MetaKG: Meta-learning on Knowledge Graph for Cold-start Recommendation

Yuntao Du, Xinjun Zhu, Lu Chen, Ziquan Fang, Yunjun Gao, *Member, IEEE*

Abstract—A knowledge graph (KG) consists of a set of interconnected typed entities and their attributes. Recently, KGs are popularly used as the auxiliary information to enable more accurate, explainable, and diverse user preference recommendations. Specifically, existing KG-based recommendation methods target modeling high-order relations/dependencies from long connectivity user-item interactions hidden in KG. However, most of them ignore the cold-start problems (i.e., user cold-start and item cold-start) of recommendation analytics, which restricts their performance in scenarios when involving new users or new items. Inspired by the success of meta-learning on scarce training samples, we propose a novel meta-learning based framework called MetaKG, which encompasses a collaborative-aware meta learner and a knowledge-aware meta learner, to capture *meta* users' preference and entities' knowledge for cold-start recommendations. The collaborative-aware meta learner aims to locally aggregate user preferences for each preference learning task. In contrast, the knowledge-aware meta learner is to globally generalize knowledge representation across different user preference learning tasks. Guided by two meta learners, MetaKG can effectively capture the high-order collaborative relations and semantic representations, which could be easily adapted to cold-start scenarios. Besides, we devise a novel adaptive task scheduler which can adaptively select the informative tasks for meta learning in order to prevent the model from being corrupted by noisy tasks. Extensive experiments on various cold-start scenarios using three real datasets demonstrate that our presented MetaKG outperforms all the existing state-of-the-art competitors in terms of effectiveness, efficiency, and scalability.

Index Terms—Cold-start Recommendation, Meta-learning, Knowledge Graph, Graph Neural Networks

✦

1 INTRODUCTION

WITH the explosive use of the Internet, recommender systems are widely deployed in real-life applications like Twitter and Weibo, in order to provide personalized recommendation services while addressing information overload issues [8], [45], [46], [50]. *User preference recommendation* aims to predict the preferences for target users based on the observed user-item interactions [18], [23], [52], which is a fundamental functionality among the current recommendation studies [6]. Despite the efforts of traditional recommendation methods (e.g., collaborative-filtering-based methods [15] and content-based methods [25]), they cannot mine the hidden high-order relations between users and items, thus restricting their effectiveness of user preference prediction. To this end, researchers start devoting knowledge graph (KG) based recommendations [36], [41], which combines the user-item interactions with the side information of KG to improve the recommendation performance.

Fig. 1 gives an example of movie recommendation on KG that consists of users, items, and entities, where items denote movies to be recommended and entities (e.g, actors, directors, and singers) are the side information provided by KG. In Fig. 1, there exists an interaction relation between the user u_1 and the movie i_1, where i_1 is *acted by* actor e_1 and is *directed by* director e_2. Traditional recommendations only focus on the short-connectivity user-item relations such as $u \xrightarrow{r} i$. That is, they only capture the low-order dependences for recommendations, e.g., only users who have watched the same movies (e.g., u_1 and u_2 watch i_1) tend to share similar preferences. Hence, the recommendation performance of
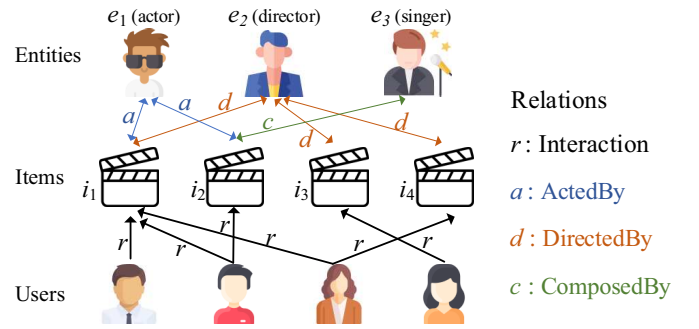

Fig. 1. Knowledge Graph Based Movie Recommendation

traditional methods is limited. In contrast, the KG-based approaches are capable of discovering the long connectivity relations based on the side information hidden in KG, e.g., $u_1 \xrightarrow{r} i_1 \xrightarrow{d} e_2 \xrightarrow{d} i_3 \xrightarrow{r} u_4$. In that case, users u_1 and u_4 are also likely to share similar preferences even when they have not watched any same movies. Overall, the KG-based recommenders can fully exploit user-item interactions for better recommendation analytics.

Despite their effectiveness, we argue that current KG-based methods only focus on improving the recommendation performance from the data level (i.e., leveraging knowledge associations to provide additional semantics beyond collaborative signals), which would inevitably experience a lack of information and poor performance in cold-start scenarios because of overfitting and popularity biases [37], [42]. Take the movie recommendation in Fig. 1 as an example. If we want to recommend movies for user u_4, the recommendation results may be unsatisfied, as u_4 is only interacted with movie i_3. Based on the long connectivity path

● *Y. Du, X. Zhu, L. Chen, Z. Fang, and Y. Gao (Corresponding Author) are with the College of Computer Science, Zhejiang University, Hangzhou 310027, China, E-mail:{ytdu, xjzhu, luchen, zqfang, gaoyj}@zju.edu.cn.*

$u_4 \xrightarrow{r} i_3 \xrightarrow{d} e_2 \xrightarrow{d} i_1$ and the popularity of movie i_1 among existing users, the recommenders tend to recommend i_1 as the interested movie for u_4. However, another possible recommendation path $u_4 \xrightarrow{r} i_3 \xrightarrow{d} e_2 \xrightarrow{d} i_4$ is ignored due to insufficient interactions of movie i_4.

Recently, the success of meta-learning sheds light on modeling tasks with scarce training samples [5], [10]. Specifically, meta-learning aims to learn prior knowledge from different tasks, which can be further employed to accommodate new tasks even when there are few data samples. Inspired by meta-learning, if we treat the preference learning for each user as a single task, we can model the preference learning for different users as different tasks under the meta-learning settings. Thus, the designed meta learner can derive *meta* user-item interactions with strong generalization, which can be employed to deal with new tasks such as cold-start users or cold-start items. Although a few studies [9], [56] utilized meta-learning for cold-start problems and have achieved promising performance, they are traditional methods without any KG information leverage. Specifically, they either consider the short connectivity relations among users and items [24] as discussed above, or rely on labor-intensive process to derive semantic facets [27]. Consequently, they cannot generalize generic prior knowledge for fast and accurate recommendations. To tackle the cold-start problems on KG-based recommendation, for the first time, we propose a meta-learning approach considering both user-item and KG. Two challenges exist as follows.

Challenge I: *How to efficiently learn the user preference when evolving the cold-start issues?* Existing KG-based recommendations highly rely on the massive user-item feedback, and thus, they suffer from dramatic performance degradation when the user-item interactions are scarce [37], [42]. This implies existing methods can not be directly adopted for our studied cold-start recommendations. Hence, it is necessary to redesign a learnable structure via meta-learning, which is able to capture the high-order *meta* collaborative information from limited user-item interactions.

Challenge II: *How to capture entities' prior knowledge of KG to provide more accurate recommendation in cold-start scenarios?* Existing KG-based methods demonstrate that the semantic embeddings of entities from KG are beneficial for recommendation analytics [43], [47]. However, none of those methods consider how to derive *meta* entities' knowledge with limited interactions. Thus, it is also challenging to make full use of the knowledge information (i.e., automatically learn strong generalizations for new users or new items) provided by KG under the meta-learning settings.

To address the above two challenges, we propose a meta-learning framework on KG for cold-start recommendation, called MetaKG, which simultaneously learns the general knowledge from collaborative relations and entities. Specifically, we first define the *collaborative knowledge graph* (CKG) [45], a hybrid structure of user-item interaction graph and knowledge graph, to represent the user behaviors and item knowledge in a unified way. Next, we design two novel meta learners to encode users' potential interests and entities' semantics from CKG, respectively. The *collaborative-aware* meta learner is developed to encode collaborative signals by exploiting gradients of aggregation layer within each user preference learning task, which can automatically generalize high-order connectivities between users and items. The *knowledge-aware* meta learner is further designed to capture the general knowledge semantics of KG by leveraging gradients of embedding and attention layers across tasks. In other words, the *collaborative-aware* meta learner learns the user preference of each learning task to handle the first challenge, while the *knowledge-aware* meta learner shares general semantic knowledge over different learning tasks to tackle the second challenge. Guided with two meta learners, MetaKG can achieve good generalization across tasks, and enables fast adaptation to cold-start scenarios with superior effectiveness and efficiency. To sum up, this paper makes the following key contributions:

- *Meta-learning on KG.* We exploits meta-learning on KG-based recommendations, i.e., MetaKG, which achieves good generalization to alleviate the cold-start problems. To the best of our knowledge, this is the first attempt to investigate meta-learning on KG-based recommendations.
- *Novel meta learners.* We design two meta learners, including *collaborative-aware* meta learner and *knowledge-aware* meta learner, to capture user preferences and knowledge associations. Thus, MetaKG can effectively capture the high-order collaborative relations and semantic representations with limited user-item interactions in cold-start scenarios.
- *Adaptive task scheduler.* To provide informative tasks for meta learners and to prevent the model from being corrupted by noisy tasks, we further propose an adaptive task scheduler for the meta learner training process that decides which task to use later by predicting the probability being sampled for each task, and the scheduler is jointly optimized with two meta learners for better generalization.
- *Extensive experiments.* We conduct extensive experiments on three public benchmarks while considering different cold-start scenarios. The results confirm that our proposed MetaKG substantially outperforms all the state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 reviews the related work, and Section 3 presents the background. Section 4 gives problem definitions. Section 5 details our framework and methods. Section 6 conducts theoretical analysis on computational complexity. The experimental results are reported in Section 7. Finally, Section 8 concludes the paper, and offers some research directions.

2 RELATED WORK

In this section, we review related work on KG-based recommendation and meta-learning, respectively.

2.1 Knowledge Graph Based Recommendation

Unlike the traditional methods that only consider the short connectivity user-item interactions for *user preference recommendation*, more and more researchers start exploring knowledge graph (KG) based recommendation by combining user-item interactions with the side information to improve the recommendation performance. Here, we only highlight KG-based recommendation studies, while

the comprehensive literature about traditional methods can refer to [55].

Existing KG-based recommendation approaches can be mainly divided into three categories, i.e., embedding-based, path-based, and propagation-based methods. *The embedding-based methods* directly embed entities and relations in KG via knowledge graph embedding (KGE) method [44], [58]. For example, CKE [54] utilizes TransR [26] to learn item embeddings of knowledge graph. UGRec [57] devises two different embedding spaces to model direct relations from KG and undirect co-occurrence relations among items. However, the embedding-based methods simply adapt the KGE method into recommenders while ignoring the user-item interactions for recommendations. Thus, they cannot capture the complex dependencies of user-item relations for user preference learning. *The path-based methods* aim to find different semantic paths in KG, and then connect items and users to guide recommendation process. For instance, MCRec [17] develops deep neural networks with the co-attention mechanism to adopt rich meta-path-based context for top-k recommendation. Similarly, KPRN [47] captures the sequential dependence within a path to generate path representations. Nevertheless, the path-based approaches rely on handcraft meta-path construction, which is time-consuming and inevitably leads to information loss. *The propagation-based methods* have attracted increasing attention in recent years, where the propagation process is performed iteratively to extract auxiliary information from KG for user preference learning. As an example, KGAT [45] creates collaborative knowledge graph (CKG) to consider the user-item interactions and the side information of KG to enrich representations of users and items. CKAN [49] employs the heterogeneous propagation strategy to encode the user preference information with knowledge associations to further improve the recommendation performance. More recently, MKGAT [36] integrates KG with additional image and text data to provide multi-modal knowledge for recommendation in an end-to-end fashion. Although existing KG-based methods achieve better performance than traditional methods, they still incur the cold-start problem that widely exists in real-world recommendation systems. For new users who have not yet interacted with enough items or new items that have not been visited before, existing KG-based methods fail to provide high quality recommendation results, since they heavily rely on abundant data to get desired performance, and inevitably lead to overfitting in cold-start scenarios. In light of this, for the first time, we apply a novel meta-learning approach to cold-start recommendation on KG, which can effectively derive prior collaborative signals and knowledge associations within and across different user preference learning tasks to support generic and accurate recommendations.

2.2 Meta-learning

Inspired by the human-learning, meta-learning [35] could learn the prior knowledge across different tasks and then quickly adapt the learned prior knowledge to new tasks. Specifically, meta-learning is often employed to learn a *learner*, which can adapt to new tasks even when there is only a few training samples. Therefore, meta-learning has achieved great success in few-shot learning settings [5], [19], [30], [32]. The existing work on meta-learning can be broadly divided into three categories: metric-based, memory-based, and optimization-based methods.

The metric-based methods aim to learn a metric to evaluate the distance between instances. Siamese Network [22] capitalizes on powerful discriminative features with a convolutional architecture and distance layer. Prototypical Network [19] learns a metric space in which classification tasks can be performed by Euclidean distance computation to prototype representations of each class. However, the metric-based methods are mostly designed for classification problems, which are unsuitable for recommendation analytics involving prediction tasks. *The memory-based methods* exploit memory architectures or specially-designed training processes to store key knowledge that can be generalized to new tasks. For example, MANN [1] introduces a memory-augmented model that can ingratiate significant short-term and long-term memory demands. SNAIL [29] proposes a generic meta learner which combines temporal convolutions and soft attention to aggregate and memorize specific information for new unseen data. Nonetheless, these memory-based methods rely on extra memory mechanisms, resulting in too many parameters to tune. Finally, *the optimization-based methods* learn parameters that are conducive to fast gradient-based adaptation to new tasks, which achieve the state-of-the-art performance. In this category, MAML [5] learns a good initialization from a base model. In that case, the adapted model can achieve strong generalization performance after applying a few gradient steps. In this paper, we utilize the optimization-based meta-learning, as to be discussed in Section 3.2.

Inspired by the success of meta-learning, researchers start devoting efforts to address the cold-start recommendations [9], [24], [38]. MeLU [24] uses the MAML framework [5] for cold-start recommendations, and provides a candidate selection strategy for customized preference estimation. MetaHIN [27] proposes semantic-wise and task-wise adaptation to explore the semantics in heterogeneous networks. Moreover, specific scenarios (such as CTR prediction [28], [31] and clinical risk prediction [56]) are also investigated. However, these methods only consider *one* meta leaner to learn *one* optimization target (i.e., the rating loss), which are not feasible for KG-based recommendation since it is necessary to simultaneously capture the prior collaborative and knowledge information for user preference learning, and the simple model architectures of these methods are unable to capture the high-order relations between users and items.

3 BACKGROUND

Graph neural networks (GNNs) have shown great success in various graph modelings, including directed/undirected graphs, heterogeneous graphs, and knowledge graphs. We design our base model by employing graph attention networks to embed knowledge graphs, as an example of relation knowledge graph is shown in Fig. 1. Then, we utilize the optimization-based meta-learning to support the cold-start recommendations on embedded KG. In this section, we give a brief introduction to the GNNs and meta-learning

TABLE 1
Symbols and Description

Notation	Description
$\mathcal{N}_h$	The set of triplets where h is the head entity
$\mathcal{G}$	The collaborative knowledge graph
$\mathcal{G}_k, \mathcal{G}_u$	The knowledge graph, the user-item bipartite graph
$\mathcal{S}_u$	The support set of user u
$\mathcal{Q}_u$	The query set of user u
$\mathcal{T}_u = (\mathcal{S}_u, \mathcal{Q}_u)$	A rating task of user u
δ	The parameters of adaptive task scheduler
ϕ	The parameters of embedding layer
ω	The parameters of graph gated attention layer
γ	The parameters of aggregation layers
v	Learning rate of local update
k	Learning rate of global update
UC, IC , UIC	user, item, user-item cold-start scenarios

related to the implementation of our presented framework. For ease of understanding, the frequently used notations are summarized in Table 1.

3.1 Graph Neural Networks (GNNs)

Given a graph consisting of nodes and edges, GNNs aim to learn the representation for each node by leveraging the information propagation from its neighborhoods in both spectral and spatial domains. Meanwhile, GNNs perform aggregation operations from those learned embeddings. Specifically, GNNs capture the deep representation for each node n_i by the following equations:

$$\mathbf{e}_{\mathcal{N}_i}^{l+1} = \mathrm{PROP}^l \left(\left\{\!\!\left\{ \left(\mathbf{e}_i^l, \mathbf{r}_{i,j}, \mathbf{e}_j^l \right) : j \in \mathcal{N}_i \right\}\!\!\right\} \right) \quad (1)$$

$$\mathbf{e}_i^{l+1} = \mathrm{AGG}^l \left(\mathbf{e}_i^l, \mathbf{e}_{\mathcal{N}_i}^{l+1} \right) \quad (2)$$

Here, $\mathbf{e}_i^l$ represents the embedding of the l-th layer of node n_i, $\mathbf{r}_{i,j}$ stands for the representation of edge connecting node n_i and node n_j, $\{\!\!\{\cdot\}\!\!\}$ denotes a multiset, and $\mathcal{N}_i$ is a set that represents the neighboring nodes of e_i. In addition, PROP denotes the information propagation operator, and AGG represents the information aggregation operator. When PROP propagates information from the neighboring edges and nodes to the source node, AGG transforms the entity embeddings into better ones at the same time. More details about GNNs can refer to [39].

There are lots of propagation methods to choose, including convolution [21], self-attention [39], and knowledge-aware attention [45]. As for the aggregation methods, we can use GCN aggregator [21], GraphSage aggregator [12], and bi-interaction aggregator [45]. In this paper, we utilize knowledge-aware attention and bi-interaction aggregator for its simplicity and good performance.

3.2 Optimization-based Meta-learning

We employ the optimization-based meta-learning methods, which have been proved to achieve the state-of-the-art performance as discussed in Section 2.2. The optimization-based meta-learning aims at finding desirable parameters θ that are sensitive to new tasks, so that fine-tuning to those parameters could obtain large improvement in the task loss. Formally, given a distribution $p(\mathcal{T})$ that all the tasks follow, the optimization-based meta-learning aims to train a model f which is able to quickly adapt to new task $\mathcal{T}_{new}$ based on the knowledge obtained from $p(\mathcal{T})$. As shown in Fig. 2,
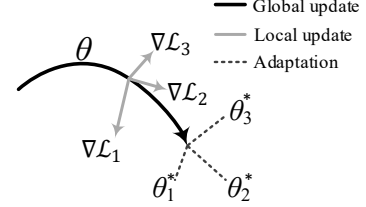

Fig. 2. Diagram of Optimization-based Meta-learning

there are two kinds of update operations during the meta-learning process, namely, local update and global update. Specifically, the model locally updates its parameters θ to $\theta_i^\star$, which is performed though gradient descent updates on task $\mathcal{T}_i$ with a local update/learning rate v:

$$\theta_i^\star = \theta - v\nabla_\theta \mathcal{L}_{\mathcal{T}_i} (f_\theta) \quad (3)$$

Based on the local update for task-specific parameter $\theta_i^\star$, the model further updates its global parameters by optimizing $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i^\star})$ over different tasks, with a global learning rate k. Thus, the global parameters θ can fit into various tasks.

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i} (f_{\theta_i^\star}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i} \left(f_{\theta - v\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)} \right) \quad (4)$$

$$\theta \leftarrow \theta - k\nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i} (f_{\theta_i^\star}) \quad (5)$$

Note that, the optimization-based meta-learning divides each task into two sets for the model training, including the support set and the query set. During the local update, the model's local parameters $\theta_i^\star$ are trained over each support set, while the model searches task-specific desired optimization directions. During the global update, the model's global parameters θ (i.e., the directions in Fig. 2) are altered based on the loss of query set using $\theta_i^\star$.

4 PROBLEM FORMULATION

In this section, we proceed to introduce key concepts related to our studied problem, and present problem formulation.

User-Item Bipartite Graph. In recommendations, the core task is how to represent the user-item relations. We model the user-item interactions with a user-item bipartite graph $\mathcal{G}_u$, defined as follows:

$$\mathcal{G}_u = \{(u, i, l) \,| u \in \mathcal{U}, i \in \mathcal{I}, l_{ui} \in \mathcal{L}\} \quad (6)$$

where $\mathcal{U}$ and $\mathcal{I}$ denote the user set and item set, respectively. $\mathcal{L} = \mathcal{U} \times \mathcal{I}$ defines the user-item edges/interactions. In $\mathcal{G}_u$, if there is an observed edge between user u and item i, $l_{ui} = 1$; otherwise $l_{ui} = 0$.

Knowledge Graph. To capture the high-order relations between uses and items discussed in Fig. 1, we introduce knowledge graph (KG) to provide the side information. A KG is a directed graph $\mathcal{G}_k$ as defined below:

$$\mathcal{G}_k = \{(h, r, t) \,| h \in \mathcal{E}, t \in \mathcal{E}, r \in \mathcal{R}\} \quad (7)$$

Here, each triple (h, r, t) in $\mathcal{G}_k$ means that there is a relation r between the head entity h and the tail entity t. $\mathcal{E}$ and $\mathcal{R}$ represent the entity set and the relation set in $\mathcal{G}_k$, respectively. For example, a triple (*Mark Hamill*, *ActorOf*, *Star War*) indicates that *Mark Hamill* is an actor of the movie *Star War*. We assume the relations in $\mathcal{R}$ are bidirectional, so that they

contain both canonical direction (i.e. *ActorOf*) and inverse direction (i.e. *ActedBy*).

Based on the defined user-item bipartite graph G_u and knowledge graph G_k, we can further define a set $\mathcal{A} = \{(i,e) \,|\, i \in \mathcal{I}, e \in \mathcal{E}\}$, where $\mathcal{I}$ belongs to $\mathcal{G}_u$, $\mathcal{E}$ belongs to $\mathcal{G}_k$, and (i,e) means that the item i from $\mathcal{G}_u$ can be aligned with the entity e (i.e., h or r) from $\mathcal{G}_k$.

Collaborative Knowledge Graph. Similar to the previous studies on KG-based recommendations [45], we define the collaborative knowledge graph (CKG) $\mathcal{G}$ as a relational graph that combines user-item bipartite graph $\mathcal{G}_u$ and knowledge graph $\mathcal{G}_k$ in a unified manner. Specifically, we integrate $\mathcal{G}_u$ with $\mathcal{G}_k$ based on the item-entity alignment set $\mathcal{A}$. Overall, the final CKG $\mathcal{G}$ is defined as:

$$\mathcal{G} = \{(h,r,t) \,|\, h \in \mathcal{E}', t \in \mathcal{E}', r \in \mathcal{R}'\} \tag{8}$$

Here, $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$, indicating that we treat users as entities as well. Besides, $\mathcal{R}' = \mathcal{R} \cup \mathcal{L}$, meaning that we treat user-item interactions as relations as well.

Problem Statement. Given a collaborative knowledge graph $\mathcal{G}$ that combines the user-item bipartite graph $\mathcal{G}_u$ and the knowledge graph $\mathcal{G}_k$, we aim to predict the unknown probability (i.e., user preference) p_{ui} from user u to item i, where $r_{u,i} \notin \mathcal{R}'$. Specifically, if u is a new user with only a small number of interactions, i.e., $|r_{u',i} \in \mathcal{R}' : u' = u|$ is small, it is known as <u>u</u>ser <u>c</u>old-start problem (UC). Similarly, if i is a new item with limited interactions from existing users, i.e., $|r_{u,i'} \in \mathcal{R}' : i' = i|$ is small, it is known as <u>i</u>tem <u>c</u>old-start problem (IC). Last but not the least, if both u and i are new, it is known as <u>u</u>ser-<u>i</u>tem <u>c</u>old-start problem (UIC).

In this paper, our objective is to alleviate the cold-start problems involving UC, IC, and UIC perspectives using meta-learning on KG-based recommendations.

5 METHODOLOGY

In this section, we first give an overview of the MetaKG framework, and then, we present the detailed methods.

5.1 Overiew of MetaKG

We propose the MetaKG framework in both architecture and learning perspectives to better illustrate each module's functionality as well as the meta-learning process.

In terms of the architecture perspective, MetaKG is established on three essential components: adaptive task scheduler, *collaborative-aware* meta learner and *knowledge-aware* meta learner, as depicted in Fig. 3. The adaptive task scheduler (to be discussed in Section 5.4) selects the most informative tasks from the task pool to filter out noisy tasks for meta learners training. The collaborative-aware meta learner (to be discussed in Section 5.5) aggregates hidden information with user preference in the aggregation layer in order to capture the high-order relations. It is worth mentioning that the collaborative-aware meta learner is able to automatically learn high-order collaborative signals, without relying on the predefined meta-paths or domain knowledge [27]. In addition, we further design the knowledge-aware meta learner (to be discussed in Section 5.6) to capture the general knowledge semantics shared by tasks. Equipped with two meta learners, MetaKG enables to explicitly encode the collaborative signals and knowledge associations in a meta-learning manner, and achieves fast adaptation for new users and new items.

In terms of the learning perspective, MetaKG consists of three procedures: local update (the orange lines at the top in Fig. 3), global update (the black lines at the bottom in Fig. 3), and adaptation process. During the local update, the collaborative-aware meta learner is utilized to learn the personalized preference for each task/user. During the global update, the knowledge-aware meta learner is further employed to learn general meta knowledge associations of KG shared across different tasks/users. Once our model has been well-trained, it can easily adapt to the new scenarios with only a few gradient descent steps, and achieve high quality predictions, as indicated in our experiments.

5.2 Meta-learning Setups

Since MAML [5] is originally designed as a model-agnostic meta-learning framework, it is necessary to clarify more setups when adapting it to the cold-start recommendations. We define a task for a user u as $\mathcal{T}_u = (\mathcal{S}_u, \mathcal{Q}_u)$, indicating that one task is bound to one user. Here, $\mathcal{S}_u$ denotes the support set and $\mathcal{Q}_u$ represents the query set, as discussed in Section 3.2. During the training/learning process, $\mathcal{Q}_u$ and $\mathcal{S}_u$ are sampled from the item set which a user u has interacted with. Note that the support set and query set are mutually exclusive. During the adaptation process, each task $\mathcal{T}_u \in \mathcal{T}_{test}$ also has a support set and a query set, and items in the query set are to be predicted. As defined in Section 4, **UC** denotes the task that new users are not seen in training process, **IC** stands for the task which new items are not seen in training process, and **UIC** represents the task that both users and items are not seen in training process.

5.3 Base Model

As shown in Fig. 3, we design the base model f_θ that contains four layers: i) embedding layer h_ϕ, which parameterizes the entities and relations as the vector representations; ii) graph gated attention layer g_ω, which utilizes relation-aware gated attention mechanism to neighborhood nodes; iii) aggregation layer y_γ, which aggregates entity representation with its ego-network information; and iv) prediction layer, which predicts a matching score based on the user and item representations. Overall, the proposed base model can be represented $f_\theta = (h_\phi, g_\omega, y_\gamma)$, as described below.

(i) Embedding layer. Both entity embeddings and relation embeddings are important to preserve the graph structure of knowledge graph. Hence, we first utilize TransR [26] on CKG to get the initial embeddings. Then, we employ a learnable projection layer to enrich their representations:

$$e_i = h_\phi(c_i) = W_e c_i + b \tag{9}$$

where c_i is a d_p-dimensional one-hot vector denoting entity or relation of CKG, and $W_e \in \mathbb{R}^{d_e \times d_p}$ and $b \in \mathbb{R}^{d_e}$ are trainable parameters, i.e., $\phi = \{W_e, b\}$.

(ii) Graph gated attention layer. Previous methods [45], [49] typically aggregate all neighbor triplets to generate the contextual representations, without differentiating the semantics that neighbor entities carry with. For example,
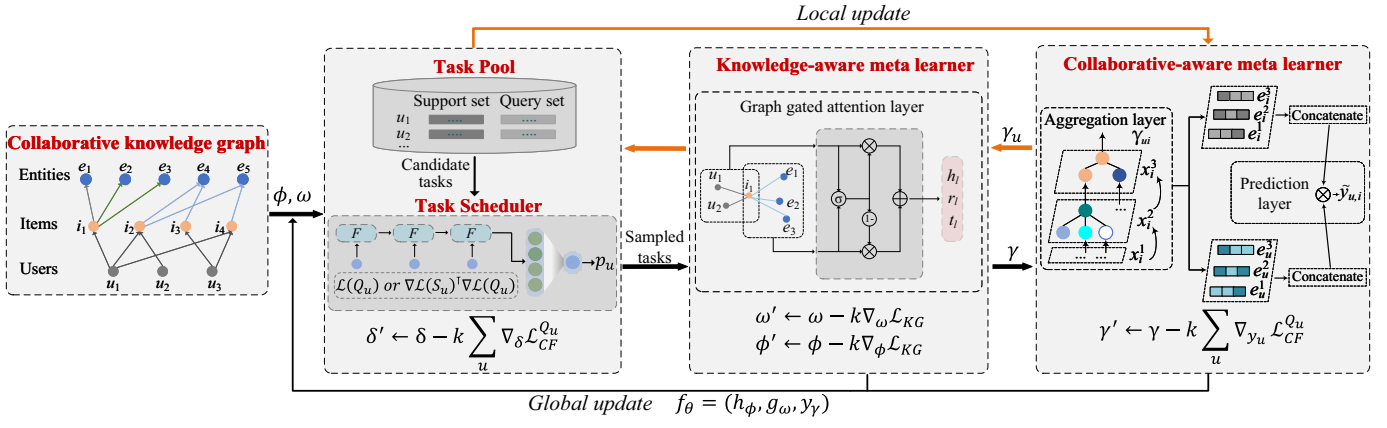
Fig. 3. An Overview of MetaKG Framework

for item h, the neighbor entity t and neighbor user u represent the knowledge relevance and collaborative connectivities of h, respectively. However, directly aggregating them would fail to generate discriminative signals towards user's preference representations, thus leading to suboptimal performance. Therefore, we aim to design a gated attention module for explicitly aggregating two kinds of information. Specifically, we use $\mathcal{N}_h^c = \{(h, r, u) | (h, r, u) \in \mathcal{G}, u \in \mathcal{U}\}$ to represent the set of triplets where h is the head entity and the tail entity u is an user, and summarize the weighted representation to capture the first-order information:

$$\overrightarrow{e_h^c} = g_\omega(h, r, u) = \sum_{(h,r,u)\in\mathcal{N}_h^c} \alpha(h, r, u)e_u \quad (10)$$

where $\alpha(h, r, u)$ is the attention weight, indicating the importance of tail u and relation r to head h. We use relation-aware attention mechanism to calculate such importance:

$$\alpha'(h, r, u) = (W_r e_u)^\top \tanh((W_r e_h + e_r) \quad (11)$$
$$\alpha(h, r, u) = \text{softmax}(\alpha'(h, r, u)) \quad (12)$$

here, $W_r \in \mathbb{R}^{d_r \times d_e}$ projects entities from the d_e-dimension entity space into the d_r-dimension relation space. We use the same attention strategy to generate the $\overrightarrow{e_h^k}$ for entity h with knowledge neighbor triplets $\mathcal{N}_h^k = \{(h, r, t) | (h, r, t) \in \mathcal{G}, t \in \mathcal{E}\}$. Inspired by the design of GRU [7] that learns gating signals to control the update of hidden states, we propose to learn a fusion gate which can adaptively control the combination of two semantic representations:

$$g_h = \sigma(W_c \overrightarrow{e_h^c} + W_k \overrightarrow{e_h^k}) \quad (13)$$
$$\overrightarrow{e_h} = g_h \cdot \overrightarrow{e_h^c} + (1 - g_h)\overrightarrow{e_h^k} \quad (14)$$

where $W_c, W_k \in \mathbb{R}^{d_e \times d_e}$ are learnable transformation parameters and σ is the sigmoid function. The $g_h \in \mathbb{R}^e$ denotes the learned gate signals to balance the contributions of collaborative signals and knowledge associations, and the gated attention layer is parameterized by $\omega = \{W_r, W_c, W_k\}$.

(iii) Aggregation layer. After finishing information propagation, a bi-interaction aggregator is used to model two kinds of feature interaction between e_h and $\overrightarrow{e_h}$ as follows:

$$e_h^{l+1} = y_\gamma^l(e_h^l, \overrightarrow{e_h}^l) = \text{LeakyReLU}\left(W_1^l(e_h^l + \overrightarrow{e_h}^l)\right) + \text{LeakyReLU}\left(W_2^l(e_h^l \odot \overrightarrow{e_h}^l)\right) \quad (15)$$

where $W_1, W_2 \in \mathbb{R}^{d_l \times d_e}$ are learnable matrices, $\odot$ denotes the element-wise product, and l is the l-layer propagation. Therefore, y_γ is the aggregative function parameterized by $\gamma = \{(W_1^l, W_2^l) | \forall l \in \{1, 2, ..., L\}\}$.

(iv) Prediction layer. In terms of preference prediction, given user's multi-layer representation $\{e_u^1, ..., e_u^L\}$ and item's multi-layer representation $\{e_i^1, ..., e_i^L\}$, we concatenate each layer representation into a single vector:

$$e_u^\star = e_u^1 \| \cdots \| e_u^L, \quad e_i^\star = e_i^1 \| \cdots \| e_i^L \quad (16)$$

Then, we predict their preference score by inner product of user and item representations, and opt for the BPR loss [33] to learn the preference of users below:

$$\mathcal{L} = \sum_{(u,i,j)\in\mathcal{O}} -\ln \sigma(e_u^{\star\top} e_i^\star - e_u^{\star\top} e_j^\star) \quad (17)$$

where $\mathcal{O} = \{(u, i, j) | (u, i) \in \mathcal{O}^+, (u, j) \in \mathcal{O}^-\}$ is the training dataset consisting of the observed interactions $\mathcal{O}^+$ and unobserved counterparts $\mathcal{O}^-$; and $\sigma(\cdot)$ is the sigmoid function. It is worth mentioning that the base model is supervised, and needs a lot of data to get reasonable performance, which is not feasible for cold-start scenarios. Thus, we regard the cold-start recommendation as a meta-learning problem. Different from previous meta-learning approaches [24], [27] that consider *one* meta learner to learn *one* optimization target (*i.e.*, the rating loss), MetaKG includes two novel meta learners with different optimization targets to encode the prior information from both user-item interactions and KG. Next, we detail our proposed adaptive task scheduler and meta learners as well as the co-adaptation process.

5.4 Adaptive Task Scheduler

Previous meta-learning based recommendations [24], [27] randomly sample meta-training tasks with a uniform probability, under the assumption that tasks are of equal importance. However, given rather limited number of meta-training tasks in cold-start scenarios, it is likely that some tasks are noisy [18] or contribute differently for user preference learning. To prevent the meta learners from being corrupted by such detrimental tasks or dominated by tasks in the majority, we propose an adaptive task scheduler to select the most informative tasks for training. Specifically, we define the scheduler as z with parameters δ, and utilize two

representative factors to quantify the information covered in $\mathcal{T}_u$:

$$p_u = z_\delta\big(\mathcal{L}(\mathcal{Q}_u), \nabla\mathcal{L}(\mathcal{S}_u)^\top \nabla\mathcal{L}(\mathcal{Q}_u)\big) \tag{18}$$

where p_u is the sampling probability of candidate task $\mathcal{T}_u$, $\mathcal{L}(\mathcal{Q}_u)$ denotes the loss on the query set, and the second part is the gradient similarity between the support and query set of task $\mathcal{T}_u$. These factors are associated with the learning outcome and the learning process of the task $\mathcal{T}_u$, and a large query set loss may represent at true hard task if the gradient similarity is large; a task with noises in its query set could lead to large query loss but small gradient similarity. Thus, we follow [51] to choose LSTM network [16] as the scheduler z to encode the historical information of each task and to capture the prediction variance [4]. The parameters δ of the scheduler are simultaneously optimized with the model, as demonstrated in Algorithm 1.

5.5 Collaborative-aware Meta Learner

The collaborative-aware meta learner aims to learn multi-faceted preference from CKG. Here, we also use the BPR loss [33] to learn the preference of each user:

$$\mathcal{L}_{\mathrm{CF}}^{\mathcal{S}_u} = \sum_{(u,i,j)\in\mathcal{S}_u} -\ln\sigma(e_u^{\star\top}e_i^\star - e_u^{\star\top}e_j^\star) \tag{19}$$

where $\mathcal{S}_u = \{(u,i,j)\,|\,(u,i)\in\mathcal{O}^+, (u,j)\in\mathcal{O}^-\}$ is the support set for task u. Given a task $\mathcal{T}_u$, the user's embedding e_u, and the weighted propagation representation $\overrightarrow{e_u}$ at l-layer, we encode feature interaction at each layer, and concatenate into one final representation as follows:

$$e_u^\star = \|_{l\in L}\, y_\gamma^l(e_u^l, \overrightarrow{e_u}^l) \tag{20}$$

In addition, the collaborative signals are automatically captured by collaborative-aware meta learner through learnable interaction aggregator γ of task $\mathcal{T}_u$:

$$\gamma_u = \gamma - v\frac{\partial\mathcal{L}_{\mathrm{CF}}^{\mathcal{S}_u}(\phi,\omega,e_u^\star)}{\partial\gamma} \tag{21}$$

where v is the learning rate of local update, and γ_u is specific information aggregator of task $\mathcal{T}_u$. Therefore, collaborative-aware meta learner locally updates the aggregation layer by optimizing $\mathcal{L}_{\mathrm{CF}}$ on the support set of each user task in order to capture the user-specific preference.

5.6 Knowledge-aware Meta Learner

The knowledge-aware meta leaner models the entity representation on the granularity of triplets. We first introduce the energy score for a given triplet (h,r,t):

$$s(h,r,t) = \|W_r e_h + e_r - W_r e_t\|_2^2 \tag{22}$$

where e_h, e_r, and e_t are the embeddings of the head entity, the relation, and the tail entity, respectively. This means we first project the entities into the relation space with the help of W_r (i.e., $W_r e_h$ and $W_r e_t$), and then model the additive relationship of knowledge triplet [26]. The smaller the energy score $s(h,r,t)$ is, the more likely the triplet $s(h,r,t)$ is true. Based on this, a pairwise ranking loss is

---

Algorithm 1: Meta Learning on knowledge graph for cold-start recommendation

Input: $\mathcal{G}, \mathcal{T}, \theta, \omega, \phi, \gamma, \delta, c, k$
Output: the learned parameters θ' in a new scenario

1 Initialize entity embedding with TransR on $\mathcal{G}$
2 **while** *not converge* **do**
3   Calculate the sampling probability p of each task in $\mathcal{T}_{train}$ by Eq. 18
4   Sample batch of task $(\mathcal{S}, \mathcal{Q}) \in \mathcal{T}_{train}$ according to the sampling probabilities
5   **for** $\mathcal{S}_u \in \mathcal{S}, \mathcal{Q}_u \in \mathcal{Q}$ **do**
6   Calculate user representation $e_u^\star$ through θ
7   Local update $\gamma_u \leftarrow \gamma - v\nabla_\gamma\mathcal{L}_{\mathrm{CF}}^{\mathcal{S}_u}$
        Global update $\omega' \leftarrow \omega - k\nabla_\omega\mathcal{L}_{\mathrm{KG}}$
8         $\phi' \leftarrow \phi - k\nabla_\phi\mathcal{L}_{\mathrm{KG}}$
        $\gamma' \leftarrow \gamma - k\sum_u \nabla_{\gamma_u}\mathcal{L}_{\mathrm{CF}}^{\mathcal{Q}_u}$
9   Update the scheduler $\delta' \leftarrow \delta - k\sum_u \nabla_\delta\mathcal{L}_{\mathrm{CF}}^{\mathcal{Q}_u}$
10 Co-adapt to the new scenario according to Eq. 26
11 **return** θ'

---

used to encourage the direct connections through negative sampling:

$$\mathcal{L}_{\mathrm{KG}} = \sum_{(h,r,t,t')\in\mathcal{S}} -\ln\sigma\left(s\left(h,r,t'\right) - s(h,r,t)\right) \tag{23}$$

Here, $\mathcal{S} = \{((h,r,t,t')\,|\,(h,r,t)\in\mathcal{G}, (h,r,t')\notin\mathcal{G}\}$, and $\sigma(\cdot)$ is the sigmoid function.

Recall that the entity representations are parameterized by ϕ and ω. The semantic information of entities should not be adapted with a specific task, but should be shared across the whole knowledge graph. Motivated by this, we design a knowledge-aware meta learner to learn knowledge associations and semantic relatedness over tasks:

$$\phi' = \phi - k\frac{\partial\mathcal{L}_{\mathrm{KG}}(\gamma,c)}{\partial\phi} \tag{24}$$

$$\omega' = \omega - k\frac{\partial\mathcal{L}_{\mathrm{KG}}(\gamma,c)}{\partial\omega} \tag{25}$$

where k is the learning rate of global update and c is the input one-hot vector of entities. In other words, knowledge-aware meta learner globally updates the embedding layer and attention layer by optimizing $\mathcal{L}_{\mathrm{KG}}$ across all the tasks.

5.7 Co-adaptation

Our MetaKG learns the model parameters θ based on the user preference and the knowledge associations by optimizing two meta learners alternatively. During the training process, the collaborative-aware meta learner locally updates aggregation layer γ through backpropagation of prediction loss according to Eq. 21, while the knowledge-aware meta learner globally optimizes the embedding layer ϕ and attention layer ω through backpropagation of KG loss according to Eq. 24 and 25. Note that, we first train our model to get the meta parameters θ', and then, we adapt θ' to new scenarios (e.g., user cold-start) as below:

$$\min_{\theta'} \sum_{\mathcal{T}_u\in\mathcal{T}_{new}} \mathcal{L}_{\mathrm{CF}}^{\mathcal{S}_u}(\phi',\omega',e_u^\star) + \mathcal{L}_{\mathrm{KG}}(\gamma',c) \tag{26}$$

where $\mathcal{T}_{new}$ denotes the new scenarios; while ϕ', ω', and γ' are learned parameters from two meta learners during local and global updates. During the adaptation process, MetaKG further updates the learned parameters using the support set of a new scenario. The detailed training process of MetaKG is presented in Algorithm 1.

6 MODEL ANALYSIS

In this section, we present the theoretical analysis on the computational complexity and the model parameters.

Computational complexity. The time complexity of MetaKG for recommendation involves three aspects. For the adaptive task scheduler, the task probability estimation has the computational complexity of $\mathcal{O}(|\mathcal{T}|d_{lstm}^2)$, where $|\mathcal{T}|$ is the number of tasks, and d_{lstm} is the hidden size of LSTM. For the collaborative-aware meta-learner, the matrix multiplication of the l-th layer has the computational complexity of $\mathcal{O}(|\mathcal{G}|d_l d_{l-1})$, where $|\mathcal{G}|$ is the number of triplets in CKG, and d_l and d_{l-1} denote the current and previous transformation size. For the knowledge-aware meta learner, the knowledge graph embedding has the computational complexity of $\mathcal{O}(|\mathcal{G}_k|d_p d_e)$, where d_p represents the category dimension, and d_e denotes the embedding size. For the prediction layer, the time complexity of inner product is $\mathcal{O}(\sum_{l=1}^{L}|\mathcal{G}_u|d_l)$, in which $|\mathcal{G}_u|$ is the number of items in the user-item bipartite graph. Assume that we have m local updates per global update, the overall training complexity of MetaKG is $\mathcal{O}(m|\mathcal{G}_k|d_l d_{l-1} + |\mathcal{T}|d_{lstm}^2 + |\mathcal{G}|d_p d_e + \sum_{l=1}^{L}|\mathcal{G}_u|d_l)$.

It is worth noting that traditional methods (i.e., KGAT) need to update all the parameters every epoch, which suffers from the slow convergence for its tremendous parameters. In contrast, MetaKG locally updates the aggregation layer only m times (m is usually small as validated in Section 7.6), and globally updates only a few steps, which yields high performance. Hence, the overall training is efficient (to be reported in Section 7.4).

Parameters. The model size largely depends on the knowledge graph embedding layer (e.g., nearly 11 million on Yelp2018 dataset), which is the same as the state-of-the-art method NFM [14]. In contrast, the weights of propagation process are much smaller (e.g., 15 thousand of three propagation layer on Yelp2018 dataset). It is worth mentioning that our two key components (i.e., collaborative-aware and knowledge-aware meta learners) do not need more parameters to achieve good generalization, thanks to the meta-learning framework.

Overall, MetaKG is a lightweight recommendation framework with high efficiency and adaptability, which is also demonstrated by empirically comparing MetaKG with various state-of-the-art methods in Section 7.4.

7 EXPERIMENTS

In this section, we first present experimental settings, and then provide extensive experiments to evaluate our proposed MetaKG in terms of the recommendation performance, the model ablation, the model efficiency, the model scalability, and the parameter analysis, respectively.

TABLE 2
Statistics of the Datasets Used in Our Experiments

		Yelp2018	Last-FM	Amazon-book
$\mathcal{G}_u$	#Users	45, 919	23, 566	70, 679
	#Items	45, 538	48, 123	24, 915
	#Interactions	1, 185, 068	3, 034, 796	847, 733
$\mathcal{G}_k$	#Entities	90, 961	58, 266	88, 572
	#Relations	42	9	39
	#Triplets	1, 853, 704	464, 567	2, 557, 746

7.1 Experimental Settings

Datasets. To evaluate the performance of MetaKG, we use three public and popular benchmark datasets involving different scenarios: Amazon-book, Last-FM, and Yelp2018. These datasets are also adopted in the state-of-the-art methods [45]. For each dataset, k-core setting is used to ensure the quality of the dataset, meaning that we only keep users and items with at least k interactions. Here, we use 10-score setting for all the three datasets. The detailed dataset descriptions are as follows.

- **Yelp2018.**[1] This is a local business rating dataset collected by Yelp. We use the 2018 edition dataset of the Yelp challenge, and we take the local businesses like restaurants and bars as the items.
- **Last-FM.**[2] This is a music listening dataset provided by Last.fm online music system, where tracks are identified as items. We follow the work [45], [48] to take the subset of the dataset where the timestamp is from Jan, 2015 to June, 2015.
- **Amazon-book.**[3] We select Amazon-book from Amazon-review, a widely used dataset for product recommendation [13]. Here, if a user u rates an item i, we set it as an observed interaction, which means $l_{ui} = 1$ in the user-item bipartite graph $\mathcal{G}_u$.

In addition, we derive the knowledge graphs from Freebase (for Amazon-book and Last-FM datasets), and the local business information network (for Yelp2018 datasets), which are widely used in preivous studies [45], [49]. The overall statistics of datasets are summarized in Table 2.

Cold-start Scenarios. We follow the previous work [27] to construct the cold-start scenarios. Specifically, we first divide users and items into two groups (i.e., old ones and new ones) according to user's joining time (or user's first action time) and item's releasing time. Then, we split each dataset into training and testing data sets. For training data set, we only collect old user ratings for old items. For testing data, users and items are partitioned into three scenarios: User Cold-start (UC), (i.e. recommending old items for new users); Item Cold-start (IC), (i.e., recommending new items for old users); and User-Item Cold-start (UIC), (i.e., recommending new items for new users).

Meta-learning Settings. We collect the support and query sets following the previous work [24]. Among the items rated by user u, we randomly select 10 items as the query set (*i.e.* $|\mathcal{Q}_u| = 10$), and use the remaining items as the support set. We will also show the impact of the support set size on the model performance in Section 7.6.

Evaluation Metrics. We treat all the items that user u has not interacted with as the negative items. Then, each

method outputs the user's preference scores over all the items, except for the rated ones in the training set. We adopt two widely-used evaluation protocols [45], [47], [48], i.e., Recall@K and normalized discounted cumulative gain at rank K (NDCG@K). Here, we set K to 20.

Compared Methods. We comprehensively compare our proposed MetaKG with traditional methods (FM and NFM), the embedding-based methods (CKE, CFKG and UGRec), the path-based methods (MCRec and RippleNet), the propagation-based methods (KGCN, KGAT and CKAN), and the state-of-the-art meta-learning method (MeLU and MetaHIN). Note that, we omit some methods (e.g., PER [53] and DKN [41]), because they have been obviously defeated by recently proposed methods: KGCN [43] and KGAT [45].

- **FM** [34] is a factorization benchmark, which considers the second-order interactions between users and items.
- **NFM** [14] is the state-of-the-art factorization method which uses MLP to learn nonlinear and high-order interaction signals.
- **CKE** [54] is an embedding-based method, which applies TransR [26] and CF to combine structural, visual, and textual knowledge in a unified framework.
- **CFKG** [2] applies TransE [3] on the unified graph including users, items, entities, and relations, casting the recommendation task as the prediction of (u, r, v) triplets.
- **UGRec** [57] is the state-of-the-art embedding-based method which not only models directed knowledge-aware relations with TransD [20], but also utilizes attentive mechanism to model undirected co-occurrence relations among items.
- **MCRec** [17] is a path-based model, which extracts qualified meta-paths as connectivities between users and items.
- **RipppleNet** [40] is the state-of-the-art path-based models, which uses a memory-like network, and propagates users' potential preferences to enrich user representations.
- **KGCN** [43] is one of the state-of-the-art propagation-based models, which extends GCN to the KG by selectively aggregating neighborhood information, to learn both structure and semantic information of the KG as well as users' personalized interests.
- **KGAT** [45] is also one of the stat-of-the-art propagation-based models, which utilizes an attention mechanism to discriminate the importance of neighbors in CKG.
- **CKAN** [49] is based on KGAT, which utilizes different aggregation schemes on the user-item graph and KG, to encode knowledge association and collaborative signals.
- **MeLU** [24] is a meta-learning method that only leverages interactions to learn embeddings of new users and items.
- **MetaHIN** [27] is the stat-of-the-art meta-learning method that captures multifaceted semantics on heterogeneous information networks to address cold-start problems.

Parameters Settings. We apply grid search for hyper-parameter tuning. We adopt Adam to optimize MetaKG, and utilize the default Xavier initializer [11] to initialize the parameters. The embedding size is fixed to 64 for users, entities and relations. For all datasets, we set the global learning rate to 0.0001, local learning rate to 0.01 and scheduler learning rate to 0.001 after tuning amongst {0.1, 0.01, ..., 0.0001}. For adaptive scheduler module, the bidirectional LSTM with hidden units set to 10 is used for embedding both the

loss and the gradient similarity. To explore the high-order connectivity information, we stack three propagation layers with three fusion gates. Finally, we implement MetaKG in Python and Pytorch[4]. All experiments were conducted on a server with Intel(R) Xeon(R) Silver 4216, 2.10GHz CPU, 256GB RAM, and a NVIDIA A100 GPU. The used datasets and source codes of MetaKG are available online[5].

For baselines, we set their parameters under the guidance of literature. The user, item, entity and relation embedding dimensions are fixed to 64 for all baselines. The learning rate is tuned amongst {0.0001, 0.001, 0.01}, and the coefficient of L_2 normalization is searched in {10^{-6}, 10^{-5}, ..., 10^{-1}}. For NFM, we set the MLP layers size to [64, 32, 1] with dropout ratio 0.1. For CKE, CFKG, and KGAT, the KG training batch size is set to 2048, and the learning rate is 0.0001. Specifically, for KGAT, we set the depth to three with the hidden size [64, 32, 16], and set dropout ratio to 0.1. For RippleNet, we set the number of hops and the memory size to 2 and 8, respectively. For UGRec, the margin value for hinge loss is set to 1.5. For CKAN, we set the size of triple set for user and item to 16 and 64 respectively. For MeLU and MetaHIN, both of the local and global learning rate are set to 0.0001, and a single local update is applied to this model.

7.2 Performance Comparison

In this section, we investigate the performance of MetaKG compared with state-of-the-art methods under three cold-start scenarios and one non-cold-start scenario, using three public datasets. Table 3 summarizes the results.

Cold-start Scenarios. As observed in Table 3, the results of three cold-start scenarios (i.e., UC, IC, and UIC) show that our MetaKG consistently yields the best performance among all the methods on all the datasets. Specifically, MetaKG achieves 62.14%, 74.95%, and 67.28% of average improvement over the competitors in Yelp2018, Last-FM, and Amazon-book, respectively. The second observation is that embedding-based methods (i.e., CKE, CFKG and UGRec) perform better than traditional methods (i.e., FM and NFM) in most cases, indicating that the effectiveness of entity embeddings for recommendation. The third observation is that the path-based methods (i.e., MCRec and RippleNet) exhibit poor competitiveness compared with the embedding-based methods. This is because the handcraft meta-paths highly depend on the massive user-item interactions and thus are insufficient to model implicit relations in cold-start scenarios. Moreover, the propagation-based methods (i.e., KGCN, KGAT and CKAN) perform well among all the baselines since they enable to capture collaborative signal in an end-to-end manner, while the meta-learning based methods (i.e., MetaHIN and our MetaKG) achieve better performance than the propagation-based methods. This further demonstrates the effectiveness of meta-learning for cold-start recommendation problems. However, MetaHIN still underperforms our MetaKG in all scenarios. The reason is that MetaHIN only considers preference learning with handcraft meta-paths, while MetaKG automatically explores the high-order connectivity within each task, and captures entities'

---

4. https://pytorch.org
5. https://github.com/ZJU-DBL/MetaKG

TABLE 3
Performance Comparison of Four Recommendation Scenarios on Three Datasets ("RI" indicates the relative improvement of our MetaKG over the corresponding baseline)

Scenario	Methods	Yelp2018			Last-FM			Amazon-book		
		Recall@20	NDCG@20	RI	Recall@20	NDCG@20	RI	Recall@20	NDCG@20	RI
User Cold-start (UC) Old items for new users	FM	0.0978	0.0812	45.74%	0.1917	0.1659	110.86%	0.1597	0.1420	116.17%
	NFM	0.1174	0.0998	19.93%	0.2143	0.1905	86.06%	0.1791	0.1573	94.00%
	CKE	0.1188	0.1050	16.18%	0.2632	0.2521	45.96%	0.2401	0.2391	35.72%
	CFKG	0.0936	0.0811	48.95%	0.2525	0.2362	53.92%	0.2180	0.2050	53.84%
	UGRec	0.0711	0.0624	94.81%	0.1912	0.1956	94.66%	0.0778	0.0717	335.56%
	MCRec	0.0920	0.0793	51.95%	0.1580	0.1532	141.70%	0.1529	0.1641	105.51%
	RippleNet	0.1001	0.0833	42.22%	0.1760	0.1607	123.51%	0.1815	0.1741	82.93%
	KGCN	0.1123	0.0901	29.23%	0.2206	0.2067	76.04%	0.2445	0.2279	37.79%
	KGAT	0.1240	0.1110	10.59%	0.2821	0.2744	35.16%	0.2804	0.2712	17.92%
	CKAN	0.0844	0.0802	57.74%	0.1634	0.1649	129.26%	0.2650	0.2567	24.67%
	MeLU	0.1102	0.1051	20.59%	0.2439	0.2344	57.25%	0.2265	0.2207	45.44%
	MetaHIN	0.1260	0.1204	5.38%	0.2610	0.2628	43.68%	0.2569	0.2504	28.21%
	MetaKG	**0.1362**	**0.1236**	–	**0.3927**	**0.3598**	–	**0.3291**	**0.3213**	–
Item Cold-start (IC) New items for old users	FM	0.0888	0.0755	78.25%	0.3574	0.3234	38.23%	0.2055	0.1792	16.57%
	NFM	0.0844	0.0740	84.69%	0.3582	0.3192	39.02%	0.2084	0.1829	14.57%
	CKE	0.0888	0.0809	72.26%	0.3476	0.3112	42.92%	0.2167	0.1931	9.33%
	CFKG	0.0917	0.0775	73.14%	0.3523	0.3259	38.65%	0.2073	0.1809	15.51%
	UGRec	0.1349	0.1188	15.29%	0.2623	0.2334	90.00%	0.1716	0.1519	38.53%
	MCRec	0.0750	0.0691	102.85%	0.3605	0.3312	35.98%	0.2070	0.1573	24.47%
	RippleNet	0.0920	0.0737	77.37%	0.3090	0.2777	60.45%	0.1710	0.1492	40.05%
	KGCN	0.0904	0.0718	81.32%	0.3270	0.3111	47.27%	0.1748	0.1529	36.83%
	KGAT	0.0905	0.0788	72.83%	0.3445	0.3272	39.91%	0.2143	0.1902	10.78%
	CKAN	0.1360	0.1204	14.06%	0.2758	0.2946	65.00%	0.1762	0.1693	29.51%
	MeLU	0.0712	0.0620	119.68%	0.2217	0.2119	116.72%	0.1920	0.1869	18.11%
	MetaHIN	0.0670	0.0528	145.64%	0.2399	0.2266	101.47%	0.1990	0.1705	21.47%
	MetaKG	**0.1571**	**0.1356**	–	**0.4780**	**0.4616**	–	**0.2336**	**0.2141**	–
User-Item Cold-start (UIC) New items for new users	FM	0.1181	0.0949	53.67%	0.3367	0.3043	33.21%	0.2412	0.1825	30.10%
	NFM	0.1121	0.0982	54.95%	0.3390	0.3020	33.32%	0.2721	0.2265	9.60%
	CKE	0.1293	0.1156	32.99%	0.3118	0.2976	39.81%	0.2691	0.2206	11.71%
	CFKG	0.0992	0.0819	80.42%	0.3208	0.3013	37.02%	0.2444	0.2102	19.98%
	UGRec	0.1173	0.1023	48.41%	0.2571	0.2369	72.68%	0.2535	0.2216	14.72%
	MCRec	0.0910	0.0859	84.05%	0.1990	0.1857	121.63%	0.1800	0.1634	58.52%
	RippleNet	0.1150	0.1025	49.76%	0.2150	0.2110	99.89%	0.1969	0.1435	62.68%
	KGCN	0.1189	0.1004	48.81%	0.3473	0.3060	30.90%	0.2653	0.2298	10.13%
	KGAT	0.1498	0.1360	13.93%	0.3352	0.3182	30.41%	0.2670	0.2305	9.62%
	CKAN	0.1462	0.1166	24.62%	0.2854	0.2943	46.88%	0.2750	0.2273	8.85%
	MeLU	0.1308	0.1237	27.94%	0.2071	0.1875	116.37%	0.2498	0.2014	21.41%
	MetaHIN	0.1430	0.1382	15.82%	0.2030	0.1871	118.67%	0.2656	0.2285	10.39%
	MetaKG	**0.1747**	**0.1513**	–	**0.4227**	**0.4287**	–	**0.2857**	**0.2587**	–
Non-cold-start Old items for old users	FM	0.0916	0.0779	77.43%	0.2193	0.1948	103.42%	0.1168	0.1073	111.06%
	NFM	0.1002	0.0870	60.47%	0.2677	0.2344	67.91%	0.1368	0.1256	80.25%
	CKE	0.1353	0.1214	16.86%	0.2952	0.2804	45.95%	0.2233	0.2181	6.95%
	CFKG	0.0802	0.0693	100.98%	0.2850	0.2625	53.59%	0.1239	0.1162	96.82%
	UGRec	0.0592	0.0499	175.81%	0.1898	0.1927	119.55%	0.0398	0.0351	532.93%
	MCRec	0.0670	0.0529	152.29%	0.1220	0.1078	266.68%	0.0670	0.0619	266.84%
	RippleNet	0.0690	0.0631	126.97%	0.1910	0.1721	131.82%	0.1061	0.0864	147.96%
	KGCN	0.0645	0.0579	145.09%	0.2006	0.1882	116.15%	0.1117	0.0910	135.47%
	KGAT	0.1270	0.1166	23.07%	0.3004	0.2892	42.45%	0.2221	0.2188	7.06%
	CKAN	0.0868	0.0692	93.73%	0.2071	0.2042	104.16%	0.1040	0.0960	136.44%
	MeLU	0.1266	0.1183	22.38%	0.2828	0.2755	50.42%	0.1709	0.1630	41.46%
	MetaHIN	0.1420	0.1226	13.56%	0.3029	0.2886	42.02%	0.2160	0.2181	8.72%
	MetaKG	**0.1562**	**0.1436**	–	**0.4221**	**0.4176**	–	**0.2344**	**0.2376**	–

general knowledge to further improve the recommendation quality across tasks.

Non-cold-start Scenario. We also evaluate the performance of MetaKG considering non-cold-start scenario, as depicted in Table 3. As expected, our MetaKG still achieves the best performance among all methods, which further
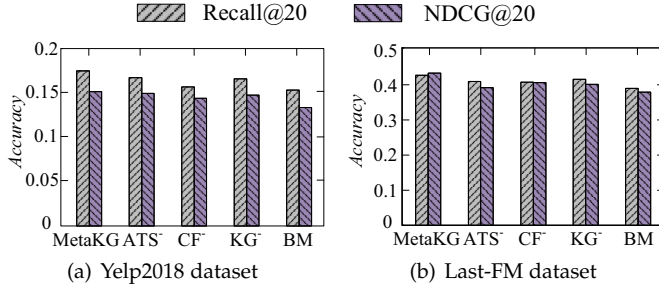
Fig. 4. Ablation Study of MetaKG in UIC Scenario

(a) Yelp2018 dataset (b) Last-FM dataset

Methods	Yelp2018		Last-FM		Amazon-book	
	T	I	T	I	T	I
NFM	4.9	1.5	20.6	7.4	7.5	1.2
KGAT	107.2	**0.3**	51.2	**0.3**	197.6	**0.3**
MetaHIN	220.1	4.8	97.5	9.1	65.3	1.5
MetaKG	**3.2**	**0.3**	**3.9**	**0.3**	**3.8**	**0.3**

confirms the robustness of MetaKG. The reason behind is that the evaluated datasets are very sparse (e.g., the sparsity of Yelp 2018 is 99.93%), and our method could alleviate the sparsity problems and have good generalization, thanks to the novel design of two meta-learners. Besides, it is worth mentioning that the performance of competitors are unstable among different scenarios. In contrast, our MetaKG model achieves the state-of-the-art recommendation performance considering both cold-start scenarios and non-cold-start scenario.

7.3 Ablation Study

In this subsection, we provide the insights into the effectiveness of each component in MetaKG. The results are shown in Fig. 4, where (i) "MetaKG" represents its completed model; (ii) "ATS$^-$" denotes MetaKG without adaptive task scheduler; (iii) "CF$^-$" denotes MetaKG without colloborative-aware meta learner; (iv) "KG$^-$" represents MetaKG without knowledge-aware meta learner; and (v) "BM" stands for the base model of MetaKG, which is trained without the meta-learning framework. Specifically, we optimize all the learnable parameters (i.e., θ, ϕ and ω) with BPR loss function, and directly use all training data to update them without the local update scheme. The details of base model can be refereed in Section 5.3.

MetaKG vs. ATS$^-$. Firstly, we remove the adaptive task scheduler module to verify its effectiveness. As observed, Recall@20 drops 3.45%, and NDCG@20 drops 5.07% on the Last-FM dataset, after sampling tasks uniformly as usual. Similar observations can also be found on the Yelp2018 and Amazon-book. This shows that adaptive task scheduler could effectively select informative tasks and filter out noisy tasks for better meta learners training.

MetaKG vs. CF$^-$. To verify the effectiveness of the collaborative-aware meta learner, we remove it by freezing the parameteres γ in the aggregation layer within tasks. As observed, taking Last-FM as an example, MetaKG improves the Recall@20 from 0.4096 to 0.4227, and boosts NDCG@20 from 0.4081 to 0.4287, compared with CF$^-$. Similar observations can also be found on the Yelp2018 and Amazon-book datasets. This shows that the collaborative-aware meta leaner can effectively capture *meta* high-order user-item relations to achieve better recommendation performance.

MetaKG vs. KG$^-$. The major difference between MetaKG and existing meta-learning models lies in whether multi-objective optimization is utilized to enrich semantic embeddings of entities in KG. By removing the knowledge-aware meta learner, we only leverage the initial embeddings to predict the user preference, without optimizing

the KG loss. As observed, Recall@20 drops 5.98%, and NDCG@20 drops 6.57%, after removing the knowledge-aware meta learner from MetaKG. This further proves that the knowledge-aware meta learner benefits the recommendation performance improvement.

MetaKG vs. BM. Finally, we explore the performance of MetaKG without meta-learning guiding, meaning that we only adopt the base model for cold-start recommendations. Compared with BM, MetaKG achieves significant improvement across all the metrics and datasets. The reason of the dramatic performance degradation is that the base model is trained without the carefully designed meta learners, and thus, it easily incurs overfitting when the data is scarce, and lacks the fast adaptation ability for new users or new items, as discussed in Section 1.

7.4 Efficiency Evaluation

Next, we inspect the model efficiency in terms of both model training (denoted as **T**) and inference (denoted as **I**) phases. Due to the limited space, we only compare MetaKG with the most popular traditional method (i.e., NFM), the state-of-the-art KG-based method (i.e., KGAT), and the state-of-the-art meta-learning method (i.e., MetaHIN), as depicted in Table 4. In addition, we only study the UIC scenario, as other scenarios show similar performance. Overall, MetaKG achieves high efficiency for both training and inference phases. Specifically, in terms of the training phase, MetaKG finishes each epoch within 3.2 seconds on Yelp2018, which is faster than the best competitor (i.e., NFM). In terms of the inference phase, MetaKG also performs well (i.e., only 0.3 seconds per epoch on all the datasets). This is because our MetaKG only needs a few gradients steps to gain good generalizations with the benefits of meta-learning, while the other methods need much more tune to get satisfying results. Note that, although MetaHIN is a meta-learning based approach, it is also inferior to MetaKG. This is because MetaHIN needs time-consuming process to construct semantic meta-paths, which is time-sensitive and not feasible for online recommendation.

7.5 Scalability Evaluation

We proceed to explore the scalability of MetaKG as it is important for online recommendation. We utilize three different kinds of state-of-the-art methods (i.e., NFM, KGAT and MetaHIN) as competitors, and vary the cardinality of dataset (i.e., the proportion of the test data size to the entire dataset) from 20% to 100%. Fig. 5 plots the UIC scenario results on Yelp2018 and Last-FM datasets, while other scenarios are omitted due to similar trends. The first observation is that the predicting time increases with the cardinality.
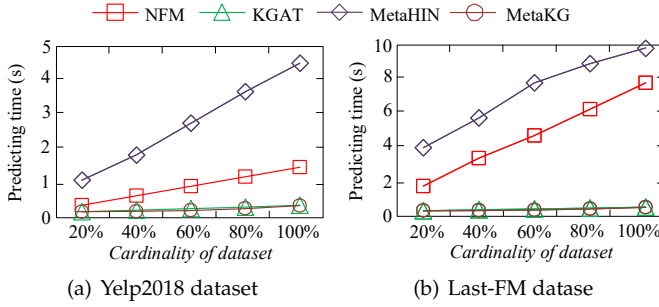
(a) Yelp2018 dataset   (b) Last-FM datase

Fig. 5. Scalability Evaluation in UIC Scenario

(a) Yelp2018 dataset   (b) Last-FM dataset
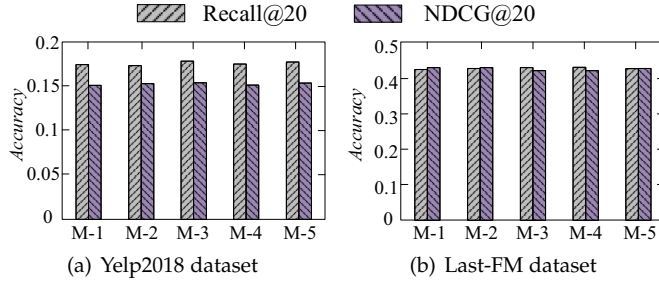
Fig. 6. Model Performance vs. the Number of Local Updates

(a) Performance at Recall@20   (b) Performance at NDCG@20

Fig. 7. Model Performance vs. Support Set Size

(a) Performance at Recall@20   (b) Performance at NDCG@20

Fig. 8. Model Performance vs. Embedding Dimension

Second, MetaKG offers significant performance improvement over existing methods, which costs 0.3 seconds to predict the entire yelp2018 dataset. Finally, the performance of MetaKG is less affected by the growth of cardinality than other methods. Note that, although KGAT achieves competitive performance in scalability, it fails to tackle cold-start scenarios as mentioned in Section 7.2. Overall, MetaKG is capable of large-scale online recommendation.

7.6 Parameter Analysis

Finally, we investigate the impact of parameters (i.e., the number of local updates, the support set size, and the embedding dimension) on the model performance. Here, we only utilize the UIC scenario due to the similar observations captured from other cold-start scenarios.

Effect of Number of Local Updates. We qualitatively analyze the effect of the local updates on the model performance, which leverages collaborative-aware meta learner to provide personalized preference information for recommendations. We denote the proposed model with m local updates as M-m. Fig. 6 depicts the results by changing numbers of local updates on Yelp2018 and Last-FM datasets. As observed, MetaKG achieves remarkable improvement on all the datasets and metrics. In addition, the performance of MetaKG is stable with the increase of the number of local updates. Thus, our MetaKG model can be adapt quickly to user preference because a single local update is enough and sufficient, indicating that MetaKG shows high potential for online recommendations.

Effect of Support Set Size. We verify the model performance when varying the size of support set (i.e., $|S_u|$) from 5 to 80. The corresponding results are shown in Fig. 7. Overall, the performance of MetaKG becomes better with the growth of the support set size (i.e., more training data). Nonetheless, the performance improves slowly as the size of support set ascends. This implies that MetaKG can achieve high performance via very few interactions/sampling data,
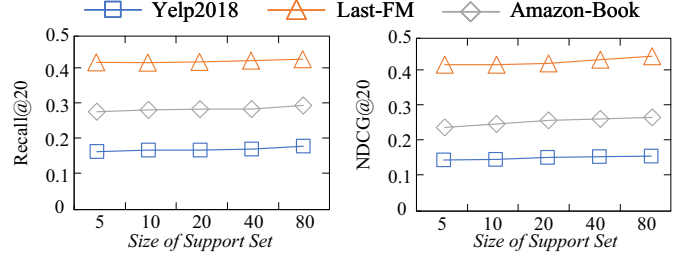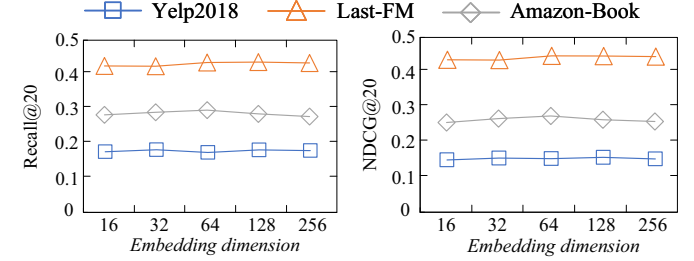
which validates the robustness of MetaKG especially in cold-start scenarios.

Effect of Embedding Dimension. Last but not the least, we explore how the embedding dimension d_e affect the model performance. The results are illustrated in Fig. 8. As observed, MetaKG achieves the best performance when the dimension is 64 in most cases. Also, the performance is stable with the growth of d_e, which confirms the robustness of MetaKG w.r.t the embedding dimension.

8 CONCLUSIONS

In this paper, we propose MetaKG, a meta-learning framework for cold-start recommendation on the knowledge graph, which consists of two meta learners, including the collaborative-aware meta learner and knowledge-aware meta learner, to achieve the generality within and across user preference learning tasks. Specifically, the collaborative-aware meta learner learns users' preference within each task, while the knowledge-aware meta learner enables to capture the semantic representations of KG shared across different tasks. Besides, we devise a novel adaptive task scheduler to select the informative task for meta learning. Extensive experiments on three real data sets demonstrate that MetaKG significantly outperforms all the state-of-the-art baselines with high efficiency and robustness. Overall, to the best of our knowledge, it is the first work to explore the meta-learning in KG-based recommendation, where we capture the prior knowledge for different learning process. In the future, we would like to explore interpretability in KG for explainable cold-start recommendations. In addition, combining our method with recent non-sampling learning methods is also attractive.

References

[1] S. Adam, B. Sergey, B. Matthew, W. Daan, and L. Timothy. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850, 2016.

[2] Q. Ai, V. Azizi, X. Chen, and Y. Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137–146, 2018.

[3] A. Bordes, N. Usunier, A. García-Durán, J. Weston, , and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795, 2013.

[4] H.-S. Chang, E. Learned-Miller, and A. McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NeurIPS*, pages 1002–1012, 2017.

[5] F. Chelsea, A. Pieter, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

[6] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *arXiv preprint arXiv:1412.3555*, 2014.

[8] M. F. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *RecSys*, pages 101–109, 2019.

[9] Z. Du, X. Wang, H. Yang, J. Zhou, and J. Tang. Sequential scenario-specific meta learner for online recommendation. In *KDD*, pages 2895–2904, 2019.

[10] S. Flood, Y. Yongxin, Z. Li, X. Tao, T. P. HS, and H. T. M. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[12] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1025–1035, 2017.

[13] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517, 2016.

[14] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–1780, 1997.

[17] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu. Leveraging meta-path based context for top- n recommendation with a neural co-attention model. In *KDD*, pages 1531–1540, 2018.

[18] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[19] S. Jake, S. Kevin, and R. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4080—-4090, 2017.

[20] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, pages 687–696, 2015.

[21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[22] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[23] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[24] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung. MeLU: Meta-learned user preference estimator for cold-start recommendation. In *KDD*, pages 1073–1082, 2019.

[25] L. Li, W. Chu, J. Langford, , and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010.

[26] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[27] Y. Lu, Y. Fang, and C. Shi. Meta-learning on heterogeneous information networks for cold-start recommendation. In *KDD*, pages 1563–1573, 2020.

[28] M. Luo, F. Chen, P. Cheng, Z. Dong, X. He, J. Feng, and Z. Li. Metaselector: Meta-learning for recommendation with user-level adaptive model selection. In *WWW*, pages 2507–2513, 2020.

[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[30] V. Oriol, B. Charles, L. Timothy, kavukcuoglu koray, and D. Wierstra. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638, 2016.

[31] F. Pan, S. Li, X. Ao, P. Tang, and Q. He. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In *SIGIR*, pages 695–704, 2019.

[32] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[33] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[34] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.

[35] V. Ricardo and D. Youssef. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, pages 77–95, 2001.

[36] R. Sun, X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang, and K. Zheng. Multi-modal knowledge graphs for recommender systems. In *CIKM*, pages 1405–1414, 2020.

[37] R. Togashi, M. Otani, and S. Satoh. Alleviating cold-start problems in recommendation through pseudo-labelling over knowledge graph. In *WSDM*, pages 931–939, 2021.

[38] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle. A meta-learning perspective on cold-start recommendations for items. In *NeurIPS*, pages 6904–6914, 2017.

[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[40] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*, pages 417–426, 2018.

[41] H. Wang, F. Zhang, X. Xie, and M. Guo. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*, pages 1835–1844, 2018.

[42] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *KDD*, pages 968–977, 2019.

[43] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo. Knowledge graph convolutional networks for recommender systems. In *WWW*, pages 3307–3313, 2019.

[44] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 29(12):2724–2743, 2017.

[45] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua. KGAT: Knowledge graph attention network for recommendation. In *KDD*, pages 950–958, 2019.

[46] X. Wang, X. He, M. Wang, F. Feng, and T. Chua. Neural graph collaborative filtering. In *SIGIR*, pages 165–174, 2019.

[47] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T. Chua. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*, pages 5329–5336, 2019.

[48] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, and T.-S. Chua. Reinforced negative sampling over knowledge graph for recommendation. In *WWW*, pages 99–109, 2020.

[49] Z. Wang, G. Lin, H. Tan, Q. Chen, and X. Liu. CKAN: Collaborative knowledge-aware attentive network for recommender systems. In *SIGIR*, pages 219–228, 2020.

[50] S. Wu, Y. Zhang, C. Gao, K. Bian, and B. Cui. Garg: anonymous recommendation of point-of-interest in mobile networks by graph convolution network. *Data Science and Engineering*, 5(4):433–447, 2020.

[51] H. Yao, Y. Wang, Y. Wei, P. Zhao, M. Mahdavi, D. Lian, and C. Finn. Meta-learning with an adaptive task scheduler. *NeurIPS*, 2021.

[52] H. Yu, T. Qian, Y. Liang, and B. Liu. AGTR: adversarial generation of target review for rating prediction. *Data Sci. Eng.*, 5(4):346–359, 2020.

[53] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, pages 283–292, 2014.

[54] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362, 2016.

[55] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1), 2 2019.

[56] X. S. Zhang, F. Tang, H. H. Dodge, J. Zhou, and F. Wang. Metapred: Meta-learning for clinical risk prediction with limited patient electronic health records. In *KDD*, pages 2487–2495, 2019.

[57] X. Zhao, Z. Cheng, L. Zhu, J. Zheng, and X. Li. UGRec: Modeling directed and undirected relations for recommendation. In *SIGIR*, pages 193–202, 2021.

[58] S. Zhu, Y. Jie, Z. Jie, B. Alessandro, H. Long-Kai, and X. Chi. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*, pages 297–305, 2018.

Yuntao Du received the BS degree in data science from East China Normal University, China, in 2020. He is currently working toward the Master degree in the College of Computer Science, Zhejiang University, China. His research interests include data mining and recommender system.

Xinjun Zhu received the BS degree in computer science from Zhejiang University of Technology, China, in 2021. He is currently working toward the Master degree in the School of Software, Zhejiang University, China. His research interests include spatio-temporal data mining and recommender system.

Lu Chen received the PhD degree in computer science from Zhejiang University, China, in 2016. She was an assistant professor in Aalborg University for a 2-year period from 2017 to 2019, and she was an associated professor in Aalborg University for a 1-year period from 2019 to 2020. She is currently a ZJU Plan 100 Professor in the College of Computer Science, Zhejiang University, Hangzhou, China. Her research interests include indexing and querying metric spaces, graph databases, and database usability.

Ziquan Fang received the BS degree in computer science from China Agricultural University, China, in 2018. He is currently working toward the PhD degree in the College of Computer Science, Zhejiang University, China. His research interests include trajectory data management and analytics.

Yunjun Gao received the PhD degree in computer science from Zhejiang University, China, in 2008. He is currently a professor in the College of Computer Science, Zhejiang University, China. His research interests include spatial and spatio-temporal databases, metric and incomplete/uncertain data management, graph databases, spatio-textual data processing, and database usability. He is a member of the ACM and the IEEE.