DEKR: Description Enhanced Knowledge Graph for Machine Learning Method Recommendation

Xianshuai Cao[1] Yuliang Shi[1,2,*] Han Yu[3] Jihu Wang[1]
Xinjun Wang[1,2] Zhongmin Yan[1] Zhiyong Chen[1]
[1]School of Software, Shandong University, Jinan, China
[2]Dareway Software Co., Ltd, Jinan, China
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
{xianshuaicao,jihu_wang}@mail.sdu.edu.cn,han.yu@ntu.edu.sg
{shiyuliang,wxj,yzm,chenzy}@sdu.edu.cn

ABSTRACT

The huge number of machine learning (ML) methods has resulted in significant information overload. Faced with an overwhelming number of ML methods, it is challenging to select appropriate ones for the given dataset and task. In general, the names of ML methods or datasets are rather condensed, thus lacking specific explanations, while the rich latent relationships between ML entities are not fully explored. In this paper, we propose a description-enhanced machine learning knowledge graph-based approach - DEKR - to help recommend appropriate ML methods for given ML datasets. The proposed knowledge graph (KG) not only includes the connections between entities but also contains the descriptions of the dataset and method entities. DEKR fuses the structural information with the description information of entities in the knowledge graph. It is a deep hybrid recommendation framework, which incorporates the knowledge graph-based and text-based methods, overcoming the limitations of previous knowledge graph-based recommendation systems that ignore the description information. There are two key components of DEKR: 1) a graph neural network aggregating information from multi-order neighbors with attention to enrich the seed (i.e. dataset or method) node's own representation, and 2) a deep collaborative filtering network based on the description text to obtain the linear and nonlinear interactions of description features. Through extensive experiments, we demonstrated the efficiency of DEKR, which outperforms the current state-of-the-art baselines by a large margin.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Machine learning**.

*Yuliang Shi is the corresponding author.

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

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462900

KEYWORDS

Recommender systems; knowledge graph; machine learning; description information

ACM Reference Format:
Xianshuai Cao, Yuliang Shi, Han Yu, Jihu Wang, Xinjun Wang, Zhongmin Yan, and Zhiyong Chen. 2021. DEKR: Description Enhanced Knowledge Graph for Machine Learning Method Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462900

1 INTRODUCTION

With the rapid advancement of scientific research and the growth of academic data, scholarly big data mining has attracted increasing attention in recent years. In order to unleash the full value of scholarly big data, research topics such as academic network analysis [6, 9, 20] and personalized knowledge recommendation [13, 17], have been more explored. The problem of information overload is particularly severe in the field of machine learning (ML), which has witnessed an explosion of ML methods in the last decade with newer models still being continuously proposed. This makes selecting the appropriate ML method for a given dataset or task a challenge. Thus, automatically recommending such ML methods could significantly improve the quality and efficiency of such effort.

Motivated by this, we perform an analysis of the ML data characteristics and have the following findings. Firstly, relationships between ML methods and datasets can be established through other entities (e.g., tasks, authors). Therefore, mining the rich connections between entities can improve the quality of recommendations and make the recommendation framework more explainable. Secondly, the names of ML datasets and methods are often abbreviated and not easily understood. Thus, the associated descriptive information can offer more specific elaboration and explanation, which can further improve the recommendation results.

In this paper, we leverage knowledge graphs (KGs) as the auxiliary information to enhance ML recommendation. A KG is a heterogeneous graph composed of numerous entities and relationships. Successful applications in the fields such as news recommendation [23] and book recommendation [28] have demonstrated the value of KGs to recommender systems. Specifically, we have constructed a description-enhanced machine learning knowledge graph by mining ML entities and defining their relationships according to actual practice. Moreover, we treat the ML datasets and methods as the

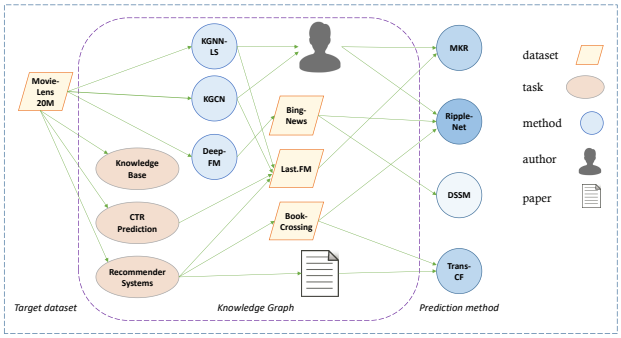

Figure 1: An example illustrating how the rich connections in the machine learning knowledge graph can be used to recommend the appropriate method for a given dataset.

core entities, linking them to their corresponding description information. Finally, the knowledge graph will be used as side information to improve the performance of machine learning method recommendations.

As shown in Figure 1, given a target dataset *MovieLens-20M*, it is possible to obtain the links to its relevant tasks (e.g., CTR Prediction) or adopted methods (e.g., KGCN). These linked entities connect to other entities, such as the datasets of the same task as well as the authors or the papers that proposed the methods. For example, it is obvious that $MovieLens\text{-}20M \xrightarrow{dataset.task} CTR\ Prediction \xrightarrow{task.dataset} Last.FM$ shows the similarity between the two datasets with regard to the applicable scenarios, while the longer path $MovieLens\text{-}20M \xrightarrow{dataset.task} CTR\ Prediction \xrightarrow{task.dataset} Last.FM \xrightarrow{dataset.method} MKR$ suggests that we can reasonably infer that *MKR* is likely to be applicable to *MovieLens-20M*. Moreover, the probability for finding *MKR* with the same author as *KGCN* is higher according to Figure 1, which helps generate the right recommendation. The above chain of inference is key to knowledge graph-based recommender systems [22, 25].

However, if we solely depend on the graph structure information without considering the detailed description of the query entities, the recommendations generated can be less well-suited at times. For example, simply relying on the link connection with path $MovieLens\text{-}20M \xrightarrow{dataset.method} DeepFM \xrightarrow{method.dataset} Bing\text{-}News \xrightarrow{dataset.method} DSSM$, we can reasonably assume that *DSSM* has a higher probability to be applicable than other entities without higher-order connections to the *MovieLens-20M* dataset. However, *DSSM* is actually more oriented towards dealing with the text ranking problems. Yet, if the description information of the two entities (i.e. ***MovieLens-20M*: a widely used benchmark dataset in movie recommendations; *DSSM*: a deep structured semantic model for document ranking**) were taken into account, this suboptimal recommendation in the above can be avoided.

Based on these considerations, we propose the Description Enhanced Knowledge Graph Recommendation (DEKR) model. It enhances the knowledge graph structural information with text descriptions to generate improved ML method recommendations.

In summary, our contributions in this paper are as follows.

- We construct a description-enhanced machine learning knowledge graph that considers not only the rich connections between machine learning entities but also the descriptive information of query (i.e. datasets and methods) entities.
- We propose the DEKR model, which combines knowledge graph-based and text-based approaches to overcome the limitation of traditional knowledge graph-based approaches where the descriptive information of entities is ignored.
- Extensive experiments have been carried out on real-world dataset, and the results show the efficiency of the DEKR model, which outperforms the state-of-the-art baseline by more than 15%.

2 RELATED WORK

In this section, we review existing works on scholarly big data mining, knowledge graphs, and knowledge graph-based recommender systems which are most relevant to ours.

2.1 Scholarly Big Data Mining

The basic motivation of scholarly big data mining is to mine useful knowledge from scholarly big data for scholars, as well as helping people better understand the intrinsic laws of science itself. The two major research directions are academic network analysis and academic recommendation. The academic network can be mainly divided into citation networks [6, 34], co-occurrence networks, such as co-author [15], co-citation [39], co-words [14], and hybrid networks [35], where citation networks and co-occurrence networks are usually homogeneous graph networks while hybrid networks are often heterogeneous graph networks. Citation networks are frequently used to analyze the influence of a given paper or scholar [6]. Co-occurrence networks are typically used for the discovery of academic communities [5]. Hybrid networks allow us to analyze complex relationships between a more diverse category of entities [19]. Academic recommendation aims to solve the information overload problem, mainly including paper recommendation [1, 31], collaboration recommendation [4, 11], and avenue recommendation [2, 30].

As we know, machine learning is a rapidly growing research field. The large amount of data generated are consistent with the general pattern of scholarly big data, yet with its own specific characteristics, such as a huge number of datasets and numerous methods. However, the related data mining research has not been well explored [10, 36]. An efficient machine learning recommendation system is necessary to solve the problem of data overload. Specialized machine learning graph networks are promising for analyzing the complex relationships among the ML entities.

2.2 Knowledge Graph

A knowledge graph is a heterogeneous graph as it contains multiple types of nodes and edges, where nodes represent entities and edges represent relationships between entities. Because of the rich facts and connections, KGs have been deployed in diverse applications such as question answering [7], text classification [26] and news recommendation [23]. Furthermore, to improve the quality of knowledge, the relationship types [12] or textual information

[32], and even visual signals [33] have been added as auxiliary information to the knowledge graph.

In this paper, by defining the machine learning related entities and the relationships between them, together with adding descriptive information to the core entities(i.e. datasets and methods), we have constructed a knowledge graph for machine learning and applied it to the downstream recommendation task. Specially, in order to avoid weakening the semantic representation of the description text, we do not jointly learn the text representation with the structure representation. Instead, we adopt a TextCF network for the text and a KGNN for the structure of the graph, respectively, to obtain the integrated representation of entities.

2.3 KG-based Recommender Systems

Incorporating knowledge graphs as side information has been shown to improve the precision of recommendations, as it can overcome both the data sparsity and the cold start problems while enhancing the explainability of recommendations. Knowledge graph-based recommender systems can be classified into three categories: 1) embedding-based approaches, 2) path-based approaches, and 3) hybrid approaches. Embedding-based approaches directly use the information in KG to enrich the representations of users or items [24, 38], and entity embedding is the core step. Path-based approaches use the pattern and similarity of the connections from entities in the knowledge graph to enhance the recommendations [29, 37]. They usually require manually defined meta-paths. Hybrid approaches integrate the semantic information of embeddings with paths in the knowledge graph via the embedding propagation to enrich its own representations through multiple neighbors [21, 22].

The DEKR model proposed in this paper belongs to the category of hybrid models. Existing hybrid models usually focus on structural information and overlook the additional information from entities for embedding representation when integrating the knowledge graph embedding and path information. In contrast, we add descriptive information to the core entities and learns the textual representation corresponding to them. This enhances the embedded representation part and, thus, improves the precision of the final recommendations.

3 PROBLEM FORMULATION

We formulate the description-enhanced knowledge-aware machine learning method recommendation problem as follows. Here, we let $\mathcal{D} = \{d_1, d_2, ..., d_M\}$ denote the set of machine learning datasets, $\mathcal{M} = \{m_1, m_2, ..., m_N\}$ denote the set of machine learning methods. The dataset-method interaction matrix $Y = \{y_{dm} \mid d \in \mathcal{D}, m \in \mathcal{M}\}$ can be expressed as:

$$y_{dm} = \begin{cases} 1, & \text{if } (d, m) \text{ interaction record exists} \\ 0, & \text{otherwise .} \end{cases} \quad (1)$$

Meanwhile, our machine learning knowledge graph is used as auxiliary information to perform the recommendation task. The knowledge graph $\mathcal{G}$ is composed of a large number of (h, r, t) triples, where $h \in \mathcal{E}$ is the head entity, $t \in \mathcal{E}$ is the tail entity, and $r \in \mathcal{R}$ is the relationship between the head entity and the tail entity. $\mathcal{E}$ is the set of entities in the knowledge graph, and $\mathcal{R}$ is the set of relations. For example, *(MovieLens-20M, dataset.task, CTR Prediction)* expresses that the *MovieLens-20M* dataset can be used for the *CTR Prediction* task.

In addition, our proposed description-enhanced knowledge graph adds description information to both core entities (i.e. dataset and method). In particular, $t_d \in \mathcal{T}$ and $t_m \in \mathcal{T}$ represent the description texts associated with the dataset and method, respectively. They are often natural language summaries outlining their core topics and key features. $\mathcal{T}$ refers to the set of description documents.

In general, under the recommendation scenario, given a user and an item, there are entities with the same name corresponding to them in the knowledge graph. In this paper, the dataset is treated as the user and the method is treated as the item. For example, dataset instance *Movielens-20M* and method instance *KGCN* correspond to the entities with the same name in the knowledge graph. In this way, the rich information in the machine learning knowledge graph (MLKG) can be used as side information to enhance the method recommendation task.

Based on the given interaction matrix Y and the description-enhanced knowledge graph $\mathcal{G}$, we aim to predict whether the dataset will interact with candidate methods that did not appear in the interaction record of the dataset. Our goal is to learn a prediction function $\hat{y}_{dm} = \mathcal{F}(d, m; \Theta)$, where $\hat{y}_{dm}$ denotes the probability of interaction between a dataset d and a method m, and Θ is the parameter inside the prediction function $\mathcal{F}$.

4 PROPOSED DEKR APPROACH

In this section, the DEKR model will be presented in detail. We will first present the overall architecture of DEKR. Then, we will introduce the two key components of the model: 1) the graph neural network which explores high-order connectivity by allowing two seed (i.e. dataset and method) entities to propagate over the knowledge graph, enriching their own representation with the information from neighbors; and 2) the deep text-based collaborative filtering network which derives text-based interaction probabilities by capturing linear and nonlinear interactions from descriptive features.

4.1 Overall Architecture

The overall framework of DEKR is illustrated in Figure 2, we will introduce the architecture from left to right. As shown in Figure 2, the input consists of a target machine learning dataset d and a candidate machine learning method m. Firstly, the two instances d and m are linked to the corresponding entities e_d and e_m on the knowledge graph with the same names as theirs.

Then, the linked entities, e_d and e_m, are used as seed nodes to traverse the knowledge graph $\mathcal{G}$. As shown in Figure 3, at the first expansion of the seed node e (e.g., *Movielens-20M* in Figure 3), two types of nodes will be distinguished. The first-order nodes without descriptive information are defined as general nodes $\{e_1^g, e_2^g, ..., e_n^g\}$, such as *KGCN* and *RS* in the figure. The first-order nodes with descriptive information are defined as descriptive nodes e^t, such as the square node *Description*. They serve different purposes in later stages. More specifically, the general nodes e_i^g continue to expand outward towards higher-order neighbor nodes $\{e_{n+1}^g, e_{n+2}^g, ..., e_m^g\}$, such as *CTR* and *Paper*. During the expansion, since the aim is to obtain the relationship structure information of the seed nodes

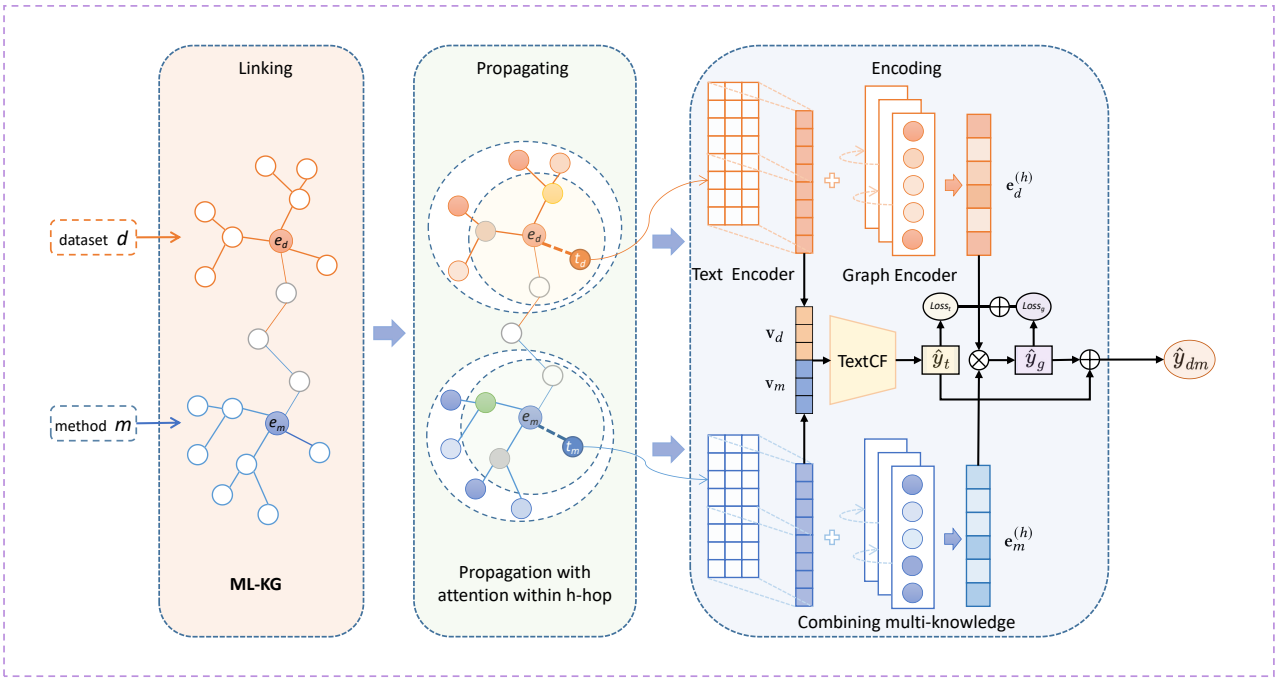

Figure 2: Illustration for the overall architecture of DEKR, which leverages the description-enhanced machine learning knowledge graph to integrate information from both graph structure and description text. It takes a dataset and a method as input and outputs the predicted probability of interaction.

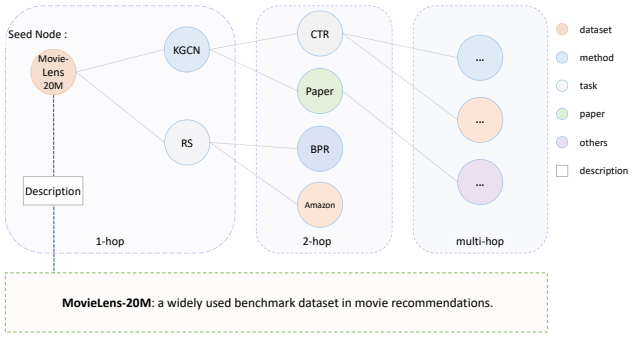

Figure 3: An illustration of the propagation of a seed node (MovieLens-20M) in the MLKG. A distinction is made between general nodes (without descriptive information) and descriptive node (with descriptive information) in the first hop. Then, the general nodes continue to propagate in higher hops.

e_d and e_m, the description information of the expanded nodes $\{e_1^g, e_2^g, ..., e_n^g, ..., e_m^g\}$ is not considered, but are treated as general nodes. In contrast, the descriptive nodes e^t are not expanded.

Among the two types of nodes obtained in the above step, the general nodes $\{e_{d1}^g, e_{d2}^g, ..., e_{dk}^g\}$ and $\{e_{m1}^g, e_{m2}^g, ..., e_{ml}^g\}$, extended by the seed nodes e_d and e_m, their information is propagated through a multilayer graph neural network to generate graph embedding representations (to be explained in subsection 4.2). Meanwhile, for the descriptive nodes e_d^t and e_m^t, their description information is applied to obtain the linear and nonlinear interactions through a deep text-based collaborative filtering network (to be presented in subsection 4.3).

Finally, by combining both structural and descriptive information, the probability of interaction between the given dataset and method can be obtained. The details of this process are explained in subsection 4.4.

4.2 Knowledge Graph Neural Network (KGNN)

Intuitively, in a graph structure, the neighboring nodes around a node reflect its own characteristics, thus providing evidence for in-depth reasoning. Specifically, in MLKG $\mathcal{G}$, benefit from the rich connections between entities in it, the similarities and differences among nodes can be captured. For example, the method m_j employed for dataset d_i can be used as features to characterize d_i. Moreover, other datasets that use the same methods as d_i may share some similarities with it. Likewise, ML methods designed for the same task are more likely to be applied to the same dataset. To some extent, it is similar to the idea of collaborative filtering.

Based on this ideology, we perform information propagation on our machine learning knowledge graph $\mathcal{G}$ via the rich connections linking to the dataset and method entities e_d and e_m to enhance

their own representations with information from their neighbors. There are two main steps: propagation and aggregation. We will introduce the propagation process of one-hop at first, then expand the propagation to multiple-hops. Finally, we will explain the interaction prediction based on the graph structures.

4.2.1 ***One-hop Propagation.*** In popular knowledge graph-based recommender systems [22, 25], there are two major ways of information propagation. One is to start from the user end; use the item entities that exist in the user's history interaction records as the seeds for propagation; augment the embedding representation of the user with its extended neighbors; then predict the interaction probability by dot product with the item representation [22]. The other starts from the item end; uses the item as the seed; augments the embedding representation of the item with its extended neighbors; then computes the probability by dot product with the user representation [25].

Although the above two methods utilize the idea of collaborative filtering, they only consider the side information about the item but do not make full use of the side information about the user. It is reasonable that adding the user side information can further improve the performance of the recommender system. Particularly, in this paper, datasets in the MLKG act as users. They are similar to the method entities (which act as items) and can serve as seeds linking to their surrounding neighbors to enhance their own representations by leveraging the rich side information in the knowledge graph.

Given a dataset-method-pair - e_d and e_m - as the seed entities, they can link to other entities through several different relationships with their own immediate first-order neighbors $\mathcal{N}(e)$. Generally, among the first-order neighbors of the seed entities, the contribution to the seed entities varies due to the different relationships linked to them. For example, when recommending methods to the dataset d_a, in most cases, the methods employed by the dataset d_b, which serve the same task as that dataset d_a, are more likely to be applied to d_a than other methods from the same author with which the dataset d_a have interacted. This is because a single author may work on multiple research directions. Thus, it is important to distinguish different relationships r of node e by learning different attention weights $\pi_{r,e}$.

Hence, we calculate the attentional scores as follows:

$$\pi^d_{r,e} = \frac{\exp(\pi^d_{r,e})}{\sum_{e\in\mathcal{N}(m)}\exp(\pi^d_{r,e})}, \tag{2}$$

$$\pi^m_{r,e} = \frac{\exp(\pi^m_{r,e})}{\sum_{e\in\mathcal{N}(d)}\exp(\pi^m_{r,e})}, \tag{3}$$

where $\pi^d_{r,e}$ and $\pi^m_{r,e}$ represent the weights calculated by the $\mathcal{N}(m)$ and $\mathcal{N}(d)$ with reference to e_d and e_m, respectively. Commonly, during propagation, the number of neighbors of each entity varies, which reflects the real-world situation. Thus, in order to make the training easier and more efficient, we set a fixed number for sampling.

Based on the sampled neighbors $\mathcal{N}(e)$ and their corresponding attention weights $\pi_{r,e}$, we can calculate a weighted sum to obtain the aggregated representation of the entity's neighbors. It will be subsequently used to enhance the representation of the entity itself, which can be calculated as follows:

$$\mathbf{e}_{\mathcal{N}(e)} = \sum_{e\in\mathcal{N}(e)} \pi_{r,e}\mathbf{e}. \tag{4}$$

Finally, we obtain the representation of e by aggregating the representations of its neighbors into its own. The formula is as follows:

$$\mathbf{e}^{(1)} = \text{LeakyReLU}(\mathbf{W}_0(\mathbf{e}^{(0)} + \mathbf{e}^{(0)}_{\mathcal{N}(e)}) + \mathbf{b}_0). \tag{5}$$

Here, LeakyRelu is the activation function, which can handle both positive and negative signals. $\mathbf{W}_0$ and $\mathbf{b}_0$ are learnable parameters. In addition, the commonly used ways when aggregating neighboring node representations are sum, concatenation, and bi-interaction [27]. We choose the simple but efficient sum aggregator.

4.2.2 ***Multi-hop Propagation.*** Similar to the one-hop propagation process, we can further extend the process ahead to obtain higher-order neighbors by using the previous neighbors as new seed nodes. Taking the 2nd-order case as an example, in order to aggregate the 2nd-order neighbors' information to the seed node, firstly, the 2nd-order neighbors' information is aggregated to the 1st-order neighbors to obtain $\mathbf{e}^{(1)}_{\mathcal{N}(e)}$. Then, the 1st-order neighbors' information is aggregated to the seed node to obtain $\mathbf{e}^{(2)}$.

Iteratively, we can define:

$$\mathbf{e}^{(k)} = \text{LeakyReLU}(\mathbf{W}_{k-1}(\mathbf{e}^{(k-1)} + \mathbf{e}^{(k-1)}_{\mathcal{N}(e)}) + \mathbf{b}_{k-1}). \tag{6}$$

Note that as the order of propagation increases, the number of aggregations also increases. The two quantities are equal since each additional layer of expansion requires one more aggregation.

4.2.3 ***Prediction Layer.*** After propagating at both the dataset and the method ends separately for h hops, the eventual representations $\mathbf{e}^{(h)}_d$ and $\mathbf{e}^{(h)}_m$ has incorporated the information of their respective neighbors in the range of h-hop. As is commonly done, we derive the interaction probability based on the graph structure by performing dot product on $\mathbf{e}^{(h)}_d$ and $\mathbf{e}^{(h)}_m$ as follows:

$$\hat{y}_g = \sigma((\mathbf{e}^{(h)}_d)^{\mathrm{T}}\mathbf{e}^{(h)}_m). \tag{7}$$

4.3 Text-based Collaborative Filtering (TextCF)

When using the knowledge graph as side information, traditional knowledge graph-based recommendation systems usually perform representation learning on the information in the knowledge graph. With representation learning, the nodes and relationships in the knowledge graph are mapped to a low-dimensional vector space in which the relationships between entities are preserved [3]. Although the knowledge graph preserves the structural information after representation learning, the descriptive information for entities is missing. Apart from that, although entities can be distinguished from each other by different embedding representations, the embeddings of the graph structure and that of the descriptive text corresponding to the entities tend to be different [23]. Thus, the natural language descriptive information corresponding to the entities has been poorly exploited.

To overcome the above-mentioned shortcomings, we add descriptive information $\mathcal{T}$ to the core entities (i.e. datasets and methods) to augment the graph structure information and improve the

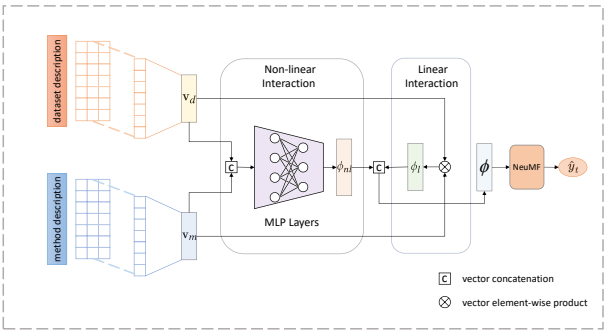

Figure 4: Architecture of TextCF.

recommendation performance. More specifically, given a machine learning dataset d or a method m, we can obtain the corresponding textual description information t_d or t_m by one-hop propagation. Then collaborative filtering based on textual information is performed with the help of deep neural networks in order to compute the probability of interaction between d and m based on t_d and t_m.

4.3.1 ***Text Embedding.*** Given the descriptive text corresponding to an entity, we use $t = w_{1:n} = [w_1, w_2, ..., w_n]$ to denote the initial input text with n words. Let p denote the dimension of each word vector, then $\mathbf{s}_{1:n} \in \mathbb{R}^{p \times n}$ denotes the embedding matrix of sentences. Here, considering both efficiency and the application scenario, we initialize the embedding representation of each word w_i with the pre-trained word vector of GloVe [16]. The GloVe model effectively utilizes global statistical information and local contextual features. We obtain the sentence representation $\mathbf{s}_t$ by aggregating the representation of each word, where $\mathbf{s}_t \in \mathbb{R}^p$. In aggregation, we use only simple averaging due to its validity.

4.3.2 ***Deep Collaborative Filtering.*** Through the text embedding, we have obtained the text vectors $\mathbf{s}_d$ and $\mathbf{s}_m$ for the dataset and the method descriptions, respectively. Then, we obtain two low-dimensional dense vectors, $\mathbf{v}_d$ and $\mathbf{v}_m$, as the hidden vectors via their transformation matrix, which can be formulated as follows:

$$\begin{aligned} \mathbf{v}_d &= \mathbf{W}_d \mathbf{s}_d, \\ \mathbf{v}_m &= \mathbf{W}_m \mathbf{s}_m. \end{aligned} \tag{8}$$

In order to better capture the interaction information between the two feature vectors, we employ a neural collaborative filtering framework [8] for its simplicity and efficiency. Specifically, we use the GMF and the MLP layers to derive the linear and nonlinear interactions of the dataset and method description features, respectively. Then, we fuse the two types of information through the NeuMF layer to compute the final prediction probabilities. The overall structure of deep collaborative filtering is shown in Figure 4. To obtain the linear interaction corresponding to the dataset and the method description features, the idea is similar to matrix factorization:

$$\phi_l = \mathbf{v}_d \odot \mathbf{v}_m, \tag{9}$$

$$\hat{y}_l = \sigma(\mathbf{G}^{\mathrm{T}} \phi_l). \tag{10}$$

Here, $\odot$ denotes the element-wise product. σ represents the activation function. $\mathbf{G}^{\mathrm{T}}$ represents the weight matrix, which is learnable so that different weights are assigned to different latent dimensions. Thus, the above process can be treated as a variant of matrix factorization with stronger expressiveness than its original form. As we aim to learn a linear interaction of description features, so we adopt a sigmoid activation function. That is, $\sigma(x) = \frac{1}{1+e^{-x}}$.

While obtaining the linear interaction of the description features, the non-linear interaction of the two is obtained via another way. Specifically, we first concatenate the two feature vectors $\mathbf{v}_d$ and $\mathbf{v}_m$. Then, we capture the nonlinear interaction between the features through a multilayer perceptron, by which we can obtain more summarized features for the interaction of $\mathbf{v}_d$ and $\mathbf{v}_m$. It can be formulated as follows:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{v}_d \parallel \mathbf{v}_m, \\ \phi_{nl} = \mathbf{h}_n &= \sigma(\mathbf{W}_n^{\mathrm{T}} \mathbf{h}_{n-1} + \mathbf{b}_n), \end{aligned} \tag{11}$$

$$\hat{y}_{nl} = \sigma(\mathbf{M}^{\mathrm{T}} \phi_{nl}), \tag{12}$$

where $\parallel$ denotes the concatenation operation of vectors.

To organically fuse the linear and nonlinear interaction of the description features, we concatenate the hidden vectors of the last layer of both networks then pass a NeuMF layer. This fuses the linear and nonlinear features ϕ_l and ϕ_{nl} to better learn the implicit interaction of the description text features, as well as to predict the final interaction probability:

$$\hat{y}_t = \sigma(\mathbf{W}_t^{\mathrm{T}} (\phi_l \parallel \phi_{nl})). \tag{13}$$

4.4 Combining KGNN and TextCF

KGNN and TextCF have learned the embedding representations of datasets and methods in terms of both structural and descriptive features, respectively. Then, based on these two types of representations, we can further predict the two probabilities $\hat{y}_g$ and $\hat{y}_t$. Thus, the next question is how to fuse these two kinds of information to jointly improve the final prediction probabilities.

A straightforward solution is to concatenate or sum up the representations learned from KGNN and TextCF, then predict interaction probability via dot product or neural network based on the combined features. However, the two types of information belong to different types, and their respective embeddings are obtained via different learning methods. Moreover, the embedding space of graph structure information and the descriptive text information tend to be different. Therefore, we let the model learn and optimize the two representations separately, and then combine the probabilities predicted by them to generate the final interaction probability as follows:

$$\hat{y}_{dm} = \mathbf{W}^{\mathrm{T}} (\sigma((\mathbf{e}_d^{(h)})^{\mathrm{T}} \mathbf{e}_m^{(h)}) + \sigma(\mathbf{W}_t^{\mathrm{T}} (\phi_l \parallel \phi_{nl})). \tag{14}$$

We empirically compare different combination methods in subsubsection 5.4.2.

The complete loss function can be constructed as:

$$\mathcal{L} = \sum_{d \in \mathcal{D}, m \in \mathcal{M}} (\mathcal{J}(\hat{y}_g, y_{dm}) + \mathcal{J}(\hat{y}_t, y_{dm})) + \lambda \|\Theta\|_2^2, \tag{15}$$

where $\mathcal{J}$ is the cross-entropy function, λ controls the L_2 regularization strength, and Θ is the parameter set.

5 EXPERIMENTAL EVALUATION

In this section, we conduct extensive experiments on a real-world dataset to evaluate DEKR. We aim to answer the following questions:

- **Q1**: How does DEKR perform compared to the state-of-the-art baseline models?
- **Q2**: How does each of the key components of DEKR affect the model?
- **Q3**: Does the addition of descriptive information enhance the performance of the recommender system interpretably?

5.1 Dataset

We collected data from open academic platforms (e.g., Paperswithcode, Github) searching for machine learning (ML) related datasets, methods, and their properties, as well as other related entities. The data we collected covers 19 areas (e.g., computer vision, natural language processing, reinforcement learning, and graphs). It consists of 2,093 ML datasets, 7,644 ML methods, 517 types of ML tasks, 4,338 academic papers, and 2,872 open source repositories. After data cleaning and pre-processing, we preserve 2,092 ML datasets and 6,239 ML methods.

Based on these data, we define the entities and relationships of machine learning, construct a description-enhanced MLKG, which adds descriptive information for the dataset and method entities. The sources of descriptive information not only include their original abstract, but also the background of the tasks which employ the datasets, and the titles of the papers corresponding to the methods. As shown in Figure 5, we have presented the properties of the machine learning entities and the relationships among them. Besides, we show the basic statistics of our dataset in Table 1.

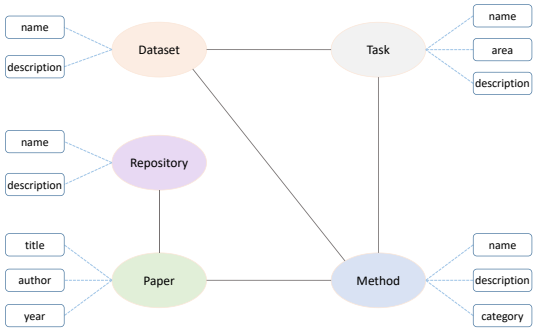

Figure 5: An illustration of major entities, their attributes, and relationships in the MLKG.

Table 1: Statistics of our dataset.

Knowledge Graph		Extracted Dataset	
# Entities	17483	# Datasets	2092
# Realtions	23	# Methods	6239
# Triples	117245	# Interactions	13732
Avg.# descriptive words	8.1	# Density	0.00105

5.2 Experiment Settings

5.2.1 ***Evaluation Metric.*** For each ML dataset, we predict the probability of its interaction with the candidate method. To evaluate DEKR, we adopt commonly used metrics AUC, accuracy, and F1-score. Moreover, we performed the evaluation of top-K recommendations by calculating precision@K, recall@K, and ndcg@K.

5.2.2 ***Baseline Methods.*** We compare DEKR with state-of-the-art hybrid knowledge graph-based recommendation models, traditional recommendation models as well as models that combine graph structure and textual information:

- **BPR** [18]: This method is based on the traditional FM model and optimized using Bayesian analysis.
- **CKE** [38]: This method is based on a regularization approach that integrates structural, text-based, visual information and CF in a unified framework. In this paper, we set up two alternatives: 1) only using graph structure and 2) using graph structure with text information.
- **RippleNet** [22]: This method is a memory-network-like approach to perform recommendation by propagating user preferences over KG.
- **KGCN** [25]: This method uses graph convolutional networks to obtain higher-order neighbor information on the knowledge graph and assigns different weights according to the relationship to select neighbors that the user is interested in.
- **KGAT** [27]: This method is a GNN-based method incorporated with a graph attention mechanism to evaluate the importance of different neighbors.

5.2.3 ***Parameter Settings.*** We implemented DEKR on Pytorch. We set the graph structure embedding and text embedding dimensions to 64, the number of node neighbors sampled to 8, and the number of graph convolution iterations to 2. We use the Adam optimizer for all the models and set the batch size to 128. For the learning rate and regularization coefficients, we select them from $\{10^{-4}, 5\times10^{-4}, ..., 10^{-1}, 5\times10^{-1}\}$ and $\{10^{-7}, 10^{-6}, ..., 10^{-3}, 10^{-2}\}$ respectively with grid search. The embedding size of all baselines is set to 64. For RippleNet, KGCN, and KGAT, we set the number of hops to 2. For RippleNet, we set the memory size to 32. For KGCN, we set the number of samples to 8 and the aggregator to sum. For KGAT, we set the aggregator to Bi-Interaction, and use node dropout as suggested in [27].

5.3 Performance Comparison (Q1)

The results of all methods in CTR prediction and top-K recommendation experiments are shown in Table 2 and Figure 6, respectively. We have the following observations:

- DEKR outperforms all the baselines on the CTR prediction task and achieves on average more than 15% improvement over the state-of-the-art models. Moreover, DEKR shows excellent performance on top-K recommendations.
- Almost all of the knowledge graph-based models outperform the traditional models, indicating that the introduction of rich relationships and information in the knowledge graph strongly enhances the effectiveness of recommendation.
- The hybrid knowledge graph-based methods KGAT, RippleNet, and KGCN achieve better performance compared

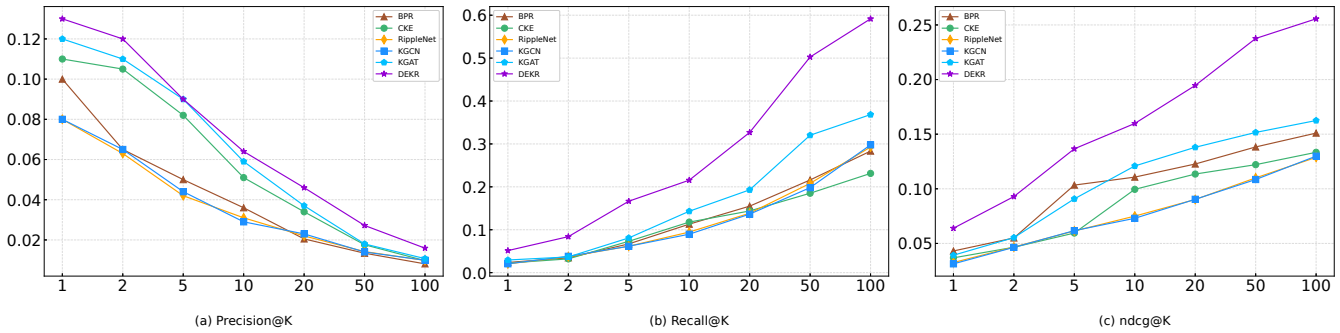

Figure 6: Precision@K, Recall@K, and ndcg@K in top-K recommendations.

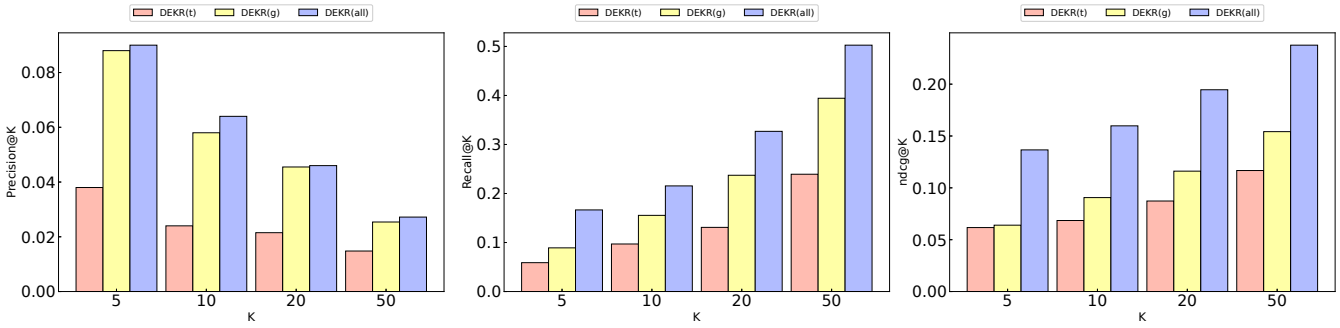

Figure 7: The impact of graph structure and description text on model performance.

Table 2: Overall Comparison in CTR prediction.

Models	AUC	Accuracy	F1-score
BPR	0.7519 (−21.6%)	0.6465 (−26.5%)	0.6861 (−22.7%)
CKE(g)	0.7152 (−25.3%)	0.6219 (−29%)	0.6721 (−24.1%)
CKE(g+t)	0.7695 (−19.8%)	0.6536 (−25.8%)	0.6967 (−21.7%)
RippleNet	0.8272 (−14.1%)	0.7267 (−18.5%)	0.7462 (−16.7%)
KGCN	0.8108 (−15.7%)	0.7338 (−17.8%)	0.7429 (−17%)
KGAT	0.8356 (−13.2%)	0.7407 (−17.1%)	0.7720 (−14.1%)
DEKR	**0.9679**	**0.9119**	**0.9133**
Improve-avg.	+18.3%	+22.5%	+19.4%

with other baseline methods. However, none of them outperforms DEKR, since they ignore the entity-related textual descriptions when using knowledge graphs.

- The CKE model using only graph structure information is weaker than the traditional model BPR, which indicates that the regularization-based approach may not make good use of the knowledge in the knowledge graph. On the one hand, CKE is only regularized by the correct triples in the knowledge graph. On the other hand, it ignores the higher-order connectivity.
- The CKE variant incorporating graph structure information and textual information outperforms the CKE variant using only graph structure information. This further validates the theory that entity-related descriptive information can enhance the performance of the recommender system.

Table 3: Comparison of different DEKR variants.

Models	AUC	Accuracy	F1-score
DEKR with single-end KG only	0.8738	0.7837	0.7992
DEKR without descriptive text	0.9279	0.8365	0.8518
DEKR without graph structure	0.9302	0.8631	0.8672
DEKR	**0.9679**	**0.9119**	**0.9133**

5.4 Study of DEKR (Q2)

5.4.1 ***Ablation Analysis.*** To investigate whether the two types of knowledge used by DEKR, graph structure information and text description information, both contribute to the effectiveness of the model, we compared the following scenarios of DEKR settings: 1) using only the side information provided in the machine learning knowledge graph on the method end; 2) using the graph structure information from the knowledge graph on both the dataset and method ends, without using the description information; 3) not using the graph structure information from the knowledge graph on the dataset and method ends, but using the textual description information; and 4) using both graph structure information and text description information in the knowledge graph on the dataset and method ends.

The experimental results are presented in Table 3 and Figure 7. From Table 3, we can see that the performance without using either the structural or descriptive information of the knowledge graph is inferior to using both of them. Moreover, using side information

Table 4: Effect of Combination Methods.

Combination Methods	AUC	Accuracy	F1-score
sum + dot product	0.9210	0.8225	0.8326
concat + dot product	0.9325	0.8304	0.8467
sum + MLP	0.9011	0.8253	0.8319
concat + MLP	0.9435	0.8762	0.8731
Ours	**0.9682**	**0.9071**	**0.9109**

in the knowledge graph on one end only is weaker than using the information at both ends, indicating that introducing more side information will provide more characterization of the features. Additionally, we can observe from Figure 7 that, in the top-K task, using only structural information outperforms the condition using only descriptive information, showing that the higher-order connections in the graph structure have more inference advantages than using text alone when faced with a larger number of candidates.

5.4.2 ***Effect of Combination Methods.*** In order to explore the effect of different methods for combining two types of information on model performance, we set different types of combination methods for comparison. Specifically, we compared five settings: 1)summing the graph structure representation and descriptive text representation, then performing dot product to calculate the interaction probability; 2) concatenating the two representations, then performing dot product to calculate the interaction probability; 3) summing the two representations, then outputting the prediction probability by the multilayer perceptron; 4) concatenating the two representations, then outputting the prediction probability by the multilayer perceptron; 5) performing dot product for the graph structure representation and MLP for the descriptive text representation then output the probability by combining both(marked by Ours).

The experimental results are summarized in Table 4, which reveals that our method is optimal in comparison. Moreover, The summed settings perform weaker than the concatenated settings. One possible reason is that the two feature representations belong to different semantic spaces due to their heterogeneity.

5.5 Case Study (Q3)

Leveraging on the rich side information in the MLKG, we are able to recommend appropriate ML methods for given ML datasets. However, due to the limitation of information collection or the fact that the entities are newly emerging, a considerable number of entities have relatively few links in the knowledge graph, and the interaction behavior cannot be well predicted solely relying on the graph structure information.

To visualize the role of description text, we randomly selected a dataset d_{1683} and one of its candidate positive ML method samples m_{5173} in the test set. As shown in Figure 8, d_{1683} has 2nd-order neighbors in the knowledge graph while m_{5173} has only one connected entity. The left part shows that a prediction based only on graph structure results in a low interaction probability between the two. Thus, m_{5173} is not recommended to d_{1683}. The right part shows a high interaction probability between the two with reference to the

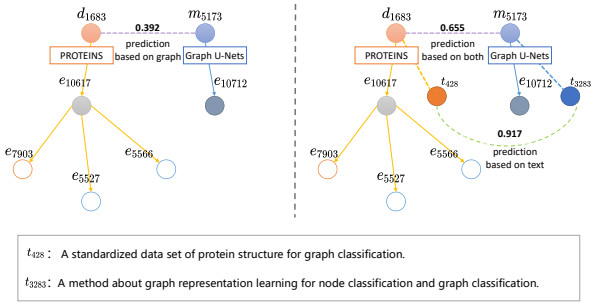

Figure 8: A real-world example from the dataset visualizing how the introduction of descriptive information can contribute to the performance of the recommendation system.

deep collaborative filtering after introducing the descriptive text information. Combining the two predictions, the method m_{5173} is ultimately recommended to the dataset d_{1683}. In addition to improving accuracy, this is also a reasonable explanation for the fact that the description text can lift the recall of the recommender system.

6 CONCLUSIONS & FUTURE WORK

In this paper, we propose DEKR, a machine learning method recommendation system based on a description-enhanced knowledge graph, which can recommend appropriate methods for a given machine learning dataset. We construct a one-of-its-kind description-enhanced machine learning knowledge graph, defining the major entities and relations in the graph, and adding description information for two core entities (i.e. dataset and method) associated with the recommendation task. DEKR has two key components: 1) a knowledge graph neural network, which starts from both dataset and the method ends, propagates over the knowledge graph, extends to higher-order neighbors, then enrich their own representation; and 2) a deep collaborative filtering network, which captures the linear and nonlinear interactions of text features. By combining the two sources of information, DEKR is capable of making highly accurate and efficient recommendations. Extensive experiments on a unique real-world dataset demonstrate the effectiveness of our model.

This work explores the main entities in the machine learning domain and the rich connections between them. In subsequent research, we plan to mine more entities as well as relationships in the knowledge graph. In addition, we will mine more description information to add on to entities other than the two core entities studied in this paper.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Plan of China (No.2018YFB1003804); the National Research Foundation, Singapore under its the AI Singapore Programme (AISG2-RP-2020-019); the RIE 2020 Advanced Manufacturing and Engineering Programmatic Fund (A20G8b0102), Singapore; and the Nanyang Assistant Professorships (NAP).

REFERENCES

[1] N. Asabere, Feng Xia, Qinxue Meng, F. Li, and H. Liu. 2015. Scholarly paper recommendation based on social awareness and folksonomy. *International Journal of Parallel, Emergent and Distributed Systems* 30 (2015), 211 – 232.

[2] N. Asabere, Feng Xia, W. Wang, J. Rodrigues, F. Basso, and J. Ma. 2014. Improving Smart Conference Participation Through Socially Aware Recommendation. *IEEE Transactions on Human-Machine Systems* 44 (2014), 689–700.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, J. Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.

[4] M. A. Brandão, M. Moro, G. R. Lopes, and J. Oliveira. 2013. Using link semantics to recommend collaborations in academic social networks. In *WWW '13 Companion*.

[5] C. Chen, Fidelia Ibekwe-Sanjuan, and Jianhua Hou. 2010. The structure and dynamics of cocitation clusters: A multiple-perspective cocitation analysis. *J. Assoc. Inf. Sci. Technol.* 61 (2010), 1386–1409.

[6] Y. Ding. 2011. Applying weighted PageRank to author citation networks. *J. Assoc. Inf. Sci. Technol.* 62 (2011), 236–245.

[7] Li Dong, Furu Wei, M. Zhou, and K. Xu. 2015. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *ACL*.

[8] X. He, Lizi Liao, Hanwang Zhang, L. Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web* (2017).

[9] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Y. Sun, N. Mamoulis, and Xiang Li. 2016. Meta Structure: Computing Relevance in Large Heterogeneous Information Networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[10] Jyun-Yu Jiang, Pu-Jen Cheng, and W. Wang. 2017. Open Source Repository Recommendation in Social Coding. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017).

[11] Xiangjie Kong, Huizhen Jiang, Z. Yang, Z. Xu, Feng Xia, and A. Tolba. 2016. Exploiting Publication Contents and Collaboration Networks for Collaborator Recommendation. *PLoS ONE* 11 (2016).

[12] Yankai Lin, Zhiyuan Liu, and M. Sun. 2016. Knowledge Representation Learning with Entities, Attributes and Relations. In *IJCAI*.

[13] Xiaozhong Liu, Yingying Yu, C. Guo, and Y. Sun. 2014. Meta-Path-Based Ranking with Pseudo Relevance Feedback on Heterogeneous Graph for Citation Recommendation. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (2014).

[14] S. Milojevic, C. Sugimoto, E. Yan, and Y. Ding. 2011. The cognitive structure of Library and Information Science: Analysis of article title words. *J. Assoc. Inf. Sci. Technol.* 62 (2011), 1933–1953.

[15] Farideh Osareh, R. Khademi, Mansoor Koohi Rostami, and M. S. Shirazi. 2014. Co-authorship Network Structure Analysis of Iranian Researchers' scientific outputs from 1991 to 2013 based on the Social Science Citation Index (SSCI). *Collnet Journal of Scientometrics and Information Management* 8 (2014), 263 – 271.

[16] Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

[17] Xiang Ren, Jialu Liu, X. Yu, Urvashi Khandelwal, Quanquan Gu, L. Wang, and Jiawei Han. 2014. ClusCite: effective citation recommendation by information network-based clustering. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014).

[18] S. Rendle, C. Freudenthaler, Zeno Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. *ArXiv* abs/1205.2618 (2009).

[19] Y. Sun, R. Barber, Manish Gupta, C. Aggarwal, and Jiawei Han. 2011. Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. *2011 International Conference on Advances in Social Networks Analysis and Mining* (2011), 121–128.

[20] Y. Sun, Jiawei Han, X. Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4 (2011), 992–1003.

[21] Chang-You Tai, M. Wu, Yun-Wei Chu, Shao-Yu Chu, and Lun-Wei Ku. 2020. MVIN: Learning Multiview Items for Recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[22] Hongwei Wang, Fuzheng Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018).

[23] Hongwei Wang, Fuzheng Zhang, X. Xie, and M. Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. *Proceedings of the 2018 World Wide Web Conference* (2018).

[24] Hongwei Wang, Fuzheng Zhang, M. Zhao, W. Li, X. Xie, and M. Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. *The World Wide Web Conference* (2019).

[25] Hongwei Wang, M. Zhao, X. Xie, W. Li, and M. Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. *The World Wide Web Conference* (2019).

[26] Jin Wang, Zhongyuan Wang, D. Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *IJCAI*.

[27] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. *CoRR* abs/1905.07854 (2019). arXiv:1905.07854 http://arxiv.org/abs/1905.07854

[28] Xiang Wang, X. He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2019).

[29] Xiang Wang, Dingxian Wang, Canran Xu, X. He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. *ArXiv* abs/1811.04540 (2019).

[30] Feng Xia, N. Asabere, H. Liu, Z. Chen, and W. Wang. 2017. Socially Aware Conference Participant Recommendation With Personality Traits. *IEEE Systems Journal* 11 (2017), 2255–2266.

[31] Feng Xia, H. Liu, I. Lee, and L. Cao. 2016. Scientific Article Recommendation: Exploiting Common Author Relations and Historical Preferences. *IEEE Transactions on Big Data* 2 (2016), 101–112.

[32] Han Xiao, Minlie Huang, Lian Meng, and X. Zhu. 2017. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions. *ArXiv* abs/1604.04835 (2017).

[33] Ruobing Xie, Z. Liu, Huanbo Luan, and M. Sun. 2017. Image-embodied Knowledge Representation Learning. In *IJCAI*.

[34] E. Yan and Y. Ding. 2010. Weighted citation: An indicator of an article's prestige. *ArXiv* abs/1012.4876 (2010).

[35] E. Yan and Y. Ding. 2012. Scholarly network similarities: How bibliographic coupling networks, citation networks, cocitation networks, topical networks, coauthorship networks, and coword networks relate to each other. *J. Assoc. Inf. Sci. Technol.* 63 (2012), 1313–1326.

[36] Rujing Yao, Lin lin Hou, Yingchun Ye, Ou Wu, J. Zhang, and Jian Wu. 2019. Method and Dataset Mining in Scientific Papers. *2019 IEEE International Conference on Big Data (Big Data)* (2019), 6260–6262.

[37] X. Yu, X. Ren, Quanquan Gu, Y. Sun, and Jiawei Han. 2013. Collaborative Filtering with Entity Similarity Regularization in Heterogeneous Information Networks.

[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, X. Xie, and W. Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[39] Star X. Zhao and F. Ye. 2013. Power-law link strength distribution in paper cocitation networks. *J. Assoc. Inf. Sci. Technol.* 64 (2013), 1480–1489.