000	0.0008	0.0749	0.2097	**0.0368**	0.1268	0.3515	0.5345
NGCF	**0.0442**	0.1245	0.2803	0.3499	0.0302	0.1167	0.3056	0.4508
KGAT	0.0324	0.1147	0.2916	0.3844	–	–	–	–
KGCN	0.0331	0.1175	0.2627	0.3500	0.0305	0.0980	0.3489	0.4808
KGNN-LS	0.0229	0.1067	0.2512	0.3365	0.0197	0.0904	0.3526	0.4943
COAT-I	0.0406	**0.1322**	**0.2992**	0.3934	**0.0343**	0.1193	0.3833	0.5368
COAT-S	0.0403	**0.1310**	0.2815	**0.3977**	0.0335	**0.1276**	**0.3945**	0.5292
COAT-C	**0.0519**	0.1157	0.2835	0.3867	0.0256	**0.1289**	**0.3910**	**0.5459**
COAT-B	0.0377	0.1258	**0.3044**	**0.4077**	0.0227	**0.1269**	**0.3903**	**0.5489**

* The top 3 results for each task are highlighted in bold.

All the reported results are the average of three different runs following the current practice [7,14]. Note that the results of KGAT on Restaurant are not presented in both experiments. The reason is that it needs to be trained in full-batch mode with similar space complexity as vanilla GCN [17], which requires expensive memory consumption for large-scale datasets such as Restaurant and exceeds the capacity of GPU memory of our computer.

5.2. Performance comparison (RQ1)

Top-K recommendation
Top-K recommendation aims to identify a set of K items that will be of interest to a certain user [44], and K is usually a small number in many real applications. The experimental results of top-2, 10, 50 and 100 recommendations are shown in Tables 4 and 5. All variants of our method COAT achieve better performance on average than the baselines across all the datasets, except for GCMC on Movie and CKE on Music.

All variants of COAT consistently outperform GNN-based knowledge-aware methods including KGAT, KGCN and KGCN-LS. For example, COAT-I, S, C and B respectively achieve significant 12.17%, 16.20%, 10.15% and 9.41% relative improvements over KGAT on Movie in average of the four top-K settings. One reason might be that COAT combines collaborative signals and attributed information in a more effective way. KGCN and KGNN-LS perform similarly on all the datasets. Our COAT-I achieves significant 25.52%, 54.08%, 34.30% and 30.85% relative gains over KGNN-LS on Movie, Book, Music and Restaurant respectively in average of the four top-K settings. Other variants of COAT also have similar performance improvements over KGNN-LS. The reasons that COAT significantly outperforms KGCN and KGNN-LS may lie in two aspects: 1) COAT explicitly models connectivity information in both the UI and the KG; 2) The proposed personalized knowledge-aware attention mechanisms can capture more fine-grained semantics in the KG.

COAT outperforms the baselines by a large margin on Book and achieves the best performance on Restaurant. On Movie, COAT performs comparably to the best baseline GCMC and significantly better than other knowledge-aware baselines, which further confirms that COAT can leverage the KG more effectively. Although GCMC achieves the best results on Movie, it performs terribly on Book and Music. Since GCMC only models the UI, it suggests that item-side information in the KG may not always be helpful, but utilizing it properly can help to achieve more stable and better performance. On Music, COAT also achieves competitive performance with the best baseline CKE. Although CKE achieves the best top-10 and top-50 recommendation results on Music, it does not perform well on other datasets, and especially its performance on Book is poor. Both FM and NFM perform much worse than COAT in most cases, which shows the clear advantages of COAT in directly modeling the complicated relations among users and items.

CTR prediction
CTR prediction aims to estimate the probability of a certain user clicking a displayed item [45]. Different from top-K recommendation, it needs to rank all the items in the item space for the user. The experimental results are presented in Table 6. It shows

Table 6
The results of AUC and F1 scores in CTR prediction.

Model	Movie		Book		Music		Restaurant	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
FM	0.9816	0.9394	0.7411	**0.6841**	0.7840	0.7329	0.8735	**0.8027**
NFM	0.9778	0.9325	**0.7538**	**0.6929**	0.7867	0.7312	0.8749	**0.8036**
CKE	0.9818	0.9411	0.6689	0.6225	0.7818	0.7098	0.8457	0.7692
PER	0.8320	0.7880	0.6170	0.5620	0.6330	0.5960	0.7460	0.7010
RippleNet	0.9743	0.9283	0.7263	0.6432	0.7938	0.7278	0.8696	0.7729
GCMC	**0.9858**	**0.9459**	0.6709	0.6393	0.7939	0.7310	0.8681	0.7997
NGCF	0.9617	0.9231	0.6453	0.5996	0.7953	0.7251	0.8461	0.7854
KGAT	0.9813	0.9383	0.7194	0.6600	0.8054	0.7389	–	–
KGCN	0.9770	0.9307	0.6883	0.6341	0.7604	0.6872	0.8450	0.7777
KGNN-LS	0.9788	0.9342	0.6882	0.6242	0.7661	0.6878	0.8449	0.7768
COAT-I	0.9840	0.9429	0.7437	**0.6800**	**0.8213**	0.7481	**0.8768**	0.7955
COAT-S	**0.9843**	**0.9434**	**0.7440**	0.6678	**0.8217**	**0.7499**	**0.8779**	**0.8009**
COAT-C	0.9840	0.9427	0.7436	0.6679	**0.8213**	**0.7491**	0.8766	0.7957
COAT-B	**0.9842**	**0.9431**	**0.7440**	0.6673	**0.8216**	**0.7490**	**0.8782**	0.7977

* The top 3 AUC and F1 scores are highlighted in bold.

Table 7
Recommendation results on music with 20% and 40% of the training set used for model training.

Model	20%			40%		
	R@10	R@50	AUC	R@10	R@50	AUC
NFM	0.0158	0.0710	0.7398	0.0308	0.1030	0.7576
CKE	0.0160	0.0513	0.5240	0.0385	0.0928	0.5867
NGCF	0.0372	0.1109	0.5809	0.0856	0.2214	0.7000
KGAT	0.0387	0.1324	0.7210	0.0587	0.1932	0.7527
KGCN	0.0309	0.0709	0.4974	0.0555	0.1456	0.5650
COAT-I	**0.0750**	0.1888	**0.7562**	**0.0860**	0.2036	**0.7697**
COAT-S	0.0729	**0.1976**	**0.7567**	0.0772	**0.2282**	**0.7709**
COAT-C	**0.0760**	**0.1956**	0.7560	**0.0834**	**0.2080**	0.7691
COAT-B	**0.0834**	**0.2237**	**0.7569**	**0.0822**	**0.2100**	**0.7705**

that our method COAT has overall the best performance on the four datasets. Specifically, it significantly outperforms all the baselines on Music, and has comparable performance with the best baselines on Music (GCMC), Book (FM, NFM) and Restaurant (FM, NFM). In particular, all four variants of COAT consistently and significantly outperform GNN-based knowledge-aware methods including KGAT, KGCN and KGNN-LS. For example, COAT-I achieves 0.53%, 8.06%, 7.21% and 3.78% relative improvements over KGNN-LS in AUC scores on Movie, Book, Music and Restaurant, respectively. It demonstrates the design advantages of our method COAT in exploiting the structural information and semantic information in the UI and the KG.

5.3. Performance on sparse user-item interactions (RQ2)

To evaluate the performance of our method on sparse user-item interactions, we keep the validation and test sets unchanged (as in Section 5.2), while randomly subsampling the interactions in the training set to achieve different levels of sparsity. Specifically, after subsampling, the number of users and items remain unchanged, while the number of user-item interactions is reduced to 20% or 40% of the original training set. Table 7 shows the results on Music. We only select several baselines for comparison to avoid repetition. The performance of CKE and KGCN drop dramatically in both top-*k* recommendation and CTR prediction compared with results in Tables 4–6, which is probably because they fail to model high-order connections in the user-item graph explicitly. NFM and NGCF also have significant performance loss under these scenarios. KGAT performs better than other baselines, since it explicitly models high-order connections in both the UI and the KG. However, it is still much worse than our COAT, which may be because the sparse collaborative signals in the UI can be easily overwhelmed by the irrelevant attributes from the KG and thus harms the performance. Overall, our method COAT achieves the best performance. The success of COAT in addressing the sparsity issue might be attributed to two reasons: firstly, the efficient graph convolutional layer directly helps to tackle the data sparsity issue; secondly, the design of two GNNs for the UI and the KG respectively improves the exploration of collaborative signals and attribute-based semantics. In addition, among all variants of COAT, COAT-B achieves the best results overall, which is probably due to the flexibility and capability of Bi-Interaction attention in modeling feature interactions.

5.4. Effect of attention mechanisms (RQ3)

In this subsection, we study the effect of the proposed personalized knowledge-aware attention mechanisms on recommendation performance. Fig. 4 shows the experimental results of COAT with 6 different attention mechanisms. Specifically, “avg” means aggregating the embeddings of a node’s neighbors with equal weights. “u_r” represents the attention mechanism that only considers user and relation embeddings as in KGCN [7]. “uhrt_ip”, “uhrt_add”, “uhrt_concat” and “uhrt_bi” represent our proposed attention mechanisms with Inner-Product, Sum, Concat and Bi-Interaction attention, respectively. From the results, it can be seen that COAT (u_r) performs better than COAT (avg) because the u_r attention can capture user-specific relation preference and importance in characterizing an entity. Further, COAT (uhrt_ip/add/concat/bi) all outperform COAT (u_r) consistently with only one exception being the Concat attention in top-50 recommendation on Movie. Overall, it shows the necessity of considering features of the tetrad (u, e_i, r_{ij}, e_j) in designing attention mechanisms to learn more fine-grained attention for recommendation.

5.5. Study of the efficient graph convolutional layer (RQ4)

In this subsection, we study the effectiveness and efficiency of the efficient graph convolutional layer (Section 4.3). We vary the order s of the graph convolutional layer from 1 to 10 to study its effect on recommendation performance and computation time. Note that when $s = 1$, it is the normal graph convolutional layer. As shown in Fig. 5, when increasing s from 1 to 10, the recommendation performance on Music gradually increases and achieves the best results when $s = 8$, and then drops. It demonstrates the advantage of the efficient graph convolutional layer compared with the normal one under the same model capacity (number of trainable model parameters). In Table 8, we present the training time

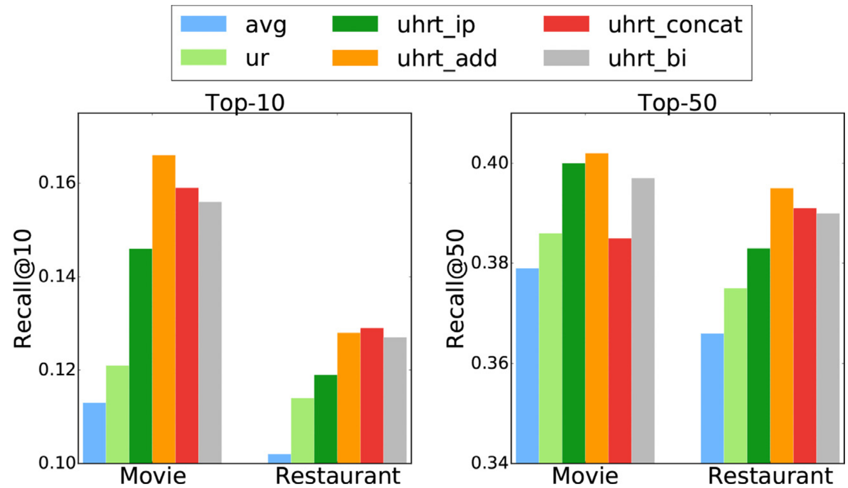

Fig. 4. Comparison of COAT with six different attention mechanisms on recommendation performance.

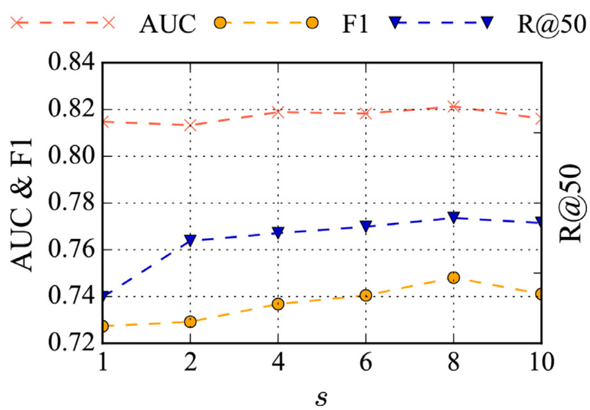

Fig. 5. Effect of the order s of the efficient graph convolutional layer on model performance.

Table 8
The training time (seconds) per epoch w.r.t. the order s of the efficient graph convolutional layer.

s	1	2	4	6	8	10
Music	0.775	0.774	0.812	1.597	1.616	1.670
Book	3.070	3.299	3.878	4.377	5.039	5.721
Movie	63.407	66.397	79.368	82.875	83.112	84.043

per epoch on Music, Book and Movie, where it can be seen that the time cost is sublinear w.r.t. the order s, demonstrating the computational efficiency of the graph convolutional layer. We also provide a comparison of the training time per epoch of different methods in Table 9. The training time of COAT under the default setting is reported. Overall, COAT is quite efficient compared with other methods, especially on large-scale datasets. For example, on the Movie dataset, COAT is more than ten times faster than KGAT (a state-of-the-art GNN-based method) and RippleNet (a path- and regularization-based method), and has comparable time efficiency with other methods.

5.6. Hyper-parameter sensitivity (RQ5)

In this subsection, we perform a sensitivity analysis of COAT w.r.t. the embedding size d, the combination coefficient α, the neighborhood size T, the coefficient λ of L2 regularization, the node dropout ratio p_1, and the message dropout ratio p_2. When studying one hyper-parameter, we fix all the others with the default settings. The experimental results in CTR prediction and top-50 recommendation on Music are shown in Figs. 6 and 7. Here, we only present the results of COAT-I for brevity, but similar results have been observed for other variants of COAT.

For the embedding size, when it increases from 8 to 128, the performance of COAT-I improves at first and then drops. The reason might be that too large a dimension of embeddings can easily cause overfitting. For the coefficient α, when it is set to 0.5, COAT-I achieves the best results, which confirms that both the collaborative signals in the UI and the knowledge-aware semantics in the KG are important for recommendation. For the neighborhood size of each entity in training, a small number such as 8 already yields quite good results; too large a value may bring in noises, thus harming the performance.

For L2 regularization, we vary the hyperparameter $\lambda \in \{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$. It can be observed in Fig. 7 that the recommendation performance gradually improves and reaches maximum at $\lambda = 1 \times 10^{-3}$ and then drops. For dropout regularization, as shown in Fig. 7, setting both node dropout ratio and message dropout ratio to 0.1 achieves the best performance, while too small or large a value can result in performance drops. Overall, it shows that an appropriate setting of the hyperparameters of the regularization methods including L2 norm,

Table 9
The training time (in seconds) per epoch of different methods.

Model	FM	NFM	CKE	RippleNet	KGCN	KGNN-LS	KGAT	COAT
Music	0.137	0.143	0.803	3.666	0.373	0.526	1.626	1.616
Book	0.561	0.611	6.582	11.111	0.886	2.504	18.491	3.613
Movie	111.965	115.200	34.657	877.521	17.282	29.401	1743.998	63.407

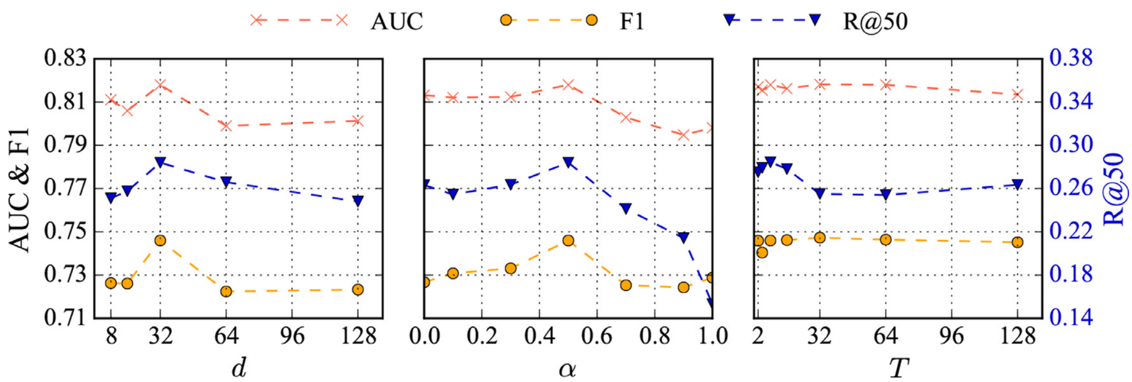

Fig. 6. Sensitivity analysis of the hyper-parameters including embedding size d, embedding combination coefficient α, and neighborhood size T.

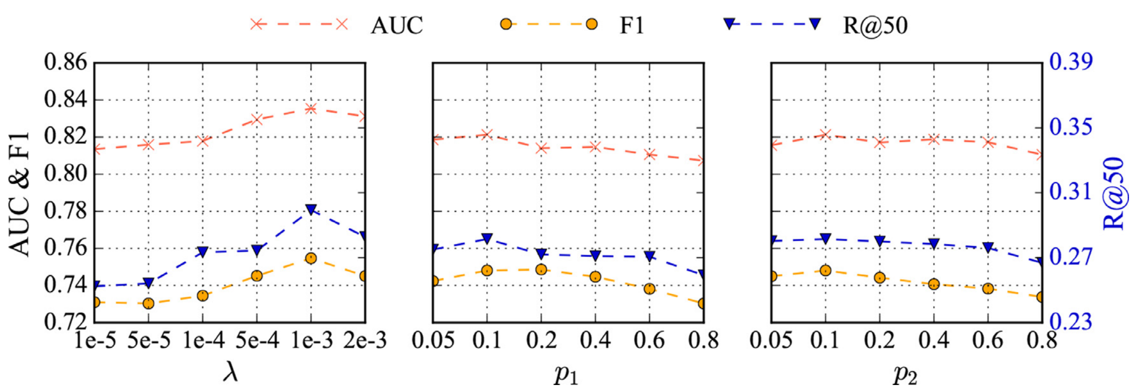

Fig. 7. Sensitivity analysis of the hyper-parameters including the coefficient λ of L2 regularization, node dropout ratio p_1, and message dropout ratio p_2.

node dropout and message dropout, can effectively alleviate overfitting and improve recommendation performance.

6. Conclusion

In this paper, we have proposed a novel and effective framework with collaborative and attentive graph convolutional networks (COAT) for personalized knowledge-aware recommendation. COAT is composed of two embedding learning modules: an efficient graph convolutional module for extracting collaborative signals from the user-item graph and a personalized knowledge graph attention module for capturing fine-grained semantics in the KG.

The strengths of our method over existing works lie in its greater flexibility in modeling both the user-item graph and the knowledge graph, its stronger attention mechanism to capture user-specific fine-grainded semantics, and its use of efficient graph convolutional layer to tackle data sparsity. We have conducted an extensive and fair empirical study by making comparison to 10 state-of-the-art baselines of five different kinds on four benchmark datasets. Our model COAT has achieved the best overall performance, which demonstrates its effectiveness in jointly modeling the user-item graph and the KG for quality recommendation.

We consider this work as a solid step towards personalized knowledge-aware recommendation. In the future, we plan to design noise-robust knowledge-aware recommender systems, investigate causal explanation in addition to relational explanation for recommendation results, and further improve model efficiency for very large-scale industrial applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the grants of P0001175 (ZVJJ) and P0030935 (ZVPY) funded by PolyU (UGC) and P0034058 (ZGAL) and P0038850 (ZGD1) funded by [1–45] Alibaba Group.

References

[1] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: ICDM, IEEE, 2008, pp. 263–272.
[2] X. Yu, F. Jiang, J. Du, D. Gong, A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains, Pattern Recognit. 94 (2019) 96–109.
[3] P. Bai, Y. Ge, F. Liu, H. Lu, Joint interaction with context operation for collaborative filtering, Pattern Recognit. 88 (2019) 729–738.
[4] S. Rendle, Z. Gantner, C. Freudenthaler, L. Schmidt-Thieme, Fast context-aware recommendations with factorization machines, in: SIGIR, 2011, pp. 635–644.
[5] X. He, T. Chua, Neural factorization machines for sparse predictive analytics, in: SIGIR, 2017, pp. 355–364.
[6] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W. Ma, Collaborative knowledge base embedding for recommender systems, in: KDD, 2016, pp. 353–362.
[7] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: WWW, 2019, pp. 3307–3313.
[8] X. Wang, X. He, Y. Cao, M. Liu, T. Chua, KGAT: knowledge graph attention network for recommendation, in: KDD, 2019, pp. 950–958.
[9] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, RippleNet: propagating user preferences on the knowledge graph for recommender systems, in: CIKM, 2018, pp. 417–426.
[10] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, J. Han, Personalized entity recommendation: a heterogeneous information network approach, in: WSDM, 2014, pp. 283–292.
[11] B. Hu, C. Shi, W.X. Zhao, P.S. Yu, Leveraging meta-path based context for top- N recommendation with A neural co-attention model, in: KDD, 2018, pp. 1531–1540.
[12] Y. Cao, X. Wang, X. He, Z. Hu, T. Chua, Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences, in: WWW, 2019, pp. 151–161.
[13] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: AAAI, 2015, pp. 2181–2187.
[14] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, Z. Wang, Knowledge-aware graph neural networks with label smoothness regularization for recommender systems, in: KDD, 2019, pp. 968–977.

[15] Z. Zhang, P. Cui, W. Zhu, Deep learning on graphs: a survey, IEEE Trans. Knowl. Data Eng. 34 (2020) 249–270.
[16] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, ICLR, 2014.
[17] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ICLR, 2017.
[18] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, NIPS, 2017.
[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, ICLR, 2018.
[20] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Process. Mag. 30 (3) (2013) 83–98.
[21] S. Chen, R. Varma, A. Sandryhaila, J. Kovacevic, Discrete signal processing on graphs: sampling theory, IEEE Trans. Signal Process. 63 (24) (2015) 6510–6523.
[22] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: NIPS, 2016, pp. 3837–3845.
[23] J. Chen, T. Ma, C. Xiao, FastGCN: fast learning with graph convolutional networks via importance sampling, ICLR, 2018.
[24] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: WWW, 2019, pp. 2022–2032.
[25] C. Zhang, D. Song, C. Huang, A. Swami, N.V. Chawla, Heterogeneous graph neural network, in: KDD, 2019, pp. 793–803.
[26] M.S. Schlichtkrull, T.N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: ESWC, 2018, pp. 593–607.
[27] S. Vashishth, S. Sanyal, V. Nitin, P.P. Talukdar, Composition-based multi-relational graph convolutional networks, ICLR, 2020.
[28] R. van den Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, KDD 2018 Deep Learning Day, 2018.
[29] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: KDD, 2018, pp. 974–983.
[30] X. Wang, X. He, M. Wang, F. Feng, T. Chua, Neural graph collaborative filtering, in: SIGIR, 2019, pp. 165–174.
[31] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: RecSys, 2018, pp. 297–305.
[32] X. Tang, T. Wang, H. Yang, H. Song, AKUPM: attention-enhanced knowledge-aware user preference model for recommendation, in: KDD, 2019, pp. 1891–1899.
[33] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, ICLR, 2015.
[34] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio, Show, attend and tell: neural image caption generation with visual attention, in: ICML, 2015, pp. 2048–2057.
[35] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, in: ACL, 2019, pp. 4710–4723.
[36] A.L. Maas, Rectifier nonlinearities improve neural network acoustic models, ICML, 2013.
[37] Q. Li, X. Wu, H. Liu, X. Zhang, Z. Guan, Label efficient semi-supervised learning via graph filtering, CVPR, 2019.
[38] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in: WWW, 2017, pp. 173–182.
[39] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: towards deep graph convolutional networks on node classification, ICLR, 2020.
[40] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P.A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: a system for large-scale machine learning, in: OSDI, USENIX Association, 2016, pp. 265–283.
[42] J.B. Diederik P. Kingma, Adam: a method for stochastic optimization, ICLR, 2015.
[43] D. Bamber, The area above the ordinal dominance graph and the area below the receiver operating characteristic graph, J. Math. Psychol. 12 (4) (1975) 387–415.
[44] M. Deshpande, G. Karypis, Item-based top-N recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.
[45] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, DeepFM: a factorization-machine based neural network for CTR prediction, in: IJCAI, 2017, pp. 1725–1731.

Quanyu Dai is currently a researcher at Huawei Noah's Ark Lab. He received the Ph.D. degree at The Hong Kong Polytechnic University. His research interests include graphbased algorithms and recommender systems. He has publications appeared in the toptier journals and conferences, such as TKDE, TNNLS, IJCAI, AAAI and WWW.

Xiao-Ming Wu is currently an assistant professor at the Department of Computing, The Hong Kong Polytechnic University. She works on machine learning and artificial intelligence. Her research has contributed to new theoretical insights of machine learning algorithms and novel methods for artificial intelligence applications in various fields. She publishes in leading venues including NeurIPS, CVPR, EMNLP, IJCAI and AAAI.

Lu Fan is currently working toward the M.Phil. degree at the Department of Computing, The Hong Kong Polytechnic University. Her research interests include natural language understanding and graph-based applications.

Qimai Li is currently a Ph.D. student at the Department of Computing, The Hong Kong Polytechnic University. He obtained B.Eng. degree from the College of Computer Science and Technology, Zhejiang University in 2017. His main area of interest includes machine learning on graph, semi-supervised Learning and graph signal processing.

Han Liu is currently an associate professor in the School of Software, Dalian University of Technology. He has published over 20 papers on journals and conferences, including TKDE, Neural Networks, ACL, CVPR, IJCAI, AAAI, etc. His research interests include machine learning, artificial intelligence and data mining.

Xiaotong Zhang is currently an associate research professor in the School of Software, Dalian University of Technology. She has published over 20 papers on journals and conferences, including TKDE, Neural Networks, ACL, CVPR, IJCAI, AAAI, etc. Her research interests include machine learning, artificial intelligence and data mining.

Dan Wang is currently an associate professor at Department of Computing, The Hong Kong Polytechnic University. He is an expert in computer networking, and he is recently working in the inter-discipline domains of smart energy systems, industry 4.0. He publishes extensively in top networking conferences, such as SIGCOMM, SIGMETRICS, INFOCOM and in top inter-discipline conference, such as ACM e-Energy, ACM Buildsys.

Guli Lin is currently an Engineer at Alibaba Group. His research interests are mainly in applying machine learning for Recommendation and Product Search. He received his Ph.D. degree in computer science from South China University of Technology.

Keping Yang is currently a senior algorithm expert at Search and Recommendation Department of Alibaba. She is working on machine learning and artificial intelligence since she received her Master degree from Zhejiang University. Her team publishes many papers in top conferences, such as KDD, SIGMOD, WWW, IJCAI, AAAI, etc.