Rec-RN: User Representations Learning over the Knowledge Graph for Recommendation Systems

Huan Zhao
College of Information Engineer
Northwest Agriculture and Forest University
Shaanxi, China
zhao_huan@nwafu.edu.cn

***Abstract*—As the main technique used in the recommendation system, Collaborative Filtering usually suffers from limited performance due to the high sparsity of user-item interactions. A Knowledge Graph (KG), thus, is introduced to mitigate the problem above. Among the already developed methods in the area of KG, a useful one is called RippleNet. It intends to represent the user embedding with the user history items and it selects key items according to the candidate item to avoid noise. However, it ignores the importance of relationships based on the user when selecting key items. For example, if key items are under the actor relationship, the user may be more like it. If under the director relationship, the user may less like it.**

This paper proposes a KG-based model Rec-RN, an end-to-end framework, for recommendation systems. This method represents the user with the entities in the preference set for mitigating the sparse problems. The preference set consists of knowledge triples that are k hop(s) away from the user's historical items. And then it considers the importance of relationships based on the user for selecting key items. Meanwhile, it considers the relevance between entities in the preference set and the candidate item. Finally, we form high-quality user embedding for predicting the clicking probability. Three commonly used datasets, namely MovieLens-1M, Book-Crossing, and Last.FM, are evaluated. The results show that our Rec-RN outperforms the classic algorithm (e.g., RippleNet) in the click-through rate prediction, with accuracy improvements of 2.2%, 3.1%, and 3.5% in the movie, book, and music fields, respectively.

***Index Terms*—recommendation systems, knowledge graph, user representations.**

I. Introduction

With the widespread Internet, users usually find it hard to decide which they prefer from much information and commodities. To this end, recommendation systems are developed. Currently, collaborative filtering (CF) is one of the most popular recommendation methods to provide users with the highest possible items according to their preferences. A user-item rating matrix reveals the relationships between users and items in CF. However, the rating matrix is often sparse and low-ranked[1], e.g., users only tend to a small fraction of abundant items. Meanwhile, less valuable information in the sparse matrix is being utilized, resulting in unsatisfactory performance. To deal with the issue above, some researchers introduced side information to CF in many domains, such as social networks, purchase history, browsing history, and search patterns.

A Knowledge Graph (KG) is a directed heterogeneous graph, where knowledge can be expressed by a factual triple, namely ($head$, $relation$, $tail$) or ($subject$, $predicate$, $object$), regarding the resource description framework[2]. In KG, $head$ and $tail$ are entities (i.e., natural objects and

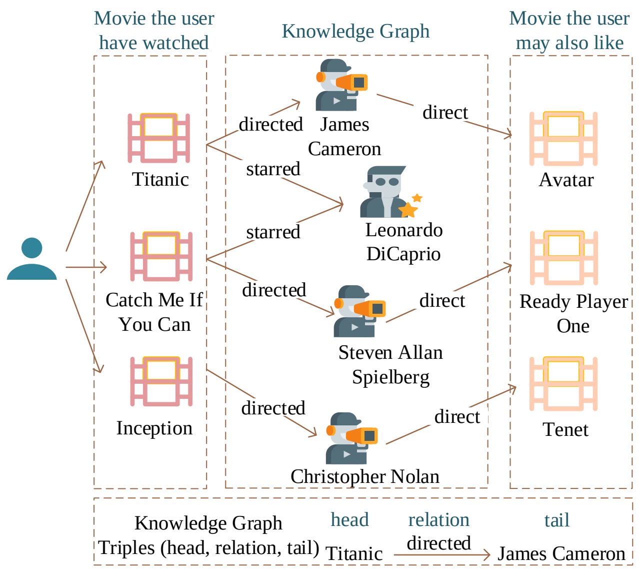

Fig. 1. The knowledge-graph-based movie recommendation systems. We propagate user preferences by relationships in the KG.

abstract concepts), and $relation$ represents the connections between entities. KG visually explains the relationships between a user and an item, e.g., a user has watched the movie "Cast Away" featuring an actor, Tom Hanks. After Tom Hanks appears in another film "Forrest Gump", a potential connection would be available between this user and "Forrest Gump" in the KG, shown in Fig. 1. With the help of KG, recommendation algorithms focus on potential connections between individual users and items and achieve excellent performance.

A large number of KG-aware recommendation algorithms are designed to solve various real-world application problems. These algorithms can be roughly classified into three categories: path-based methods[3–6], embed-based methods[7–9], and hybrid methods[10–12]. Path-based approaches manually design connection patterns between items (i.e., meta-paths or meta-graphs) to extract user-specific relatedness. However, they rely primarily on manually designed meta-paths with predefined domain knowledge. It is hard to uncover and reason on unseen connectivity patterns automatically. The embedding-based approach utilizes the Knowledge Graph Embedding (KGE) methods to regularize the representation of items. To this end, items with similarly connected entities in the KG have similar representations, which are conducive to the collaborative learning of item representation in the recommendation algorithm and KGE task, and extend further user interests and representations. However, the KGE algorithms employed in the recommendation framework are usually more suitable for in-graph applications, such as link prediction[2], rather than recommendation systems.

Recently, hybrid-based techniques combine the benefits of path-based methods and embedding-based methods while avoiding their limits. These methods learn the user and item representations while considering the connection in the KG, e.g., RippleNet[10] intends to represent the user embedding. This method proposes to consider the user history items. And it selects key items from user history items according to the candidate item for avoiding the noise. And then it obtains final user representations for click-through rate prediction. However, this method ignores the importance of relationships based on the user. For example, when selecting the key items in the history items, if key items are under the actor relationship, the user may be more like it. If under the director relationship, the user may less like it. Therefore, it ignores the importance of the relationship resulting in the unsatisfactory performance of the top-k recommendation task.

To address the existing methods' limitations. We propose a model Rec-RN, an end-to-end framework that naturally integrates KG into recommendation systems. To mitigate the sparsity problem in RS, this method represents the user with the entities in the preference set for mitigating the sparse problems. The preference set consists of knowledge triples that are k hop(s) away from the user's historical items. And then this method chooses the key feature in the preference set to represent the user. It considers the relevance between the preference set and candidate item to select key items from user history items. Meanwhile, it also considers the importance of relationships based on the user when selecting key items. Finally, this method concatenates the user and item embedding as the input of Multi-layer Perception (MLP) for predicting the final clicking probability. Our significant contributions are summarized below:

- When selecting key items, we consider the importance of each relationship based on the user. And we are also concerned about the relevance between the preference set and the candidate item. Finally, we form the final high-quality user embedding.
- Three public datasets, including MovieLens-1M, Book-Crossing, and Last.FM, are evaluated for algorithm performance. The results show that our Rec-RN outperforms the classic algorithm (e.g., RippleNet) in the click-through rate prediction, with accuracy improvements of 2.2%, 3.1%, and 3.5% in the movie, book, and music fields, respectively.

II. Background

A knowledge Graph (KG) G is a directed graph whose nodes are entities E and edges denote their relations R. Formally, we define the KG as $G = \{(h, r, t) \mid h, t \in E, r \in R\}$, where each triplet (h, r, t) indicates a fact that there is a relationship r from head entity h to tail entity t, e.g., the triple (Leonardo DiCaprio, film.film.actor, Titanic) states the fact that Leonardo DiCaprio acts in the movie Titanic.

In recommendation systems, let $U = \{u1, u2, ...\}$ and $V = \{v1, v2, ...\}$ denote the sets of users and items respectively. The user-item interaction matrix $Y = \{Y_{uv} \mid u \in U, v \in V\}$ is defined according to users' implicit feedback, where

$$y = \begin{cases} 1, & \text{if interaction (u,v) is observed;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

a value of 1 for y_{uv} indicates an implicit interaction exists between user u and item v, such as behaviors of browsing, watching, clicking, etc. In many recommendation scenarios, an item $v \in V$ association with one or more entities in G, e.g., the movie "Titanic" in V is connected with its entity in the KG.

Our task is formulated as follows. Given interaction matrix Y as well as knowledge graph G, our goal is to predict whether user u has potential interests in candidate item v with which she has had no interaction before. Our goal is to estimate the interaction by:

$$\hat{y}_{uv} = F(u, v; \theta), \tag{2}$$

where $\hat{y}_{uv}$ denotes the probability that user u will click item v and θ denotes the model parameters of function F.

III. Method

In this section, we discuss the proposed Rec-RN in detail and then give some discussions on our model.

A. Framework

The framework of Rec-RN is illustrated in Fig. 2. Rec-RN takes a user u and an item v as input and outputs the prediction probability that user u will click item v. For the input user u, his historical set of interests V_u is treated as seeds in the KG for extending along edges in the KG to form the preference set S_u^k $(k = 1, 2, ..., H)$. A preference set S_u^k is the set of knowledge triples that are k hop(s) away from the seed set V_u. E.g., consider the movie recommendation shown in Fig. 1, where the user watches three movies "Cast Away", "Back to the Future", and "The Green Mile", all of which are referred to by the following: Using those movies as the seeds in the KG, we connect seeds with other items, such as "Adventure", "Tom hanks", etc, we call them relevant items. The relevant items for describing user preferences are accessible since we extend these seeds along different semantic edges in the triple of the KG, these items as the first propagation results add to the preference set. We do it iteratively to collect relevant items during user preferences propagation within different layers to further facilitate embedding the user.

The preference set is firstly used to interact with the candidate item embeddings (the blue blocks) iteratively for strengthening relevant items that are direct connections with the candidate item. And then we compute the importance of each relationship based on the user, which is naturally added to the design of weight. These weights are combined with the corresponding relevant items for the responses of user u (the purple blocks) concerning item v, which are then combined to form the final layer of user embeddings (the grey block). We then concatenate the user embedding with the candidate item embedding as the input of Multi-layer Perception (MLP) for obtaining to compute the prediction probability $\hat{y}_{uv}$.

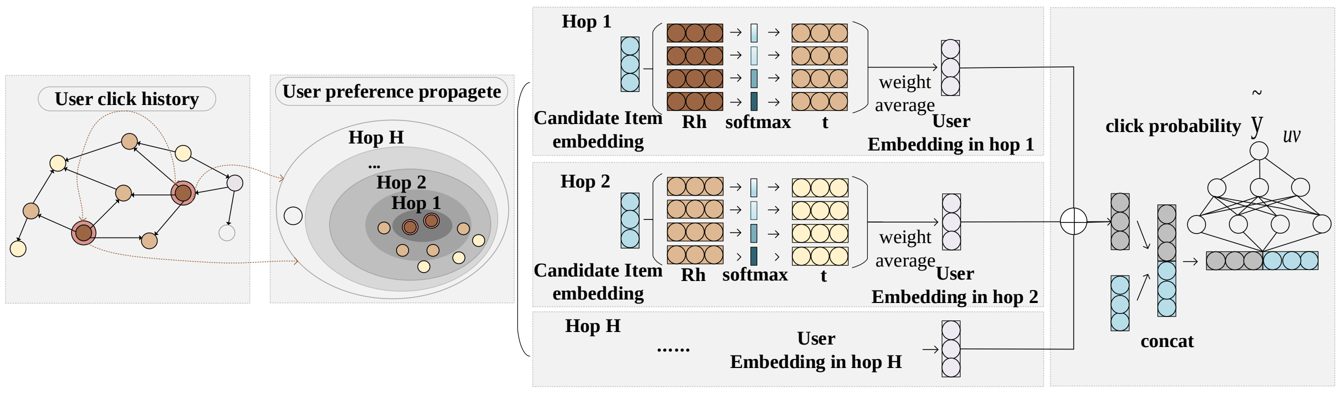

Fig. 2. The overall framework of the Rec-RN.

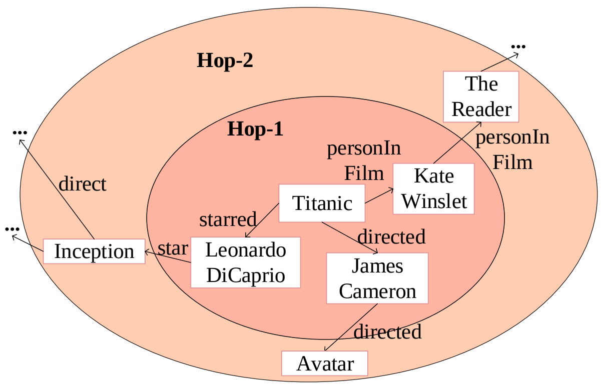

Fig. 3. “Titanic” as seed in the KG to extend user preferences along KG relationships.

B. Preference Set

The preference set is treated to describe user preferences. Rec-RN regards user historical interest or interacted items as seeds set in the KG. Seeds as start points in the KG are extended along different semantic edges to propagate user hierarchy preferences iteratively. We collect relevant items from the above processes and add them to the users' preference sets. Meanwhile, we excavate user latent interest in relevant items in the preference set. As shown in Fig. 3, a user watches the movie “Titanic” which is directed by James Cameron, he directs another movie “Avatar”. Besides, the actor Leonardo DiCaprio participates in the movie “Titanic” and also participates in another movie “Inception”, thus the user may like the movie “Inception” because she prefers the actor Leonardo DiCaprio, and also may like the movie “Avatar” because she prefers director James Cameron.

To conveniently describe users' hierarchically propagated preferences in the KG. For the input user u, her historical set of interests V_u is treated as seeds in the KG, the relevant entity set of user u is defined by:

$$E_u^k = \{t \mid (h,r,t) \in G \cup h \in E_u^{k-1}\},\ k = 1,2,\dots,H, \quad (3)$$

where $E_u^0 = V_u = \{v \mid y_{uv} = 1\}$ represents user u interacted items set. We regard this set as seeds set for user u in the KG. By extending user preferences along edges in the KG iteratively, we obtain the entire preference set for user u in the KG:

$$E_u = \{E_u^0 \vee E_u^1 \vee E_u^2 \vee \cdots \vee E_u^h\}, \quad (4)$$

where h is treated as the number of preferences propagation layers.

The seed set in the KG is then extended along edges to form the preference set S_u^k $(k = 1, 2, ..., H)$, where S_u^k is the set of knowledge triples that are k hop(s) away from the seed set V_u. According to the relevant entity definition, we define the k hop preference set for user u as:

$$S_u^k = \{(h,r,t) \in G \vee h \in E_u^{k-1}\},\ k = 1,2,\dots,H, \quad (5)$$

where the k hop preference set S_u^k regards the seeds set as the start points in the KG.

C. Embedding User Preferences

In this process, we intend to design the weight to strengthen relevant items that are direct connections with the candidate item. Meanwhile, we intend to weaken relevant items that are indirect connections with the candidate item. Besides, we intend to obtain the importance of relevant entities compared to the user and integrate it into the design of the weight. And we combine the weight with corresponding relevant items to form the final user embedding.

Weight Factor. We intend to obtain the weights among relevant entities in the preference set. Following[13], we use a three-way tensor factorization method to define whether the relevant items are triple in the KG with the candidate item. If they are triple in the KG the weight factor is low and high otherwise. Since the importance of each relationship based on the user influences relevant items, e.g., user u may highly like the movie “Back to the Future” when considering its genre but have less in common if measured by the writer. Thus we count the number of relationships b with different types in the preference set for obtaining the importance of each relationship based on the user. And then we naturally integrate it into the weight factor. Thus the weight factor is computed by:

$$\begin{aligned} a_i &= softmax(v^T R_i h_i + b_i) \\ &= \frac{exp(v^T R_i h_i + b_i)}{\sum_{(h,r,t)\in S_u^1} \sum_{b\in S_u^1} exp(v^T Rh + b_i)}, \end{aligned} \quad (6)$$

where $v \in \mathbb{R}^d$, $R_i \in \mathbb{R}^{d\times d}$,$h_i \in \mathbb{R}^d$ are the representation of candidate item v, relationship r_i, and head entity h_i, respectively, i is the size of current layer, and d is the size of dimension. b_i is the number of relationship with type i in S_u^1. Thus we multiply the weights with corresponding relevant entities by:

$$t_i = a_i h_i, \tag{7}$$

where $t_i \in \mathbb{R}^d$ is the embedding of tail t_i. Tail t_i is the result of the multiplication of weights with relevant entities.

We intend to combine t above to obtain the current hierarchical user preferences embeddings and obtain current user preferences embeddings by:

$$o_u^1 = \sum_{(h_i,r_i,t_i)\in S_u^1} a_i W t_i, \tag{8}$$

where $o_u^1 \in \mathbb{R}_u^d$ is seen as the 1 order response of user u's click history V_u with respect to the candidate item v, $W \in \mathbb{R}^{d\times d}$, we obtain the accurate value by later experiment.

We obtain different hierarchical of user preferences embeddings by repeating the process between (6) and (8), thus we obtain the final user preferences embeddings:

$$u = o_u^1 + o_u^2 + \cdots + o_u^H, \tag{9}$$

D. Multi-layer Perception

In this part, we concatenate the user embedding with the candidate item embedding as the input of Multi-layer Perception $\mathcal{G}$ for obtaining to compute the prediction probability:

$$\hat{y}_{uv} = \mathcal{G}(u, v), \tag{10}$$

E. Learning Algorithm

In Rec-RN, our goal is to maximize the following posterior probability of model parameters θ after observing the matrix of implicit feedback Y and the knowledge graph G:

$$max\ p(\theta \mid G, Y), \tag{11}$$

where θ includes the embeddings of all entities, relations in the KG, and items in user-item reaction. This is equivalent to maximizing

$$\begin{aligned} p(\theta \mid G, Y) &= \frac{p(\theta, G, Y)}{p(G, Y)} \\ &\propto p(\theta)\cdot p(G \mid \theta)\cdot p(Y \mid \theta, G), \end{aligned} \tag{12}$$

In (12), the first term $p(\theta)$ measures the priori probability of model parameters θ according to Bayes' theorem. Following[7], we set $p(\theta)$ as Gaussian distribution with zero mean and a diagonal covariance matrix:

$$p(\theta) = N(0, \lambda_1^{-1} I), \tag{13}$$

The second term in (12) is the likelihood function of the observed knowledge graph G given θ, Following[13], we use a three-way tensor factorization method to define the likelihood function for KGE:

$$\begin{aligned} p(G \mid \theta) &= \prod_{(h,r,t)\in E\times R\times E} p((h,r,t) \mid \theta) \\ &= \prod_{(h,r,t)\in E\times R\times E} N(X_{h,r,t} - h^T R t, \lambda_2^{-1}), \end{aligned} \tag{14}$$

where the indicator $X_{h,r,t}$ equals 1 if $(h,r,t) \in G$ and is 0 otherwise. In (14), item-entity pairs in the preference propagation and the scoring functions of entity-entity pairs in the KG are under the same calculation model. The last term in (12) is the likelihood function of the observed implicit feedback given θ and the KG, which is defined as the product of Bernoulli distributions:

$$p(Y \mid \theta, G) = \prod_{(u,v)\in Y} \sigma(u^T v)^{y_{uv}} \cdot (1 - \sigma(u^T v))^{(1-y_{uv})}, \tag{15}$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the sigmoid function. Based on (6) and (9). Thus we have the following loss function for Rec-RN:

$$\begin{aligned} &min\ L = -log(p(Y \mid \theta, G)\cdot p(G \mid \theta)\cdot p(\theta)) \\ &= \sum_{(u,v)Y} -\big(y_{uv} log\sigma(u^T v) + (1 - y_{uv}) log(1 - \sigma(u^T v))\big) \\ &+ \frac{\lambda_2}{2}\sum_{r\in R} ||X_r - E^T R E||_2^2 \\ &+ \frac{\lambda_1}{2}(||V||_2^2 + ||E||_2^2 + \sum_{r\in R} ||R||_2^2), \end{aligned} \tag{16}$$

where V and E are the embeddings matrices for all items and entities, respectively, X_r is the slice of the indicator tensor X in the KG for relation r, and R is the embeddings matrix of relation r. In (16), the first term measures the cross-entropy loss between the ground truth of interactions Y and the predicted value by Rec-RN, and the second term measures the squared error between the ground truth of the KG X_r and the reconstructed indicator matrix $E^T R E$. The third term is the regularizer for preventing over-fitting.

We will demonstrate the efficacy of the model in the experiments section.

F. Discussion

We intend to discuss the depth and width of the propagation. The number of maximal hop H is usually not too large in practice, since entities that are close to a user's history may maximal represent user preference. And entities that are a long distance from a user's history may bring negative signals of noise. We will discuss the choice of H in the experiments part. The number of maximal neighbors is usually not too large in practice. Since we intend to reduce the computation overhead. Thus we sample a fixed-size set of neighbors in each propagation layer. We will discuss the choice of neighbors in the experiments part.

IV. Evaluation

In this section, we evaluate Rec-RN on three public datasets, covering the movie, book, and music fields. We conduct two aspects of the experiment:

- We compare Rec-RN with classical algorithms in two different scenarios, e.g, the click-through rate prediction and the top-K recommendation.
- We experiment to obtain the best width and the depth of the user preference propagation.

TABLE I
BASIC STATISTICES OF THE THREE DATASETS

Dataset	Data Type			
	users	*items*	*interactions*	*KG triple*
MovieLens-1M	6036	2347	753772	20195
Book-Crossing	17860	14910	139746	19793
Last.FM	1872	3846	42346	15518

TABLE II
HYPER-PARAMETER SETTINGS FOR THE THREE DATASETS

Dataset	Hyper-parameter				
	d	H	λ_1	λ_2	η
MovieLens-1M	16	2	10^{-7}	0.01	0.02
Book-Crossing	4	3	10^{-5}	0.01	0.001
Last.FM	4	3	10^{-5}	0.01	0.001

TABLE III
THE RESULTS OF AUC AND ACCURACY IN CTR PREDICTION

Model	MovieLens-1M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
Rec-RN	**0.932**	**0.866**	**0.731**	**0.693**	**0.779**	**0.726**
RippleNet	0.920	0.844	0.729	0.662	0.768	0.691
CKE	0.796	0.739	0.674	0.635	0.744	0.673
DKN	0.655	0.589	0.621	0.598	0.602	0.581

TABLE IV
THE RESULTS OF AUC IN DIFFERENT SIZES OF A USER'S PREFERENCE SET

Dataset	Size of Preference Set					
	8	16	34	64	128	256
MovieLens-1M	0.918	0.921	0.923	**0.932**	0.929	0.931
Book-Crossing	0.713	0.717	0.719	**0.731**	0.718	0.723
Last.FM	0.758	0.760	0.762	**0.779**	0.761	0.772

A. Experiment Setup

a) Dataset: We conduct experiments on MovieLens-1M, Book-Crossing, and Last.FM datasets to evaluate the effectiveness of Rec-RN. All datasets are public and accessible. Table I shows the statistics of the three datasets, where $users$ denotes the total number of users, $items$ denotes the total number of items, $interactions$ denotes the total number of user-item interactions, and $KG\ triple$ denotes the total number of triples.

b) Hyper-parameter Setup: The complete hyper-parameter settings are given in Table II, where d denotes the dimension of embeddings for items and the knowledge graph, and η denotes the learning rate. The hyper-parameters are determined by optimizing AUC on a validation set.

c) Evaluation Metrics: We evaluate our method in two experiment scenarios: (1) In click-through rate (CTR) prediction. Two widely evaluation protocols are used to estimate the effectiveness and the preference result of CTR prediction: AUC and Accuracy. (2) In the top-K recommendation, we use the trained model to select K items with the highest predicted click probability for each user in the test set, and Precision@K, Recall@K, and F1@K are chosen to evaluate the recommended sets.

d) Baselines: To show the effectiveness of our method, we compare the proposed Rec-RN with the following methods:

- CKE [7] integrates structural knowledge, textual knowledge, and visual knowledge into CF in a unified framework for the recommendation.
- DKN [8] treats word embeddings and entity embeddings as multiple channels and combines them in CNN for CTR prediction. It integrates various information to promote the performance of recommendations.
- RippleNet [10] propagates user preferences by extending relevant entities along KG edges iteratively to form final user preferences embeddings. It integrates KG information with items to reduce the parameters.

B. Results

The results of all methods in CTR prediction and top-K recommendation are presented in Table III and Fig. 4, 5, and 6, respectively. Several observations stand out:

- CKE performs pretty poorly than other baselines, which is probably because we only have structural knowledge available, without visual and textual input.
- DKN performs worst in movie, book, and music recommendations. This is because it's hard to extract useful information from movie and book names that are too short and ambiguous.
- RippleNet performs better than CKE and DKN. This means that RippleNet precisely captures user interests.
- In general, our Rec-RN performs best among all methods in the three datasets. Specifically, Rec-RN achieves average Accuracy gains of 14.1%, 5.6%, and 7.4% in movie, book, and music recommendations, respectively. And Rec-RN achieves average AUC gains of 14.2%, 6.1%, and 8.1% in movie, book, and music recommendations, respectively. Rec-RN also achieves better performance in the top-K recommendation as shown in Fig. 4, 5, and 6.

C. Size of Preference Set in Each Hop

We intend to vary the size of the user preference set in each hop to further investigate the robustness of Rec-RN. The AUC results for the three datasets are presented in Table IV, from which we observe that the performance of Rec-RN improves at first as the preference set size increases. Since a larger preference set can increase the probability of the candidate items connecting with these entities. Meanwhile, it can encode more information from KG. And notice that the performance decreases when the size is too large. In general, according to the experimental results, a size of 32 or 64 is sufficient for most datasets.

Hop number. We also vary the maximal hop number H to observe how performance changes in Rec-RN. The results shown in Table V illustrate that the best performance is achieved when H is 2 or 3. Since entities that are close to a user's history may maximally represent user preference, entities that are a long distance from a user's history may bring negative signals of noise.

V. RELATED WORK

Knowledge Graph Embedding. With the Knowledge Graph Embedding (KGE) coming out, researchers now focus

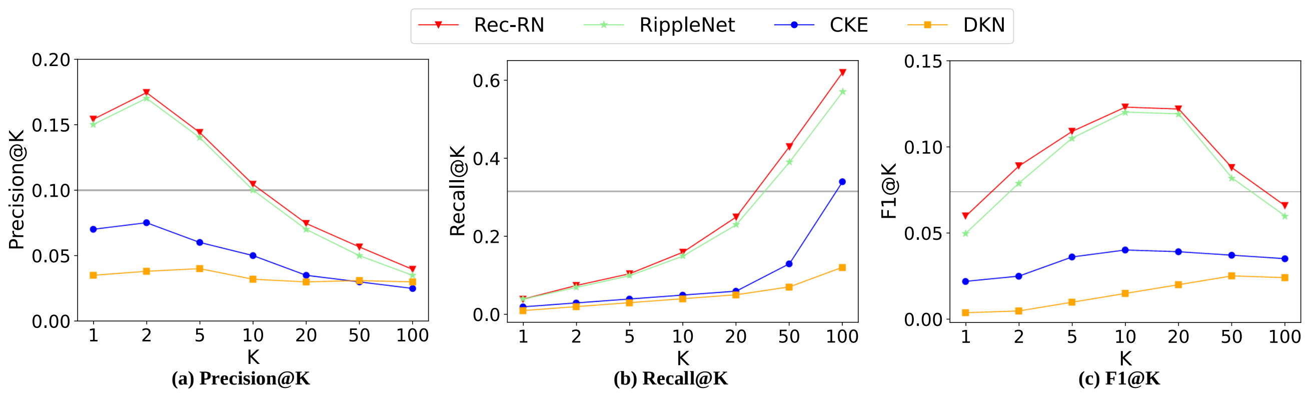

Fig. 4. Precision@K, Recall@K, and F1@K in the top-K recommendation for MovieLens-1M.

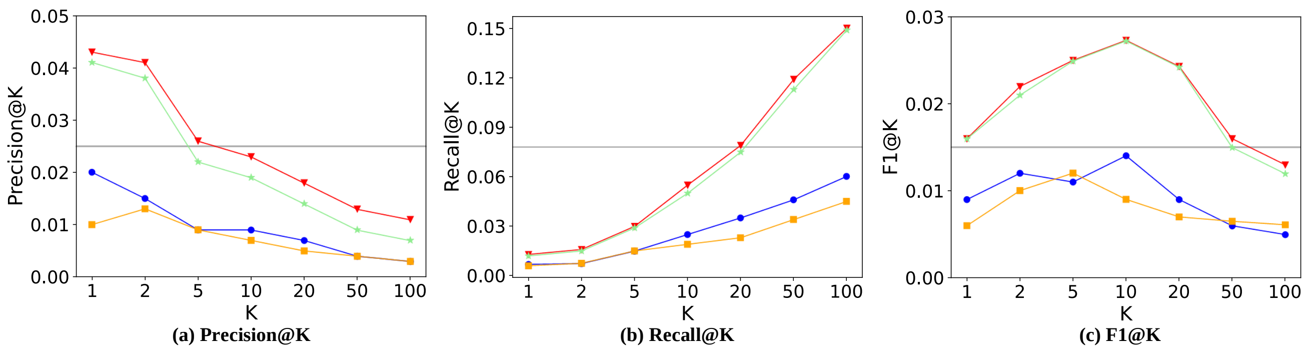

Fig. 5. Precision@K, Recall@K, and F1@K in the top-K recommendation for Book-Crossing.

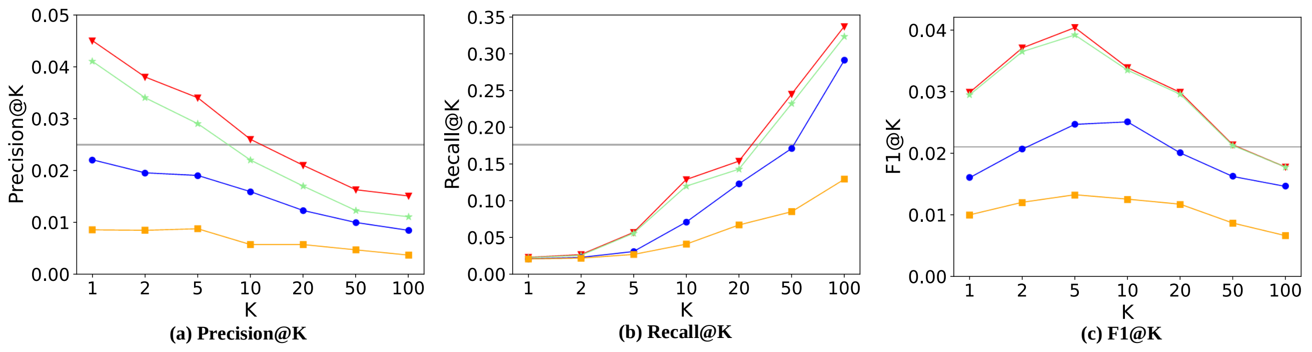

Fig. 6. Precision@K, Recall@K, and F1@K in the top-K recommendation for Last.FM.

TABLE V
THE RESULTS OF AUC IN DIFFERENT HOP NUMBERS

Dataset	**Hop Numbers**			
	1	2	3	4
MovieLens-1M	0.923	**0.931**	0.918	0.928
Book-Crossing	0.725	0.722	**0.736**	0.718
Last.FM	0.762	0.773	**0.779**	0.765

on excavating information in the KG by embedding learning. KGE intends to embed the entities and relations in the KG into a continual representation space, while simultaneously preserving their properties. Many common KGE methods are available, which are mainly classified into two categories: (1) Translational distance models, such as TransE [14], TransH[15], and TransD[16], which design distance-based scoring functions when learning representations of entities and relations, e.g., TransE[14] wants $h + r \approx t$ when (h, r, t) holds, where h, r, and t are the corresponding representation of h, r, and t. Therefore, TransE assumes the score function $f_r(h, t) = || h+r-t ||_2^2$ is low if (h, r, t) holds, and high otherwise. (2) Semantic matching models, such as rescal[13], ANALOGY[17], and ComplEx[18], measure the trustworthiness of a knowledge triples by matching the underlying semantics of entities and relationships, e.g., rescal[13] factor a huge tensor matrix X which consists of entities and relations into three different small matrices, where the head h, relation r, tail t, respectively. We obtain a new tensor matrix by multiplying those three different small matrices and guiding it gradually close to the original tensor matrices X. Therefore, rescal[13] assumes the score function $f(E, R_k) = \frac{1}{2}\left(\sum_k \| X_k - ER_kE^T \|_F^2\right)$ is low if (h, r, t) holds, and high otherwise, where X_k is the value from the original tensor matrix and h, r and t are the corresponding representation in three matrices. KGE is easily combined with RS, thus our approach combines KGE and RS to learn more accurate user and item representations.

VI. CONCLUSION

In this paper, we propose a KG-based model Rec-RN, an end-to-end framework that naturally incorporates the knowledge graph into recommendation systems. Our model achieves the goal that a user may prefer the candidate item when the candidate item is directly connected with one or more interacted items of the user in the KG. We conduct extensive experiments in three recommendation scenarios. The results demonstrate the advantages of Rec-RN in being simple and effective.

REFERENCES

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[3] 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 *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, ser. WSDM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 283–292.

[4] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 635–644.

[5] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1563–1573.

[6] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1531–1540.

[7] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 353–362.

[8] H. Wang, F. Zhang, X. Xie, and M. Guo, "DKN: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 1835–1844.

[9] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," 2019.

[10] 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 *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 417–426.

[11] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, and C. Xu, "Recurrent knowledge graph embedding for effective recommendation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 297–305.

[12] X. Wang, D. Wang, C. Xu, X. He, and T. S. Chua, "Explainable reasoning over knowledge graphs for recommendation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5329–5336, 2019.

[13] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *ICML*, 2011, pp. 809–816. [Online]. Available: https://icml.cc/2011/papers/438_icmlpaper.pdf

[14] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 2787–2795.

[15] J. Zhang, "Knowledge graph embedding by translating on hyperplanes," *AAAI - Association for the Advancement of Artificial Intelligence*, 2014.

[16] G. Ji, S. He, L. Xu, L. Kang, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Meeting of the Association for Computational Linguistics the International Joint Conference on Natural Language Processing*, 2015.

[17] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 1955–1961.

[18] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, 2016, p. 2071–2080.