# KPG4Rec: Knowledge Property-Aware Graph for Recommender Systems

Hao Ge[1], Qianmu Li[1,2(✉)], Shunmei Meng[1], and Jun Hou[3]

[1] Nanjing University of Science and Technology, Nanjing, China
{2429642242,qianmu,mengshunmei}@njust.edu.cn
[2] Intelligent Manufacturing Department, Wuyi University, Nanping, China
[3] School of Social Science, Nanjing Vocational University of Industry Technology, Nanjing, China

**Abstract.** The collaborative filtering (CF) based models have the powerful ability to use the interaction of users and items for recommendation. However, many existing CF-based approaches can only grasp the single relationship between users or items, such as item-based CF, which utilizes the single relationship of similarity identified from user-item matrix to compute recommendations. To overcome these shortcomings, we propose a novel approach named KPG4Rec which integrates multiple property relationships of items for personalized recommendation. In the initial step, we extract properties and corresponding triples of items from an existing knowledge graph, and utilize them to construct property-aware graphs based on user-item interaction graphs. Then, continuous low-dimensional vectors are learned through node2vec technology in these graphs. In the prediction phase, the recommendation score of one candidate item is computed by comparing it with each item in the user history preference sequence, where the pretrained embedding vectors of items are used to take all the properties into consideration. On the other hand, Locality Sensitive Hashing (LSH) mechanism is adopted to generate brand new preference sequences of users to improve the efficiency of KPG4Rec. Through extensive experiments on two real-world datasets, our approach is proved to outperform several widely adopted methods in the Top-N recommendation scenario.

**Keywords:** Knowledge graph · Property-aware graph · Semantic information · Recommendation system

## 1 Introduction

Existing methods for recommender systems (RS) can be divided into two categories in general: collaborative filtering and content-based model. CF-based methods [1–3] find items that users may like through their historical behavior and then filter out items that is not worth being recommended. According to the metadata of the item or content, content-based methods [4, 5] recommend similar items to a user based on the correlation of the items. In recent decades, many different methods have been derived based on the idea of


M. R. Khosravi et al. (Eds.): CloudComp 2021, LNICST 430, pp. 107–122, 2022.
https://doi.org/10.1007/978-3-030-99191-3_9

CF. Sarwar et al. [6] proposed Item-based CF to calculate the similarity between items by counting the co-occurrence times, and recommends the most similar items to users. Matrix factorization [7] characterizes both items and users with vectors of factors for recommendation. BPRMF [8] is directly optimized for ranking and NMF [9] is designed to solve CF problems subject to the constraint of non-negativity.

To deal with the limitations in traditional CF-based methods, researchers have proposed incorporating side information into RS. During many kinds of side information, knowledge graph (KG) contains the most abundant semantic information. Over the past years, several typical KGs have been constructed such as DBpedia, YAGO and Satori which aim to describe all kinds of items or concepts and their relationships in the real world. Thus, making good use of KG has beneficial effects on recommendation results. Personalized Entity Recommendation (PER) [10] combines the heterogeneous relationship information of each user differently to provide high-quality recommendation results. FMG [11] integrates meta-graph into the HIN-based recommendation system.

Inspired by the widely use of KG and its abundant semantic information, we explore an approach to calculate the property similarities between items. In order to fully take advantage of the semantic information, we restructure the user-item bipartite graph by adding property edges between items. So, not only the semantic information is contained in our graphs, collaborative signals of user-item interaction are also considered. Then, node representations are gain by simulating random walks on these graphs. Like the method in [12], the LSH technique is also adopted for the real-time optimization. At last, we compute the final recommended score through aggregating semantic similarities. Experiments are conducted on two real-world recommendation datasets and the results show the efficacy of our model.

## 2 Preliminary and Overview

### 2.1 Knowledge Graph (KG)

A classical KG is a kind of directed heterogeneous graph like Fig. 1. It consists of massive triples and these triples are represented as $(h, r, t)$. Here $h$, $r$, and $t$ denote the head entity, relation and tail entity respectively.

### 2.2 Node2vec Mechanism

Node2vec [13] generates sequences of nodes by simulating random walks on a given graph, and then these sequences are fed into a neural language model to learn the embedding vectors. The probability of walking from node $v$ to $x$ is:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} \text{ if } (v, x) \in E \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

where $\pi_{vx}$ is the unnormalized transition probability between node $v$ and $x$, $Z$ is the normalizing constant. With the introduced parameters, $\pi_{vx}$ can be computed through

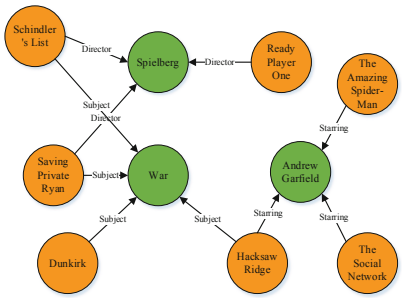



**Fig. 1.** A simple illustration of KG. The orange nodes represent head entities and the green nodes represent tail entities. Relations such as subject and director are formed as directed edges. (Color figure online)

$\pi_{vx} = \alpha_{pq}(v, x) \cdot w_{vx}$, $w_{vx}$ is the weight of edge $(v, x)$ and $\alpha_{pq}$ is defined as:

$$\alpha_{pq}(v, x) = \begin{cases} \frac{1}{p} \text{ if } d_{vx} = 0 \\ 1 \text{ if } d_{vx} = 1 \\ \frac{1}{q} \text{ if } d_{vx} = 2 \end{cases} \tag{2}$$

where $d_{vx}$ denotes the shortest path distance between node $v$ and node $x$. Then, a mapping function $f(v) \rightarrow R^d$ is learned for a given graph $G$ by optimizing the object function of Node2vec:

$$\max_f \sum_{v \in G} \left[ -\log Z_v + \sum_{n_i \in N_s(v)} f(n_i) \cdot f(v) \right] \tag{3}$$

where $v$ refers to one node, $R$ represents the embedding of $v$, $d$ is the dimension. $Z_v = \sum_{e \in G} \exp(f(v) \cdot f(e))$ is the per-node partition function which is approximated by negative sampling. $N_s(v)$ is the neighbors of $v$ defined by random walks.

### 2.3 Locality Sensitive Hashing (LSH)

The LSH mechanism [14] is described as follow: After passing two adjacent points through same hash functions, the possibility that the two points are still adjacent is high. For these, hash functions need to meet the conditions follow:

**Condition 1.** If $d(x, y) \leq d_1$, then the probability of $h(x) = h(y)$ is at least $p_1$;
**Condition 2.** If $d(x, y) \leq d_2$, then the probability of $h(x) = h(y)$ is at most $p_2$;

where $d(x, y)$ represents the distance between data $x$ and $y$, $d_1 < d_2$, $h(x)$ and $h(y)$ represent the hash transformation of $x$ and $y$ respectively. The LSH technique can also be used in the field of security [15]. In recent years, researchers have also incorporated the LSH technology into recommender systems in order to improve the efficiency of models, such as the method proposed in [16].

### 2.4 Problem Formulation

The formulation of the recommendation problem in this paper is given as follows. Like many of others [17, 18], the sets of users and items are denoted by $U = \{u_1, u_2, u_3 ... u_M\}$ and $V = \{v_1, v_2, v_3 ... v_N\}$ respectively. $M$ and $N$ represent the number of users and items. The matrix of user-item interaction is defined as $Y$, where

$$Y_{ij} = \begin{cases} 1 \text{ if user } i \text{ interacted with item } j \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

If user $u_i$ has an implicit interaction with an item $v_j$ before, such as behaviors of browsing or clicking, then $Y_{ij}$ equals to 1, otherwise 0. Given a user $u_i$, we denote his or her history preference sequence $l_i$ as $\{v_1, v_2, v_3 ... v_{N_i}\}$ and $N_i$ is the number of interacted items. In addition to $Y$, a knowledge graph $KG$ is also available. With different properties extracted from $KG$, property-aware graphs $G_p$ of the corresponding property $p$ can be constructed. The goal of our model is to predict whether a user will like an item which he or her has not noticed before. The framework of KPG4Rec is illustrated in Fig. 2. It takes one recommended user $u_i$ and one candidate item $v_c$ as input, and output is the probability score $s$ that $u_i$ like $v_c$.

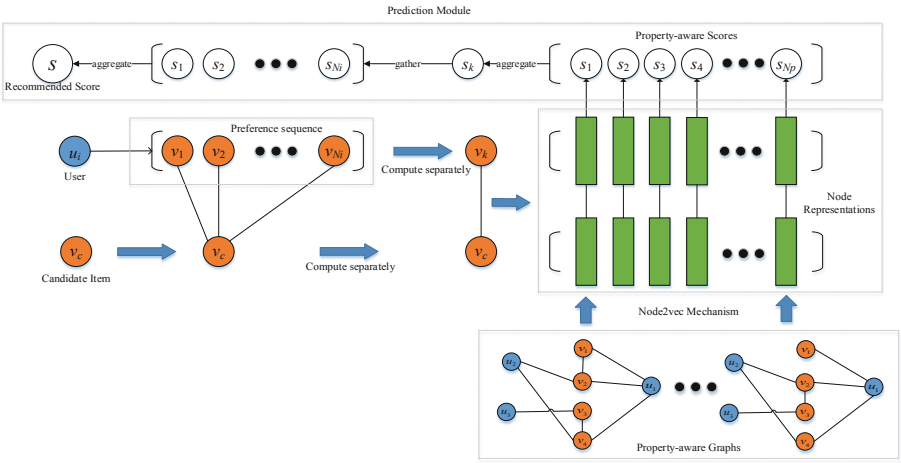



**Fig. 2.** The framework of KPG4Rec.

## 3 Methodology

### 3.1 Construction of Property-Aware Graphs

In order to get the semantic information of an item, we utilize knowledge graph as the side information to assist our recommendation task. KG usually consists of fruitful facts and connections among entities (see Fig. 1). The semantic properties of an entity can be queried through its corresponding (*h, r, t*) triples, where *r* refers to the relationships of properties we need. An item in RS usually has a matched entity in KG, thus the semantic

information of the item is equal to its mapped entity and they can be extracted as the supplementary of our model. What follows in this paper, the proper noun item represents both the item in RS and its mapped entity in KG.

Apart from KG, we can also construct the user-item interaction graph (u-i graph) from the interaction matrix $Y$ (see Fig. 3 left). Note that this type of graph is the representation of user-item interaction information, so there are only explicit collaborative signals contained. In this work, semantic information is integrated into u-i graphs by adding relationships among items. We can extract these relationships from KG by simplifying pairs of triples. Intuitively, two items could be linked up directly in the graph through their common ground (see Fig. 3 right). For instance, two nodes representing movies will be connected if they have the same director. By linking up all items with common ground in KG based on the u-i graph, a brand-new graph named property-aware graph is generated. In KPG4Rec, we construct several different property-aware graphs under corresponding relationships.

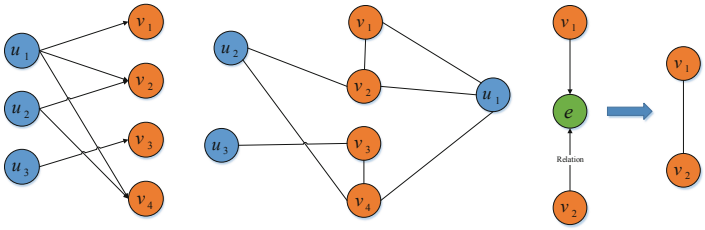



**Fig. 3.** The user-item interaction graph (left) and property-aware graph (mid).

### 3.2 Generation of Property-Aware Vectors

When considering different aspects, items could have different degrees of divergence. So, it is necessary to take all the properties of items into consideration. Aiming to learn different vector representations of each item in all of the property-aware graphs, we adopt the efficient random walk algorithm Node2vec to deal with this task. Node2vec works in our model by simulating random walks on the property-aware graphs. Sequences of items are generated and then fed into a neural language model Word2Vec [19] to learn the vector representations. Then, the proposed approach is that for each property-aware graph $G_p$ generated in Sect. 3.1, we learn a mapping function $f^p(v) \rightarrow R^d$ where $v$ refers to an item, $R$ represents the vector representation of $v$ in $G_p$ and $d$ is the dimension, by optimizing the object function of Node2vec:

$$\max_{f^p} \sum_{v \in G_p} \left[ -\log Z_v + \sum\nolimits_{n_i \in N_s(v)} f^p(n_i) \cdot f^p(v) \right] \tag{5}$$

$Z_v = \sum_{e \in G_p} \exp(f^p(v) \cdot f^p(e))$ is the per-node partition function and Eq. 5 is optimized by SGD. Through the above steps, an item $v_j$ can be expressed by representation of vectors $R_j$ in all the property-aware graphs:

$$R_j = \{R_1, R_2, R_3 \cdots R_{N_p}\} \tag{6}$$

$N_p$ is the number of property-aware graphs and $R_k (k = 1, 2 \ldots N_p)$ is the vector representation of $v_j$ in graph $G_k$. With these vectors, the similarities of items under different properties can be separately and easily computed through vector computed measures.

### 3.3 Regeneration of User Preference Sequence with LSH

In real application scenarios, a user may have purchased a huge amount of items during the accumulated time. The calculation of property similarities between the candidate item and each of all the interacted items would consume too much resources. In consideration of the real time requirements in recommender systems, we adopt the LSH mechanism in two different forms which are termed as Local-LSH and Global-LSH respectively. The basic idea of our LSH mechanism is to hash similar items into a same "bucket" with a high probability of collision. Since a large number of irrelevant items are filtered out, the cost of property similarities calculation is reduced reasonably. Details of Local-LSH and Global-LSH mechanism in KPG4Rec are described as follows.

**Local-LSH.** Given a complete sequence $l_i$ of items liked by the user $u_i$, calculating the semantic similarities of all items in $l_i$ and items in the candidate set will consume a huge amount of time. Note that in these items, many of them have similar properties. So, it is feasible to cluster the sequence $l_i$ before making prediction. We use the first form of LSH called Local-LSH to separate all items in $l_i$ into different buckets, and items with similar properties will be placed in the same bucket. Since using one single hash function for bucketing is prone to lead decreased accuracy, a group of $k$ hash functions are selected randomly and uniformly to ensure similar items more likely to fall into the same bucket. The group of functions is formed as:

$$G = \{f(v) : R^d \rightarrow R^k\} \tag{7}$$

where $v$ is the particular item in $l_i$ and

$$f(v) = (h_1(v), h_2(v) \ldots h_k(v)) \tag{8}$$

where $h(v)$ is a hash function. In this work, we hash all items in $l_i$ through $G$. Each hash function $h(v)$ maps the origin item into two different data spaces and a total of $2^k$ buckets are generated since there are $k$ functions. Each bucket represents a class of items which are similar in some respects. Thus, we select one item from each bucket as a representative, and then a new preference sequence $l_i^{'}$ is obtained by combining representatives in all the buckets.

**Global-LSH.** Note that in the actual recommendation scenario, items with higher similarity to the candidate items often have a greater influence on the user's decision. We adopt Global-LSH to quickly find the neighbors of the candidate item and effectively reduce the time overhead. Similar to Local-LSH, Global-LSH also use a group of functions $G = \{f(v) \colon R^d \rightarrow R^k\}$ to hash the original data. The difference is that in this approach, the candidate item is also hashed by $G$. After this operation, items in each bucket can be seen as the most similar items in the global state. Thus, all the items in

the bucket where the candidate item is located are selected to form a new preference sequence $l_i'$.

In order to improve the efficiency of our model, both the sequence $l_i'$ generated from Local-LSH and Global-LSH can be used to replace the sequence $l_i$ in the prediction module.

### 3.4 Prediction Module

Aiming to recommend a list of items to a target user, each candidate item will get a score calculated in the prediction module. All of the candidate items are ranked then, and items with the highest N scores would be recommended.

In the recommender system, a user could have lots of interacted items. We extract these items of one user as an interaction sequence in order to fully discover the user's preference. Given user $u_i$ and candidate item $v_c$, we use $l_i$ to represent the interaction list of $u_i$. In order to improve the efficiency of our approach, $l_i$ can also be replaced with sequence $l_i'$ which is described in Sect. 3.3, and the performance of different sequences would be discussed in the experimental part. Each item $v_j (j = 1, 2...N_i)$ in $l_i$ and the candidate item $v_c$ are able to be denoted by vector representations. Thus, given the pair of items $v_j$ and $v_c$ in our model, the score $s_p$ which is termed as property-aware score between them under the property $p$ can be calculated by cosine similarity with the vector representations generated from $f^p(v) \rightarrow R^d$:

$$s_p = \frac{\sum_{m=1}^{d} (f_m^p(v_j) \times f_m^p(v_c))}{\sqrt{\sum_{m=1}^{d} (f_m^p(v_j))^2} \times \sqrt{\sum_{m=1}^{d} (f_m^p(v_c))^2}} \tag{9}$$

Note that there are a total of $N_p$ property-aware graphs, the property-aware scores between $v_j$ and $v_c$ can be expressed by the set $S_p$ as follow:

$$S_p = \{s_1, s_2, s_3 \cdots s_{N_p}\} \tag{10}$$

where $s_k (k = 1, 2 \ldots N_p)$ is the property-aware score between $v_j$ and $v_c$ under graph $G_k$. Get the scores of semantic relationships, we aggregate them to get the final similarity score of $v_j$ and $v_c$ which could be calculated by a pooling operation, such as taking the average or the maximum value. In our model, softmax weighted average operation is adopted in order to fully reflect the most similar semantic relationship between items. The final similarity measure score $s_j$ of item $v_j$ and $v_c$ is defined as:

$$s_j = \sum_{s_k \in S_p} \left(\frac{\exp(s_k)}{\sum_{s_m \in S_p} \exp(s_m)} \cdot s_k\right) \tag{11}$$

Extend the situation of one single item $v_j$ to the complete sequence, each item in $l_i$ could get a similarity score by computing with the candidate item. Thus, scores between each item in $l_i$ and the target item can be given in the form of set $S_i$:

$$S_i = \{s_1, s_2, s_3 \ldots s_{N_i}\} \tag{12}$$

Where $N_i$ is the length of $l_i$. Similar to Eq. 11, the recommended score $s$ of a candidate item $v_c$ for the specific user is calculated by:

$$s = \sum_{s_k \in S_i} \left( \frac{\exp(s_k)}{\sum_{s_m \in S_i} \exp(s_m)} \cdot s_k \right) \tag{13}$$

After sorting all candidate items according to the final score $s$, items with N highest scores are recommended to the user $u_i$.

## 4 Experiments

### 4.1 Datasets

In this section, our proposed model KPG4Rec is evaluated on two real-world datasets. The first we utilized is MovieLens-1M which contains 1,000,209 anonymous ratings made by 6,040 users on 3,952 movies. In this experiment, items rated by users are taken as positive feedback to construct user-item interaction graphs. The second dataset is LastFM from the online music system Last.fm. LastFM contains 92,834 implicit feedback of 1,892 users on 17,632 musical artists.

Items in MovieLens-1M and LastFM can be mapped to the corresponding DBpedia entities in a previous work [20]. By using these Linked Open Data, we build a KG and construct property-aware graphs based on u-i graphs. Note that not every item in the datasets can correspond to a DBpedia entity, we kick out items with no matched or multiple matched entities for simplicity. In order to fully reflect the importance of user history preferences, users with no more than ten interactions are also excluded. After the above two operations, the number of items and ratings of MovieLens-1M becomes 3,226 and 948,976 respectively; the number of users and musical artists of LastFM attenuates to 1,865 and 9,765, the number of feedback tags reduce to 78,633 at the same time. For each dataset, we divide training, evaluation and test set according to the ratio of 7:1:2. The three parts are used to represent the user's historical behavior, determine the hyperparameters and evaluate KPG4Rec respectively.

### 4.2 Evaluation Metrics

Evaluation metrics occupies a pivotal position in the recommendation system. Since our recommendation task is a Top-N recommendation based on the click-through rate, methodologies which are based on error metrics (such as RMSE and MAE) are not appropriate for evaluating our model [21]. In this work, we take two classification accuracy measures Precision and Recall, the average of which are defined as follows:

$$\text{Precision@}k = \frac{\sum_{u \in U} \sum_{i=1}^{k} \frac{\text{hit}(u, v_i)}{k}}{|U|} \cdot \text{Recall@}k = \frac{\sum_{u \in U} \sum_{i=1}^{k} \frac{\text{hit}(u, v_i)}{l(u)}}{|U|}$$

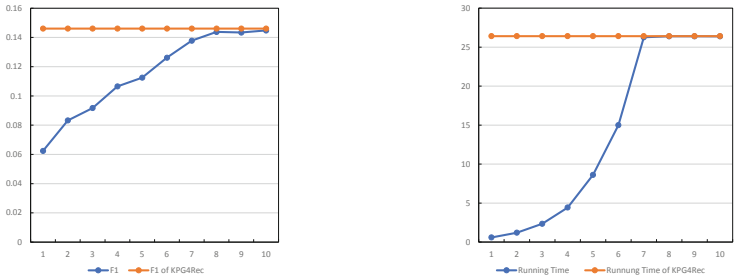
where $k$ is the number of items to be recommended, $\text{hit}(u, v_i) = 1$ if the recommended item $v_i$ is in user $u'$s interaction list of the test set, otherwise 0, and $l(u)$ is the length

of the interaction list. In order to determine the hyperparameters in KPG4Rec, F1 score which is the comprehensive consideration of precision and recall is also adopted:

$$\mathrm{F1@k} = \frac{2 \times \mathrm{Precision@k} \times \mathrm{Recall@k}}{\mathrm{Precision@k} + \mathrm{Recall@k}}$$

### 4.3 Impact of Parameters

Aiming at constructing property-aware graphs, we select the most frequently occurring properties from the DBpedia Ontology and extract all triples from KG for the corresponding property. Then, we look for items which connected to the same entity through SPARQL queries, and link them up directly in u-i graphs under the particular property. In our experiment, hyperparameters in Node2vec are set as follows. Number of walks per entity $n = 50$, length of each walk $l = 100$, the return hyperparameter $p = 1$ and the in-out hyperparameter $q = 1$, the dimension of vector embedding $d = 200$, context size for optimization is 30 and number of epochs in SGD is 5.

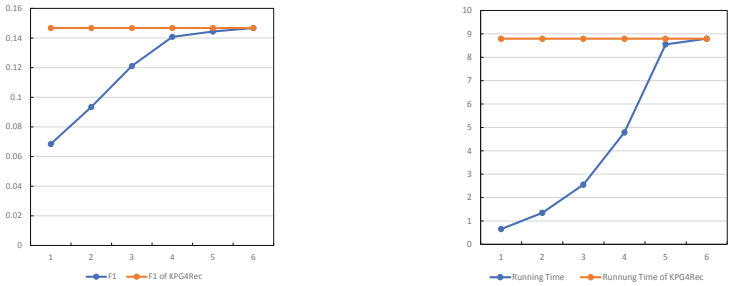



(a) (b)

**Fig. 4.** Effect of parameter $k$ on MovieLens-1M. The value of the x-axis refers to $k$, the orange line refers to the evaluation of KPG4Rec without LSH. (a) shows the effect of our model under different $k$ values and (b) shows the running time.

Another important hyperparameter is the number $k$ of hash functions. In order to select the appropriate $k$ based on the time consumption and F1 value, KPG4Rec with Local-LSH is evaluated on the evaluation set of MovieLens-1M and LastFM. Precision@5 and Recall@5 are used to compute the F1 value and results are shown in Figs. 4 and 5.

It can be clearly found that as the value of $k$ increases, the model effect and running time are increasing simultaneously. Our requirement for selecting $k$ is to minimize the running time while maintaining the original efficiency without too much fluctuation. For the reason that the F1 value only changes less than 0.02 while the running time is reduced to nearly half, we choose 6 as the number of hash functions in Local-LSH for MovieLens-1M. This configuration also means that there is a total of 64 buckets and the length of the user preference sequence is 64. For KPG4Rec with Global-LSH, we choose 4 as the value of $k$ to ensure that the length of the user preference sequence is consistent with that in Local-LSH. Similar to MovieLens-1M, we choose 4 and 1 respectively as the number of Local-LSH and Global-LSH hash functions for the dataset LastFM.




(a) (b)

**Fig. 5.** Effect of parameter $k$ on LastFM. The value of the x-axis refers to $k$, the orange line refers to the evaluation of KPG4Rec without LSH. (a) shows the effect of our model under different $k$ values and (b) shows the running time.

### 4.4 Performance Evaluation

In this subsection, we evaluate our novel model KPG4Rec and compare it against the following baselines:

**BPRMF:** A matrix factorization recommendation algorithm optimized by Bayesian Personalized Ranking.
**ItemKNN:** A classic CF recommendation algorithm based on item similarity.
**NMF:** A Non-negative Matrix Factorization-based approach to perform collaborative filtering and implement recommendations.
**SLIM** [22]**:** A sparse linear method (SLIM) which generates Top-N recommendations through aggregating from user purchase profiles.
**SpectralCF** [23]**:** A deep recommendation model which is able to explore deep connections between users and items.
**TopPop:** A recommendation strategy which recommends top-N items with the highest popularity.

In addition to the baselines mentioned above, we also take KPG4Rec with Local-LSH and Global-LSH into consideration which are termed as **KPG4Rec(L)** and **KPG4Rec(G)** respectively. Figure 6 shows the performance of different methods on MovieLens-1M, and Fig. 7 shows that on LastFM.

It can be seen from the experimental results significantly that our proposed model performs much better than the other methods in both MovieLens-1M and LastFM. In addition, as the number $N$ changes, our model can still maintain the best performance. In the comparison of KPG4Rec and its two variants, KPG4Rec (L) has a slightly worse performance while KPG4Rec (G) performs best of all the three models. The reason is that the buckets generated from the hash function in Local-LSH are not the most accurate

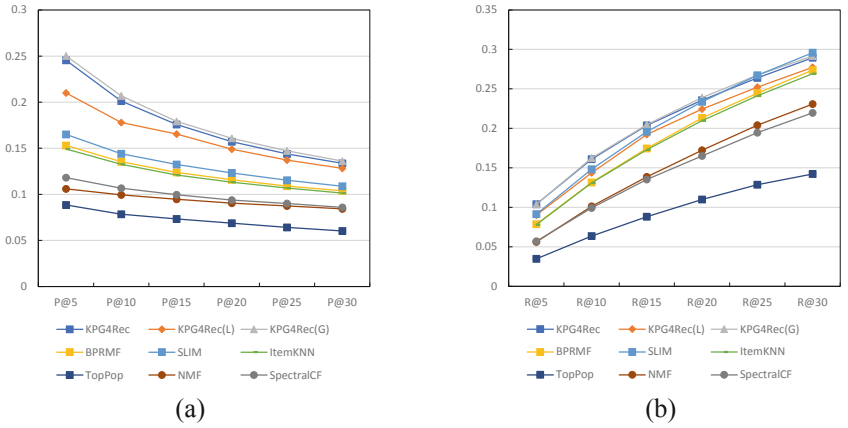



**Fig. 6.** Precision@$N$ (a) and Recall@$N$ (b) of Top-$N$ recommendation for MovieLens-1M.

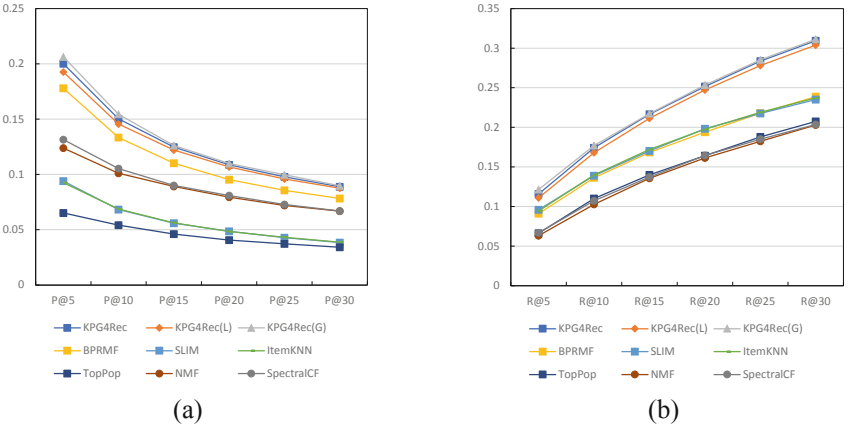



**Fig. 7.** Precision@$N$ (a) and Recall@$N$ (b) of Top-$N$ recommendation for LastFM.

classification of each item in the user's preference sequence. Different from this, the hash functions in Global-LSH map a candidate item and its similar items to a same bucket. The outstanding performance of KPG4Rec with Global-LSH could reflect a fact: items that are similar to the candidate item usually occupy an important position in the user's decision-making consideration.

### 4.5 Evaluation of Different Properties

Since our recommendation is based on the semantic information, the effect of using only one single property-aware graph in KPG4Rec will be discussed in this subsection. In order to fully explore the semantic relevance between items, we select the nine most frequently occurring properties from the DBpedia Ontology to construct property-aware graphs. Aiming at discovering the differences among different semantic properties, one single graph is used in KPG4Rec(G) to make prediction in this part. Results are shown in Table 1 and Table 2. It is not difficult to find that using only one single property-aware graph for recommendation is also very effective. Note that the property-aware graph contains not only the semantic information of items, but also the interactive information between users and items. Thus, the collaborative signals of user-item interaction considered when constructing property-aware graphs play a vital role in our proposed model KPG4Rec.

In the movie recommendation scene, using the graph constructed under the single property of MusicComposer, Subject or Distributor has a higher recommendation significance. This means that when a user chooses to watch a movie, the music, theme, and other factors play a more important determinant. Similar to the movie field, music subjects and genres are more considered by the users which can be inferred from Table 2. By using the semantic information of items, the recommendation results are also interpretable for ordinary users. Give a simple example, a movie is recommended to a user because it has the same subject or the same director as movies the user has watched before.

**Table 1.** Evaluation of using one single property in KPG4Rec(G) on MovieLens-1M.

| Property | P@5 | P@10 | P@15 | R@5 | R@10 | R@15 |
|---|---|---|---|---|---|---|
| Director | 0.2376 | 0.1975 | 0.1719 | 0.0985 | 0.1564 | 0.1959 |
| Starring | 0.2281 | 0.1899 | 0.1662 | 0.0944 | 0.1484 | 0.1886 |
| Distributor | 0.2413 | 0.1963 | 0.1711 | 0.1008 | 0.1557 | 0.1968 |
| Writer | 0.2344 | 0.1939 | 0.1681 | 0.0973 | 0.1537 | 0.1941 |
| MusicComposer | 0.2435 | 0.2002 | 0.1739 | 0.1019 | 0.1584 | 0.1997 |
| Producer | 0.2377 | 0.1961 | 0.1689 | 0.0982 | 0.1544 | 0.1918 |
| Cinematography | 0.2395 | 0.1986 | 0.1733 | 0.1001 | 0.1574 | 0.1995 |
| Editing | 0.2370 | 0.1945 | 0.1697 | 0.0991 | 0.1536 | 0.1949 |
| Subject | 0.2404 | 0.1995 | 0.1735 | 0.1003 | 0.1560 | 0.1961 |
| KPG4Rec(G) | 0.2503 | 0.2068 | 0.1791 | 0.1037 | 0.1625 | 0.2045 |

**Table 2.** Evaluation of using one single property in KPG4Rec(G) on LastFM.

| Property | P@5 | P@10 | P@15 | R@5 | R@10 | R@15 |
|---|---|---|---|---|---|---|
| Genre | 0.1937 | 0.1458 | 0.1221 | 0.1126 | 0.1691 | 0.2129 |
| RecordLabel | 0.1977 | 0.1501 | 0.1242 | 0.1147 | 0.1741 | 0.2162 |
| Hometown | 0.1932 | 0.1456 | 0.1216 | 0.1113 | 0.1679 | 0.2101 |
| AssociatedBand | 0.1890 | 0.1442 | 0.1185 | 0.1099 | 0.1674 | 0.2060 |
| AssociatedMusicalArtist | 0.1867 | 0.1434 | 0.1193 | 0.1084 | 0.1665 | 0.2075 |
| BirthPlace | 0.1913 | 0.1429 | 0.1196 | 0.1103 | 0.1650 | 0.2071 |
| BandMember | 0.1807 | 0.1367 | 0.1145 | 0.1046 | 0.1574 | 0.1981 |
| FormerBandMember | 0.1812 | 0.1373 | 0.1148 | 0.1046 | 0.1578 | 0.1984 |
| Subject | 0.1988 | 0.1511 | 0.1279 | 0.1161 | 0.1766 | 0.2238 |
| KPG4Rec(G) | 0.2064 | 0.1544 | 0.1262 | 0.1214 | 0.1771 | 0.2174 |

## 5 Related Work

### 5.1 Random Walk Algorithm

In recent years, some models in the natural language processing (NLP) field are utilized in random walk algorithms to extract the structural features of graphs. Nodes in graphs are represented by low dimensional continuous vectors which can be used for downstream tasks then. Word2Vec [19] is an efficient NLP model which can capture a large number of precise syntactic and semantic word relationships by learning high-quality distributed vector representations (Skip-gram). DeepWalk [24] aims at learning latent representations of vertices in a network like graphs. These latent representations can be easily exploited by NLP models for the reason that they encode social relations in a continuous vector space. Node2vec [13] adjusts the weight of random walk to control the tendency of BFS or DFS based on DeepWalk. Due to its high performance, Node2vec technique is adopted in KPG4Rec.

### 5.2 Recommendations Using Knowledge Graph

Recently, the usage of the KG in RS is attracting increasing attention and has shown the effectiveness already. For example, DKN [25] is a news recommendation model which utilizes knowledge graph representation and deep networks. KGCN [26] captures relatedness of items by mining their associated semantic relationships in KG, and the model is learned through an end-to-end way. CKAN [27] proposes a natural method of combining collaborative signals which are encoded by collaboration propagation with semantic associations contained in KG together. In this paper, we utilize multiple property relationships of items extracted from KG for personalized recommendation.

## 6 Conclusion

In this paper, we propose a novel model KPG4Rec with the enhancement of a knowledge graph for Top-N recommendation. Our approach aims to make better use of the semantic information of items. Following this destination, we first construct several property-aware graphs which contain both the collaborative signals and the semantic information. Next, a random walk algorithm of graph Node2vec is adopted on these graphs to generate item vector representations under different properties. Then these representations are used to make the final prediction. In addition, we also use the LSH technique to reduce the time overhead and improve the accuracy of our model. Extensive experiments have been conducted on two real-world datasets, and the effectiveness of our KPG4Rec framework is validated. For future work, we plan to (1) apply graph mining methods to our property-aware graphs to better explore user's potential interests; (2) utilize the recently popular deep networks for a more efficient recommendation like [28, 29].

**Acknowledgement.** This work is supported in part by the Fundamental Research Fund for the Central Universities (30920041112, 30919011282), the National Key R&D Program of China (Funding No. 2020YFB1805503), Jiangsu Province Modern Education Technology Research Project (84365); National Vocational Education Teacher Enterprise Practice Base "Integration of Industry and Education" Special Project (Study on Evaluation Standard of Artificial Intelligence Vocational Skilled Level). the Postdoctoral Science Foundation of China (2019M651835), National Natural Science Foundation of China (61702264).

## References

1. Bell, R.M., Koren, Y.: Improved neighborhood-based collaborative filtering. In: KDD Cup Workshop 13th ACM SIGKDD International Conference on Knowledge Discovery, pp. 7–14. ACM, San Jose (2007)
2. Wang, F., Zhu, H., Srivastava, G., Li, S., Khosravi, M.R., Qi, L.: Robust collaborative filtering recommendation with user-item-trust records. IEEE Trans. Comput. Soc. Syst. 1–11 (2021)
3. Li, B., Zhu, X., Li, R., Zhang, C.: Rating knowledge sharing in cross-domain collaborative filtering. IEEE Trans. Cybern. **45**(5), 1054–1068 (2015)
4. Linden, G., Smith, B., York, J.: Amazon. com recommendations: item-to-item collaborative filtering. IEEE. Internet. Comput. **7**(1), 76–80 (2003)
5. Liu, J., Dolan, P., Pedersen, E.R.: Personalized news recommendation based on click behavior. In: ACM 15th International Conference Intelligence User Interfaces, pp. 31–40. ACM, Hong Kong (2010)
6. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation. In: 10th International Conference World Wide Web, pp. 285–295. ACM, Hong Kong (2001)
7. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
8. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: 25th Conference Uncertainity Artificial Intelligence, pp. 452–461. UAI 2009 (2009)
9. Luo, X., Zhou, M., Xia, Y., Zhu, Q.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Ind. Inform. **10**, 1273–1284 (2014)

10. Yu, X., et al.: Personalized entity recommendation: A heterogeneous information network approach. In: WSDM 2014 - Proceedings of 7th ACM International Conference Web Search Data Mining, pp. 283–292. ACM, New York (2014)
11. Zhao, H., Yao, Q., Li, J., Song, Y., Lee, D.L.: Meta-graph based recommendation fusion over heterogeneous information networks. In: ACM SIGKDD International Conference on Knowledge Discovery Data Mining, pp. 635–644. ACM, Halifax (2017)
12. Xu, X., Huang, Q., Zhang, Y., Li, S., Qi, L., Dou, W.: An LSH-based offloading method for IoMT services in integrated cloud-edge environment. ACM Trans. Multimed. Comput. Commun. **16**(3), 1–19 (2021)
13. Grover, A., Leskovec, J.: Node2Vec. In: KDD 2016: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM, San Francisco (2016)
14. Datar, M., Indyk, P., Immorlica, N., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Annual Symposium on Computational Geometry, pp. 253–262. ACM, Brooklyn (2004)
15. Xu, X., et al.: Secure service offloading for internet of vehicles in SDN-enabled mobile edge computing. IEEE Trans. Intell. Transp. Syst. **22**(6), 3720–3729 (2021)
16. Qi, L., Wang, X., Xu, X., Dou, W., Li, S.: Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing. IEEE Trans. Netw. Sci. Eng. **8**, 1145–1153 (2020)
17. Wang, H., et al.: RippleNet: propagating user preferences on the knowledge graph for recommender systems. In: International Conference on Information and Knowledge Management Proceedings, pp. 417–426. ACM, Torino (2018)
18. Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., Guo, M.: Multi-task feature learning for knowledge graph enhanced recommendation. In: Web Conference on 2019 - Proceedings of World Wide Web Conference, pp. 2000–2010. ACM, San Francisco (2019)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26, pp. 3111–3119. NIPS (2013)
20. Ostuni, V.C., Di Noia, T., Mirizzi, R., Di Sciascio, E.: Top-N recommendations from implicit feedback leveraging linked open data. In: CEUR Workshop Proceedings, pp. 20–27. ACM, Hong Kong (2014)
21. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-N recommendation tasks. In: RecSys 2010 – Proceedings of 4th ACM Conference on Recommendation Systems, pp. 39–46. ACM, Barcelona (2010)
22. Ning, X., Karypis, G.: SLIM: sparse linear methods for top-N recommender systems. In: IEEE International Conference Data Mining, ICDM, pp. 497–506. IEEE, Vancouver (2011)
23. Zheng, L., Lu, C.T., Jiang, F., Zhang, J., Yu, P.S.: Spectral collaborative filtering. In: RecSys 2018 - 12th ACM Conference on Recommendation Systems, pp. 311–319. ACM. Vancouver (2018)
24. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: Online learning of social representations. In: ACM SIGKDD International Conference Knowledge Discovery and Data Mining, pp. 701–710. ACM, New York (2014)
25. Wang, H., Zhang, F., Xie, X., Guo, M.: DKN: Deep knowledge-aware network for news recommendation. In: Web Conference on 2018 – Proceedings of World Wide Web Conference WWW 2018, pp. 1835–1844. ACM, Lyon (2018)
26. Wang, H., Zhao, M., Xie, X., Li, W., Guo, M.: Knowledge graph convolutional networks for recommender systems. In: Web Conference on 2019 – Proceedings of World Wide Web Conference WWW 2019, pp. 3307–3313. ACM, San Francisco (2019)

27. Wang, Z., Lin, G., Tan, H., Chen, Q., Liu, X.: CKAN: collaborative knowledge-aware attentive network for recommender systems. In: SIGIR 2020 – Proceedings of 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 219–228. ACM, Virtual Event (2020)
28. Xu, X., et al.: Edge content caching with deep spatiotemporal residual network for IoV in smart city. ACM Trans. Sens. Netw. **17**(3), 1–33 (2021)
29. Liu, Y., et al.: An attention-based category-aware GRU model for the next POI recommendation. Int. J. Intell. Syst. **36**, 3174–3189 (2021)