algorithms

MDPI

Article

Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation

Qingyao Ai [1], Vahid Azizi [2], Xu Chen [3] and Yongfeng Zhang [2,*]

1 College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01002, USA; aiqy@cs.umass.edu
2 Department of Computer Science, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08854, USA; va190@cs.rutgers.edu
3 School of Software, Tsinghua University, Beijing 100084, China; xu-ch14@mails.tsinghua.edu.cn
* Correspondence: yongfeng.zhang@rutgers.edu; Tel.: +1-848-445-2001

Received: 8 August 2018; Accepted: 10 September 2018; Published: 13 September 2018

check for updates

Abstract: Providing model-generated explanations in recommender systems is important to user experience. State-of-the-art recommendation algorithms—especially the collaborative filtering (CF)-based approaches with shallow or deep models—usually work with various unstructured information sources for recommendation, such as textual reviews, visual images, and various implicit or explicit feedbacks. Though structured knowledge bases were considered in content-based approaches, they have been largely ignored recently due to the availability of vast amounts of data and the learning power of many complex models. However, structured knowledge bases exhibit unique advantages in personalized recommendation systems. When the explicit knowledge about users and items is considered for recommendation, the system could provide highly customized recommendations based on users' historical behaviors and the knowledge is helpful for providing informed explanations regarding the recommended items. A great challenge for using knowledge bases for recommendation is how to integrate large-scale structured and unstructured data, while taking advantage of collaborative filtering for highly accurate performance. Recent achievements in knowledge-base embedding (KBE) sheds light on this problem, which makes it possible to learn user and item representations while preserving the structure of their relationship with external knowledge for explanation. In this work, we propose to explain knowledge-base embeddings for explainable recommendation. Specifically, we propose a knowledge-base representation learning framework to embed heterogeneous entities for recommendation, and based on the embedded knowledge base, a soft matching algorithm is proposed to generate personalized explanations for the recommended items. Experimental results on real-world e-commerce datasets verified the superior recommendation performance and the explainability power of our approach compared with state-of-the-art baselines.

Keywords: recommender systems; explainable recommendation; knowledge-base embedding; collaborative filtering

1. Introduction

Most of the existing collaborative filtering (CF)-based recommendation systems work with various unstructured data such as ratings, reviews, or images to profile the users for personalized recommendation. Though effective, it is difficult for existing approaches to model the explicit relationship between different information that we know about the users and items. In this paper, we would like to ask a key question, i.e., *"can we extend the power of CF upon large-scale structured user behaviors?"*. The main challenge to answer this question is how to effectively integrate different types of user behaviors and item properties, while preserving the internal relationship between them to enhance the final performance of personalized recommendation.

Fortunately, the emerging success on knowledge-base embeddings (KBE) may shed some light on this problem, where heterogeneous information can be projected into a unified low-dimensional embedding space. By encoding the rich information from multi-type user behaviors and item properties into the final user/item embeddings, we can enhance the recommendation performance while preserving the internal structure of the knowledge.

Equipping recommender systems with structured knowledge also helps the system to generate informed explanations for the recommended items. Researchers have shown that providing personalized explanations for the recommendations helps to improve the persuasiveness, efficiency, effectiveness, and transparency of recommender systems [1–7]. By preserving the knowledge structure about users, items, and heterogenous entities, we can conduct fuzzy reasoning over the knowledge-base embeddings (KBE) to generate tailored explanations for each user.

Inspired by the above motivations, in this paper, we design a novel explainable CF framework over knowledge graphs. The main building block is an integration of traditional CF with the learning of knowledge-base embeddings. More specifically, we first define the concept of user-item knowledge graph, which encodes our knowledge about the user behaviors and item properties as a relational graph structure. The user-item knowledge graph focuses on how to depict different types of user behaviors and item properties over heterogeneous entities in a unified framework. Then, we extend the design philosophy of CF to learn over the knowledge graph for personalized recommendation. For each recommended item, we further conduct fuzzy reasoning over the paths in the knowledge graph based on soft matching to construct personalized explanations.

Contributions. The main contributions of this paper can be summarized as follows:

- We propose to integrate heterogeneous multi-type user behaviors and knowledge of the items into a unified graph structure for recommendation.
- Based on the user-item knowledge graph, we extend traditional CF to learn over the heterogeneous knowledge for recommendation, which helps to capture the user preferences more comprehensively.
- We further propose a soft matching algorithm to construct explanations regarding the recommended items by searching over the paths in the graph embedding space.

In the following part of the paper, we first present related work in Section 2, and then provide the problem formalization in Section 3. Section 4 goes over the model for CF over knowledge graphs, and in Section 5 the soft matching method for generating explanations is illustrated. Experimental setup and discussion of the results are provided in Section 6 and Section 7, respectively. We conclude the work and point out some of the future research directions in Section 8.

2. Related Work

Using knowledge base to enhance the performance of recommender system is an intuitive idea, which has attracted research attention since the very early stage of the recommender system community. For example, Burke [8] and Trewin [9] discussed the strengths and weaknesses of both knowledge-based and CF-based recommender systems, and introduced the possibility of a hybrid recommender system that combines the two approaches. Ghani and Fano [10] presented a case study of a system that recommends items based on a custom-built knowledge base that consists of products and associated semantic attributes. Heitmann [11] proposed to conduct cross-domain personalization and recommendation based on multi-source semantic interest graphs.

More recently, Musto et al. [12] and Noia et al. [13] conducted semantic graph-based recommendation leveraging linked open data as external knowledge, Oramas et al. [14] adopted knowledge graphs to produce sound and music recommendations, and Catherine et al. [15] proposed to jointly rank items and knowledge graph entities using a personalized page-rank procedure to produce recommendations together with the explanations.

Though intuitive, the difficulty of reasoning over the hard-coded paths on heterogeneous knowledge graph prevents existing approaches from leveraging CF on very different entities and relations, which further makes it difficult to take advantage of the wisdom of crowd.

Fortunately, recent advances on KBE shed light on this problem. In KBE, entities and relations are learned as vector representations, and the connectivity between entities under a certain relation can be calculated in a soft manner based on their representations. Early approaches to knowledge-base embedding are based on matrix factorization [16,17] or non-parametric Bayesian frameworks [18,19]. More recently, the advance of neural embedding methods led to a lot of neural KBE approaches [20]. Bordes et al. [21] designed a translation-based embedding model (transE) to jointly model entities and relationships within a single latent space, which were later generalized into hyperplane translation (transH) [22] and translation in separate entity space and relation spaces (transR) [23].

Researchers attempted to leverage knowledge base embeddings for recommendation. For example, Zhang et al. [24] proposed collaborative KBE to learn the items' semantic representations from the knowledge base for recommendation, but the whole knowledge of an item is learned as a single item vector and the model did not preserve the knowledge-base structure for reasoning; He et al. [25] leveraged translation-based embedding for recommendation by modeling items as entities and users as relations, but they did not consider the knowledge information of users and items for recommendation. Even though many studies have applied neural techniques for recommender systems [26–28], none of the previous work has leveraged KBE for explainable recommendation [1]. However, we will show that a great advantage of learning KBEs is to generate very straight forward explanations by soft-reasoning over the user-to-item paths.

3. Problem Formulation

In this paper, we focus on *explainable product recommendation*, where the objective of the recommender system is to recommend products to users and explain why the products are recommended.

Formally, we first construct a knowledge-base as a set of triplets $S = \{(e_h, e_t, r)\}$ for recommendation, where e_h is a head entity, e_t is a tail entity, and r is the relationship from e_h to e_t. Then the goal of explainable recommendation is twofold, 1) for each user u, find one or a set of items i that are most likely to be purchased by the user, and 2) for each retrieved user-item pair, construct a natural language sentence based on S to explain why the user should purchase the item.

For simplicity, we consider 5 types of entities (i.e., e_h or e_t) for explainable recommendation:

- *user*: the users of the recommender system.
- *item*: the products in the system to be recommended.
- *word*: the words in product names, descriptions or reviews.
- *brand*: the brand/producers of the product.
- *category*: the categories that a product belongs to.

Also, we consider 6 types of relationships (i.e., r) between entities:

- *Purchase*: the relation from a user to an item, which means that the user has bought the item.
- *Mention*: the relation from a user or an item to a word, which means the word is mentioned in the user's or item's reviews.
- *Belongs_to*: the relation from an item to a category, which means that the item belongs to the category.
- *Produced_by*: the relation from an item to a brand, which means that the item is produced by the brand.
- *Bought_together*: the relation from an item to another item, which means that the items have been purchased together in a single transaction.
- *Also_bought*: the relation from an item to another item, which means the items have been purchased by same users.

- *Also_viewed*: the relation from an item to another item, which means that the second item was viewed before or after the purchase of the first item in a single session.

Therefore, for explainable product recommendation, the first goal is to retrieve item i that are likely to have the *Purchase* relationship with user u, and the second goal is to provide explanations for the (u, i) pair based on the relations and entities related to them.

4. Collaborative Filtering on Knowledge Graphs

We now describe our model for explainable recommendation. Our model is a CF model built on user-item knowledge graph. In this section, we first introduce how to model the entities and relations as a product knowledge graph, and then we discuss how to optimize the model parameters for recommendation.

4.1. Relation Modeling as Entity Translations

As discussed previously, we assume that the product knowledge can be represented as a set of triplets $S = \{(e_h, e_t, r)\}$, where r is the relation from entity e_h to entity e_t. Because an entity can be associated with one or more other entities through a single or multiple relations, we propose to separate the modeling of entity and relation for CF. Specifically, we project each entity to a low-dimensional latent space and treat each relation as a translation function that converts one entity to another. Inspired by [21], we represent e_h and e_t as latent vectors $\boldsymbol{e_h} \in \mathbb{R}^d$ and $\boldsymbol{e_t} \in \mathbb{R}^d$, and model their relationship r as a linear projection from e_h to e_t parameterized by $\boldsymbol{r} \in \mathbb{R}^d$, namely,

$$\boldsymbol{e_t} = trans(e_h, r) = \boldsymbol{e_h} + \boldsymbol{r} \quad (1)$$

To learn the entity embeddings, we can construct a product knowledge graph by linking entities with the translation function in the latent space. An example generation process of such a graph is shown in Figure 1.

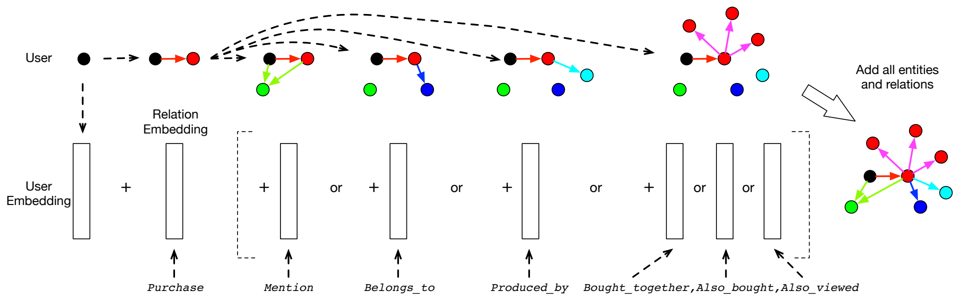


Figure 1. The construction process of knowledge graph with our model. Each entity is represented with a latent vector, and each relation is modeled as a linear translation from one entity to another entity parameterized by the relation embedding.

Solving Equation (1) for all $(e_h, e_t, r) \in S$, however, is infeasible in practice. On one hand, a trivial solution that constructs a single latent vector for all entities with the same type will lead to inferior recommendation performance as it ignores the differences between users and items. On the other hand, deriving a solution that assigns different latent vectors for all entities in S is mathematically impossible because an entity can be linked to multiple entities with a single relationship. For example, we cannot find a single vector for *Also_viewed* that translates an item to multiple items that have different latent representations.

To solve the problems, we propose to relax the constrains of Equation (1) and adopt an embedding-based generative framework to learn it. Empirically, we want the translation model $trans(e_h, r) \approx \boldsymbol{e_t}$ for an observed relation triplet $(e_h, e_t, r) \in S$ and $trans(e_h, r) \neq \boldsymbol{e'_t}$ for an unobserved triplet

$(e_h, e'_t, r) \notin S$. In other words, we want $trans(e_h, r)$ to assign high probability for observing e_t but low probability for observing e'_t, which is exactly the goal of the embedding-based generative framework. The embedding-based generative framework is first proposed by Mikolov et al. [29] and has been widely used in word embedding [29,30], recommendation [31,32], and information retrieval tasks [33,34]. Formally, for an observed relation triplet $(e_h, e_t, r) \in S$, we can learn the translation model $trans(e_h, r)$ by optimizing the generative probability of e_t given $trans(e_h, r)$, which is defined as:

$$P(e_t|trans(e_h, r)) = \frac{\exp(\boldsymbol{e_t} \cdot trans(e_h, r))}{\sum_{e'_t \in E_t} \exp(\boldsymbol{e'_t} \cdot trans(e_h, r))} \tag{2}$$

where E_t is the set of all possible entities that share the same type with e_t.

Because Equation (2) is a softmax function of e_t over E_t, the maximization of $P(e_t|trans(e_h, r))$ will explicitly increase the similarity of $trans(e_h, r)$ and e_t but decease the similarity between $trans(e_h, r)$ and other entities. In this way, we convert Equation (1) into an optimization problem that can be solved with iterative optimization algorithms such as gradient decent. Another advantage of the proposed model is that it provides a theoretically principled method to conduct soft match between tail entities and the translation model. This is important for the extraction of recommendation explanations, which will be discussed in Section 5.

4.2. Optimization Algorithm

For model optimization, we learn the representations of entities and relations by maximizing the likelihood of all observed relation triplets. Let S be the set of observed triplets (e_h, e_t, r) in the training data, then we can compute the likelihood of S defined as

$$\mathcal{L}(S) = \log \prod_{(e_h, e_t, r) \in S} P(e_t|trans(e_h, r)) \tag{3}$$

where $P(e_t|trans(e_h, r))$ is the posterior probability of e_t computed with Equation (2).

The computation cost of $\mathcal{L}(S)$, however, is prohibitive in practice because of the softmax function. For efficient training, we adopt a negative sampling strategy to approximate $P(e_t|trans(e_h, r))$ [35]. Specifically, for each observed relation triplet (e_h, e_t, r), we randomly sample a set of "negative" entities with the same type of e_t. Then the log likelihood of (e_h, e_t, r) is approximated as

$$\log P(e_t|trans(e_h, r)) \approx \log \sigma(\boldsymbol{e_t} \cdot trans(e_h, r)) + k \cdot \mathbb{E}_{e'_t \sim P_t}[\log \sigma(-\boldsymbol{e'_t} \cdot trans(e_h, r))] \tag{4}$$

where k is the number of negative samples, P_t is a predefined noisy distribution over entities with the type of e_t, and $\sigma(x)$ is a sigmoid function as $\sigma(x) = \frac{1}{1+e^{-x}}$. Therefore, $\mathcal{L}(S)$ can be reformulated as the sum of the log-likelihood of $(e_h, e_t, r) \in S$ as

$$\mathcal{L}(S) = \sum_{(e_h, e_t, r) \in S} \log \sigma(\boldsymbol{e_t} \cdot trans(e_h, r)) + k \cdot \mathbb{E}_{e'_t \sim P_t}[\log \sigma(-\boldsymbol{e'_t} \cdot trans(e_h, r))] \tag{5}$$

We also tested the ℓ_2-norm loss function used in TransE model and it does not provide any improvement compared to our inner product-based model with log-likelihood loss, and it is also difficulty for ℓ_2-norm loss to generate expansions, as a result, we adopt our loss function for embedding, recommendation, and explanation in this work. To better illustrate the relationship between our model and a traditional CF method based on matrix factorization, we conduct the following analysis. Inspired by [36], we derive the local objective for the maximization of Equation (5) on a specific relation triplet (e_h, e_t, r):

$$\ell(e_h, e_t, r) = \#(e_h, e_t, r) \cdot \log \sigma(\boldsymbol{e_t} \cdot trans(e_h, r)) + k \cdot \#(e_h, r) \cdot P_t(e_t) \cdot \log \sigma(-\boldsymbol{e_t} \cdot trans(e_h, r)) \tag{6}$$

where $\#(e_h, e_t, r)$ and $\#(e_h, r)$ are the frequency of (e_h, e_t, r) and (e_h, r) in the training data. If we further compute the partial derivative of $\ell(e_h, e_t, r)$ with respect to $x = \boldsymbol{e_t} \cdot trans(e_h, r)$, we have

$$\frac{\partial \ell(e_h, e_t, r)}{\partial x} = \#(e_h, e_t, r) \cdot \sigma(-x) - k \cdot \#(e_h, r) \cdot P_t(e_t) \cdot \sigma(x) \tag{7}$$

When the training process has converged, the partial derivative of $\ell(e_h, e_t, r)$ should be 0, and then we have

$$x = \boldsymbol{e_t} \cdot trans(e_h, r) = \log\left(\frac{\#(e_h, e_t, r)}{\#(e_h, r)} \cdot \frac{1}{P_t(e_t)}\right) - \log k \tag{8}$$

As we can see, the left-hand side is the product of the latent vectors for e_t and $trans(e_h, r)$; and the right-hand side of Equation (8) is a shifted version of the pointwise mutual information between e_t and (e_h, r). Therefore, maximizing the log likelihood of observed triplet set S with negative sampling is actually factorizing the matrix of mutual information between the head-tail entity pairs (e_h, e_t) of relation r. From this perspective, our model is a variation of factorization methods that can jointly factorize multiple relation matrix on a product knowledge graph.

As shown in Equation (8), the final objective of the proposed model is controlled by the noisy distribution P_t. Similar to previous studies [30,33,35], we notice that the relationships with tail entities that have high frequency in the collection reveal less information about the properties of the head entity. Therefore, we define the noisy probability $P_t(e_t)$ for each relation r (except *Purchase*) as the frequency distribution of (e_h, e_t, r) in S so that the mutual information on frequent tail entities will be penalized in optimization process. For *Purchase*, however, we define P_t as a uniform distribution to avoid unnecessary biases toward certain items.

5. Recommendation Explanation with Knowledge Reasoning

In this section, we describe how to create recommendation explanations with the proposed model. We first introduce the concept of explanation path and describe how to generate natural language explanations with it. Then we propose a soft matching algorithm to find explanation path for any user-item pair in the latent space.

An overview of our algorithm is shown in Algorithm 1. In the algorithm, we first conduct breath first search (BFS) with maximum depth z from the user e_u and the item e_i to find an explanation path that can potentially link them. We memorize the paths and compute the path probability with soft matching (Equations (10) and (11)). Finally, we rank the explanation paths by their probabilities and return the best path to create the natural language explanation. In this work, we take z as a hyper-parameter and tune the parameter by increasing its value in the experiments to search for non-empty intersections.

5.1. Explanation Path

The key to generate an explanation of the recommendation is to find a sound logic inference sequence from the user to the item in the knowledge graph. In this work, we propose to find such a sequence by constructing an explanation path between the user and the item in the latent knowledge space.

Formally, let E_h^r and E_t^r be the sets of all possible head entities and tail entities for a relation r. We define an explanation path from entity e_u to entity e_i as two sets of relation $R_\alpha = \{r_\alpha | \alpha \in [1, m]\}$ and $R_\beta = \{r_\beta | \beta \in [1, n]\}$ such that

$$\boldsymbol{e_u} + \sum_{\alpha=1}^{m} \boldsymbol{r_\alpha} = \boldsymbol{e_i} + \sum_{\beta=1}^{n} \boldsymbol{r_\beta} \tag{9}$$

where $e_u \in E_h^{r_\alpha}$ for $\alpha = 1$, $e_i \in E_h^{r_\beta}$ for $\beta = 1$, $E_t^{r_{\alpha-1}} = E_h^{r_\alpha}$ for $\alpha \in [2, m]$, $E_t^{r_{\beta-1}} = E_h^{r_\beta}$ for $\beta \in [2, n]$, and $E_t^{r_\alpha} = E_t^{r_\beta}$ for $\alpha = m, \beta = n$. In other words, there is an explanation path between e_u and e_i if there is an entity that can be inferred by both e_u (with R_α) and e_i (with R_β) with the observed relations in the knowledge graph.

```
Algorithm 1: Recommendation Explanation Extraction
  Input: $S = \{(E_h, E_t, r)\}$, $e_u$, $e_i$, maximum depth $z$
  Output: $e_x, R_\alpha, R_\beta$
  Procedure Main()
1     $V_u, P_u, \mathcal{R}_u = BFS(S, e_u, z)$.
2     $V_i, P_i, \mathcal{R}_i = BFS(S, e_i, z)$.
3     $P \leftarrow \{\}$.
4     for $e \in V_u \cap V_i$ do
5         $P[e] = P_u(e) \cdot P_i(e)$.
      end
6     Pick up $e_x \in V_u \cap V_i$ with the largest $P[e]$.
7     $R_\alpha = \mathcal{R}_u[e_x], R_\beta = \mathcal{R}_i[e_x]$.
8     return $e_x, R_\alpha, R_\beta$
  Function BFS($S, e, z$)
9     $V_e \leftarrow$ all entities in the entity set $E_t$ within $z$ hops from $e$.
10    $P_e \leftarrow$ the probability of each entity in $V_e$ computed by Equation (10).
11    $\mathcal{R}_e \leftarrow$ the paths from $e$ to the space of each entity in $V_e$.
12    return $V_e, P_e, \mathcal{R}_e$;
```

For better illustration, we depict a recommendation example where an item (*iPad*) is recommended to a user (*Bob*) in Figure 2. As we can see, the word "IOS" can be inferred by *iPad* and *Bob* using the relation *Mention*; the brand *Apple* can be inferred by *iPad* using *Produced_by*, and by *Bob* using *Purchase* + *Produced_by*. Thus, we have two explanation paths that link *Bob* with *iPad*.

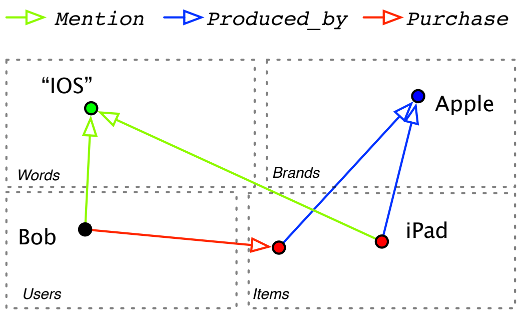


Figure 2. Example explanation paths between a user *Bob* and a recommended item *iPad* in the product knowledge graph.

To generate explanations, we can create simple templates base on the relation type and apply them to the explanation paths. For example, we can say that *Bob* may be interested in *iPad* because he often *mentions* "IOS" in his reviews, and "IOS"is often *mentioned* in the reviews of *iPad*; or that *Bob* may be interested in *iPad* because he often *purchases* products *produced by Apple*, and *iPad* is also *produced* by *Apple*. In these explanations, the italic words are entities and relations on the explanation path.

5.2. Entity Soft Matching

Finding a valid explanation path with observed relations, however, is often difficult for an arbitrary user-item pair. In practice, product knowledge graphs tend to be sparse. For example, the density of user-item matrix in Amazon review datasets is usually below 0.1% [37]. Thus, the available relation triplets in the observed data are limited. To solve the problem, we propose to conduct entity soft matching in the latent space for explanation construction.

As discussed in Section 4.1, we learn the distributed representations of entities and relations by optimizing the generative probability of observed relation triplets in Equation (2). Thus, we can extend the softmax function to compute the probability of entity $e_x \in E_t^{r_m}$ given e_u and the relation set $R_\alpha = \{r_\alpha | \alpha \in [1, m]\}$ as

$$P(e_x | trans(e_u, R_\alpha)) = \frac{\exp(\boldsymbol{e_x} \cdot trans(e_u, R_\alpha))}{\sum_{e' \in E_t^{r_m}} \exp(\boldsymbol{e'} \cdot trans(e_u, R_\alpha))} \tag{10}$$

where $E_t^{r_m}$ is the tail entity set of r_m, and $trans(e_u, R_\alpha) = \boldsymbol{e_u} + \sum_{\alpha=1}^{m} \boldsymbol{r_\alpha}$. Therefore, we can construct an explanation path for an arbitrary user e_u and item e_i with relation sets $R_\alpha = \{r_\alpha | \alpha \in [1, m]\}$ and $R_\beta = \{r_\beta | \beta \in [1, n]\}$ through any intermediate $e_x \in E_t^{r_m}$, and compute the probability of this explanation path as:

$$P(e_x | e_u, R_\alpha, e_i, R_\beta) = P(e_x | trans(e_u, R_\alpha)) P(e_x | trans(e_i, R_\beta)) \tag{11}$$

To find the best explanation for (e_u, e_i), we can rank all paths by $P(e_x | e_u, R_\alpha, e_i, R_\beta)$ and pick up the best one to generate natural language explanations with predefined templates. It should be noted that learning with single hops may not guarantee the quality of multiple hops matching, but it also significantly simplifies the design of the training algorithm and increases the generalizability of the model, and helps to generate explanations more easily. Besides, using the single-hop training strategy has already been also to compete with many of the baselines. However, we believe that model training with multiple hops directly is a promising problem and we will design new models for this problem as a future work.

6. Experimental Setup

In this section, we introduce the test bed of our experiments and discuss our evaluation settings in details.

6.1. Datasets

We conducted experiments on the Amazon review dataset [37], which contains product reviews in 24 categories on Amazon.com and rich metadata such as prices, brands, etc. Specifically, we used the 5-core data of *CDs and Vinyl*, *Clothing*, *Cell Phones*, and *Beauty*, in which each user or item has at least 5 associated reviews.

Statistics about entities and relations used in our experiments are shown in Table 1. Overall, the interactions between users, items and other entities are highly sparse. For each dataset, we randomly sampled 70% of user purchase as the training data and used the rest 30% as the test set. This means that each user has at least 3 reviews observed in the training process and 2 reviews hidden for evaluation purposes. Thus, the objective of product recommendation is to find and recommend items that are purchased by the user in the test set.

Table 1. Statistics of the 5-core datasets for *CDs & Vinyl*, *Clothing*, *Cell Phones & Accessories*, and *Beauty* in Amazon.

	CDs & Vinyl	Clothing	Cell Phones & Accessories	Beauty
Entities				
#Reviews	1,097,591	278,677	194,439	198,502
#Words per review	174.57 ± 177.05	62.21 ± 60.16	93.50 ±131.65	90.90 ± 91.86
#Users	75,258	39,387	27,879	22,363
#Items	64,443	23,033	10,429	12,101
#Brands	1414	1182	955	2077
#Categories	770	1,193	206	248
Density	0.023%	0.031%	0.067%	0.074%
Relations				
#*Purchase* per user	14.58 ± 39.13	7.08 ± 3.59	6.97 ± 4.55	8.88 ± 8.16
#*Mention* per user	2545.92 ± 10,942.31	440.20 ± 452.38	652.08 ± 1335.76	806.89 ± 1344.08
#*Mention* per item	2973.19 ± 5490.93	752.75 ± 909.42	1743.16 ± 3482.76	1491.16 ± 2553.93
#*Also_bought* per item	57.28 ± 39.22	61.35 ± 32.99	56.53 ± 35.82	73.65 ± 30.69
#*Also_viewed* per item	0.27 ± 1.86	6.29 ± 6.17	1.24 ± 4.29	12.84 ± 8.97
#*Bought_together* per item	0.68 ± 0.80	0.69 ± 0.90	0.81 ± 0.77	0.75 ± 0.72
#*Produced_by* per item	0.21 ± 0.41	0.17 ± 0.38	0.52 ± 0.50	0.83 ± 0.38
#*Belongs_to* per item	7.25 ± 3.13	6.72 ± 2.15	3.49 ± 1.08	4.11 ± 0.70

6.2. Evaluation

To verify the effectiveness of the proposed model, we adopt six representative and state-of-the-art methods as baselines for performance comparison. Three of them are traditional recommendation methods based on matrix factorization (BPR [38], BPR-HFT [39], and VBPR [40]), and the other three are deep models for product recommendation (DeepCoNN [32], CKE [24], and JRL [31]).

- **BPR:** The Bayesian personalized ranking [38] model is a popular method for top-N recommendation that learns latent representations of users and items by optimizing the pairwise preferences between different user-item pairs. In this paper, we adopt matrix factorization as the prediction component for BPR.
- **BPR-HFT:** The hidden factors and topics (HFT) model [39] integrates latent factors with topic models for recommendation. The original HFT model is optimized for rating prediction tasks. For fair comparison, we learn the model parameters under the pairwise ranking framework of BPR for top-N recommendation.
- **VBPR:** The visual Bayesian personalized ranking [40] model is a state-of-the-art method that incorporate product image features into the framework of BPR for recommendation.
- **TransRec:** The translation-based recommendation approach proposed in [25], which takes items as entities and users as relations, and leveraged translation-based embeddings to learn the similarity between user and items for personalized recommendation. We adopted L_2 loss function, which was reported to have better performance in [25]. Notice that TransRec is different from our model because our model treats both items and users as entities, and learns embedding representations for different types of knowledge (e.g., brands, categories) as well as their relationships.
- **DeepCoNN:** The Deep Cooperative Neural Networks model for recommendation [32] is a neural model that applies a convolutional neural network (CNN) over the textual reviews to jointly model users and items for recommendation.
- **CKE:** The collaborative KBE model is a state-of-the-art neural model [24] that integrates text, images, and knowledge base for recommendation. It is similar to our model as they both use text and structured product knowledge, but it builds separate models on each type of data to construct item representations while our model constructs a knowledge graph that jointly embeds all entities and relations.

- **JRL:** The joint representation learning model [31] is a state-of-the-art neural recommender, which leverage multi-model information including text, images and ratings for Top-N recommendation.

The performance evaluation is conducted on the test set where only purchased items are considered to be relevant to the corresponding user. Specifically, we adopt four ranking measures for top-N recommendation, which are the Normalized Discounted Cumulative Gain (**NDCG**), Precision (**Prec.**), **Recall**, and the percentage of users that have at least one correct recommendation (Hit-Ratio, **HR**). All ranking metrics are computed based on the top-10 results for each test user. Significant test is conducted based on the Fisher randomization test [41].

6.3. Parameter Settings

Our model is trained with stochastic gradient decent on a Nvidia Titan X GPU. We set the initial learning rate as 0.5 and gradually decrease it to 0.0 during the training process. We set the batch size as 64 and clip the norm of batch gradients with 5. For each dataset, we train the model for 20 epochs and set the negative sampling number as 5. We tune the dimension of embeddings from 10 to 500 ([10, 50, 100, 200, 300, 400, 500]) and report the best performance of each model in Section 7.

We also conduct five-fold cross-validation on the training data to tune the hyper-parameters for baselines. For BPR-HFT, the best performance is achieved when the number of topics is 10. For BPR and VBPR, the regularization coefficient $\lambda = 10$ worked the best in most cases. Similar to our model, we tune the number of latent factors (the embedding size) from 10 to 500 and only report the best performance of each baseline.

7. Results and Discussion

In this section, we discuss the experimental results. We first compare the performance of our model with the baseline methods on top-N recommendation. Then we conduct case studies to show the effectiveness of our model for recommendation explanation.

7.1. Recommendation Performance

Our experiments mainly focus on two research questions:

- **RQ1:** Does incorporating knowledge-base in our model produce better recommendation performance?
- **RQ2:** Which types of product knowledge are most useful for top-N recommendation?
- **RQ3:** What is the computational efficiency of our model compared to other recommendation algorithms?

To answer **RQ1**, we report the results of our model and the baseline methods in Table 2. As shown in Table 2, the deep models with rich auxiliary information (DeepCoNN, CKE, and JRL) perform better in general than the shallow methods (BPR, BPR-HFT, VBPR, TransRec) on most datasets, which is coherent with previous studies [24,31,32]. Among different neural baselines, JRL obtains the best performance in our experiments. It produced 80% or more improvements over the matrix factorization baselines and 10% or more over the other deep recommendation models. Overall, our model outperformed all the baseline models consistently and significantly. It obtained 5.6% NDCG improvement over the best baseline (i.e., JRL) on *CDs and Vinyl*, 78.16% on *Clothing*, 23.05% on *Cell Phones*, and 45.56% on *Beauty*. This shows that the proposed model can effectively incorporate product knowledge graph and is highly competitive for top-N recommendation.

Figure 3 depicts the recommendation performance of our model and baseline methods with different embedding sizes on *CDs & Vinyl* and *Beauty* datasets. Observations on the other two datasets are similar. As shown in Figure 3, the recommendation methods based on shallow models (BPR, BPR-HFT, and VBPR) obtain the best NDCG when the embedding size is fairly small (from 10 to 100), and larger embedding sizes usually hurt the performance of these models. In contrast to the shallow models, the results of neural models (i.e., JRL, DeepCoNN, CKE, and our model) show positive correlations with the increase of embedding sizes. For example, the NDCG of the best baseline (JRL) and our model improves when the embedding size increases from 10 to 300, and remains stable afterwards. Overall, our model is robust to the variation of embedding sizes and consistently outperformed the baselines.

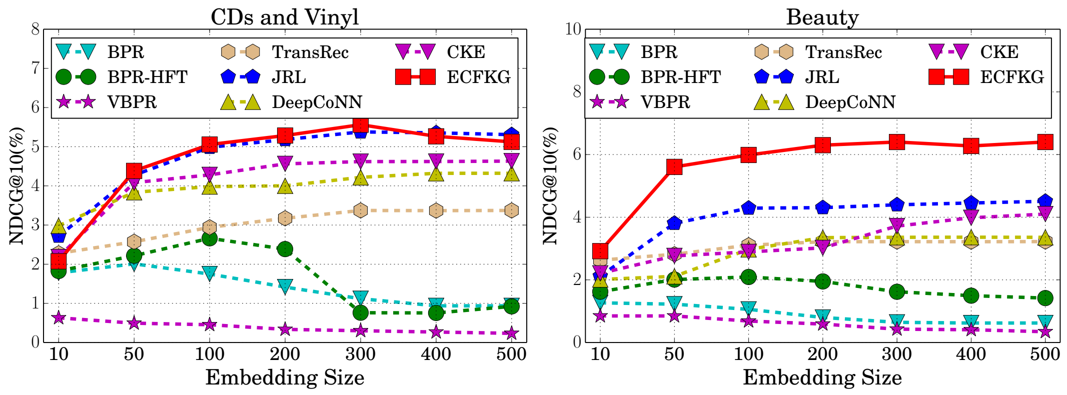


Figure 3. The NDCG@10 performance of our model (ECFKG, Explainable Collaborative Filtering over Knowledge Graph) and the baseline methods under different embedding sizes.

In Table 2, we see that the CKE model did not perform as well as we expected. Although it has incorporated reviews, images and all other product knowledge described in this paper, the CKE model did not perform as well as JRL and our model. One possible reason is that CKE only considers heterogeneous information in the construction of item representations, but it does not directly leverage the information for user modeling. Another potential reason is that CKE separately constructs three latent spaces for text, image and other product knowledge, which makes it difficult for information from different types of data to propagate among the entities. Either way, this indicates that the embedding-based relation modeling of our model is a better way to incorporate structured knowledge for product recommendation.

From the results we can also see that datasets of different density result in different performance in our model. In particular, denser datasets (*Cell Phone* and *Beauty*) generally get better ranking performance than sparser datasets (*CD & Vinyl* and *Clothing*) in our model, which means more sufficient information can help to learn better models in our algorithm.

To answer **RQ2**, we experiment the performance of our model when using different relations. Because we eventually need to provide item recommendations for users, our approach would at least need the *Purchase* relation to model the user purchase histories. As a result, we train and test our model built with only the *Purchase* relation, as well as *Purchase* plus one another relation separately. As shown in Table 3, the relative performance of our models built on different relations varies considerably on the four datasets, which makes it difficult to conclude which type of product knowledge is the globally most useful one. This, however, is not surprising because the value of relation data depends on the properties of the candidate products. On *CDs and Vinyl*, where most products are music CDs, the CD covers did not reveal much information, and people often express their tastes and preferences in the reviews they wrote. Thus *Mention* turns out to be the most useful relation. On *Clothing*, however, reviews are not as important as the appearance or picture of the clothes, instead, it is easier to capture item similarities from the items that have been clicked and viewed by the same user. Therefore, adding *Also_view* relation produces the largest performance improvement for our model on *Clothing*.

Table 2. Performance of the baselines and our model on top-10 recommendation. All the values in the table are percentage numbers with '%' omitted, and all differences are significant at $p < 0.05$. The stared numbers (*) indicate the best baseline performances, and the bolded numbers indicate the best performance of each column. The last line shows the percentage improvement of our model against the best baseline (i.e., JRL), which are significant at $p < 0.001$.

Dataset	**CDs and Vinyl**				**Clothing**				**Cell Phones**				**Beauty**			
Measures (%)	**NDCG**	**Recall**	**HR**	**Prec.**	**NDCG**	**Recall**	**HR**	**Prec.**	**NDCG**	**Recall**	**HR**	**Prec.**	**NDCG**	**Recall**	**HR**	**Prec.**
BPR	2.009	2.679	8.554	1.085	0.601	1.046	1.767	0.185	1.998	3.258	5.273	0.595	2.753	4.241	8.241	1.143
BPR-HFT	2.661	3.570	9.926	1.268	1.067	1.819	2.872	0.297	3.151	5.307	8.125	0.860	2.934	4.459	8.268	1.132
VBPR	0.631	0.845	2.930	0.328	0.560	0.968	1.557	0.166	1.797	3.489	5.002	0.507	1.901	2.786	5.961	0.902
TransRec	3.372	5.283	11.956	1.837	1.245	2.078	3.116	0.312	3.361	6.279	8.725	0.962	3.218	4.853	9.867	1.285
DeepCoNN	4.218	6.001	13.857	1.681	1.310	2.332	3.286	0.229	3.636	6.353	9.913	0.999	3.359	5.429	9.807	1.200
CKE	4.620	6.483	14.541	1.779	1.502	2.509	4.275	0.388	3.995	7.005	10.809	1.070	3.717	5.938	11.043	1.371
JRL	5.378 *	7.545 *	16.774 *	2.085 *	1.735 *	2.989 *	4.634 *	0.442 *	4.364 *	7.510 *	10.940 *	1.096 *	4.396 *	6.949 *	12.776 *	1.546 *
Our model	**5.563**	**7.949**	**17.556**	**2.192**	**3.091**	**5.466**	**7.972**	**0.763**	**5.370**	**9.498**	**13.455**	**1.325**	**6.399**	**10.411**	**17.498**	**1.986**
Improvement	3.44	5.35	4.66	5.13	78.16	82.87	72.03	72.62	23.05	26.47	22.99	20.89	45.56	49.82	36.96	28.46

Table 3. Performance of our model on top-10 recommendation when incorporating *Purchase* with other types of relation separately. And the bolded numbers indicate the best performance of each column. All the values in the table are percentage numbers with '%' omitted, and all differences are significant at $p < 0.05$.

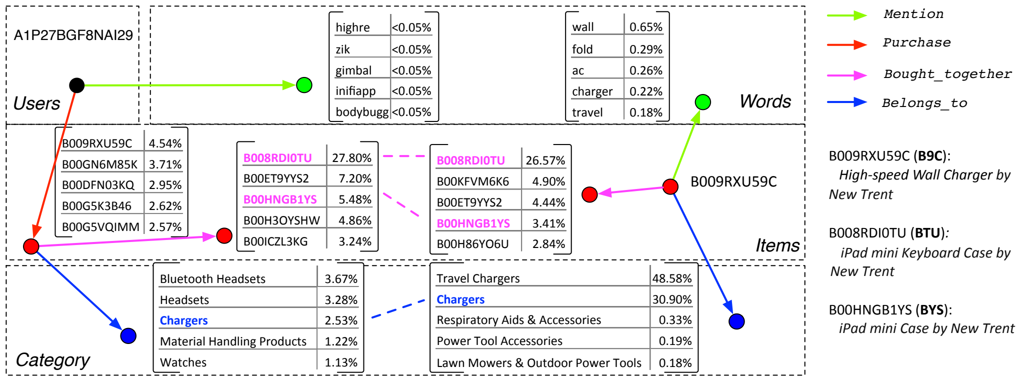
Relations	**CDs and Vinyl**				**Clothing**				**Cell Phones**				**Beauty**			
Measures(%)	**NDCG**	**Recall**	**HT**	**Prec**	**NDCG**	**Recall**	**HT**	**Prec**	**NDCG**	**Recall**	**HT**	**Prec**	**NDCG**	**Recall**	**HT**	**Prec**
***Purchase* only**	1.725	2.319	7.052	0.818	0.974	1.665	2.651	0.254	2.581	4.526	6.611	0.649	2.482	3.834	7.432	0.948
+Also_view	1.722	2.356	6.967	0.817	1.800	3.130	4.672	0.448	2.555	4.367	6.417	0.630	4.592	7.505	12.901	1.511
+Also_bought	3.641	5.285	12.332	1.458	1.352	2.419	3.580	0.343	4.095	7.129	10.051	0.986	4.301	6.994	11.908	1.408
+Bought_together	1.962	2.712	7.473	0.861	0.694	1.284	2.026	0.189	3.173	5.572	7.952	0.784	3.341	5.337	9.556	1.181
+Produced_by	1.719	2.318	6.842	0.792	0.579	1.044	1.630	0.155	2.852	4.982	7.274	0.719	3.707	5.939	10.660	1.287
+Belongs_to	2.799	4.028	10.297	1.200	1.453	2.570	3.961	0.376	2.807	4.892	7.242	0.717	3.347	5.382	9.994	1.193
+Mention	3.822	5.185	12.828	1.628	1.019	1.754	2.780	0.265	3.387	5.806	8.548	0.848	3.658	5.727	10.549	1.305
***+all* (our model)**	**5.563**	**7.949**	**17.556**	**2.192**	**3.091**	**5.466**	**7.972**	**0.763**	**5.370**	**9.498**	**13.455**	**1.325**	**6.370**	**10.341**	**17.131**	**1.959**

Overall, it is difficult to find a product knowledge that is universally useful for recommending products from all categories. However, we see that by modeling all of the heterogenous relation types, our final model outperforms all the baselines and outperforms all the simplified versions of our model with one or two types of relation, which implies that our KBE approach to recommendation is scalable to new relation types, and it has the ability to leverage very heterogeneous information sources in a unified manner.

To answer **RQ3**, we compare the training efficiency of different methods in our experiments on the same Nvidia Titan X GPU platform. The testing procedures for all methods are quite efficient, and generating the recommendation list for a particular user on the largest *CDs & Vinyl* dataset requires less than 20 milliseconds. This is because after the model has learned the embeddings of the users and items, generating the recommendation list does not require re-computation of the embeddings and only needs to calculate their similarity. In terms of training efficiency, shallow models can be much more efficient than deep neural models, which is not surprising. For example, BPR or BPR-HFT can be trained within 30 minutes on the largest *CDs & Vinyl* dataset, while deep models such as DeepCoNN, CKE, and JRL takes about 10 hours on the same dataset, but they also bring much better recommendation performance. Our Explainable CF over Knowledge Graph (ECFKG) approach takes comparable training time on the largest dataset (about 10 hours), while achieving better recommendation performance than other deep neural baselines, which is a good balance between efficiency and effectiveness.

7.2. Case Study for Explanation Generation

To show the ability of our model to generate knowledge-enhanced explanations, we conduct case study for a test user (i.e., *A1P27BGF8NAI29*) from *Cell Phones*, for whom we have examined that the first recommendation (i.e., *B009RXU59C*) provided by the system is correct. We plot the translation process of this user to other entity subspaces with *Purchase*, *Mention*, *Bought_together*, and *Belongs_to* relations, as shown in Figure 4. We also show the translation of the first recommended item *B009RXU59C* (*B9C*) using the same relations. The top 5 entities retrieved by our system for each translation are listed along with their probabilities computed based on Equation (10).


A1P27BGF8NAI29
Users
Words
Items
Category
B009RXU59C
Mention
Purchase
Bought_together
Belongs_to
B009RXU59C (B9C): High-speed Wall Charger by New Trent
B008RDI0TU (BTU): iPad mini Keyboard Case by New Trent
B00HNGB1YS (BYS): iPad mini Case by New Trent

highre	<0.05%
zik	<0.05%
gimbal	<0.05%
inifiapp	<0.05%
bodybugg	<0.05%

wall	0.65%
fold	0.29%
ac	0.26%
charger	0.22%
travel	0.18%

B009RXU59C	4.54%
B00GN6M85K	3.71%
B00DFN03KQ	2.95%
B00G5K3B46	2.62%
B00G5VQIMM	2.57%

B008RDI0TU	27.80%
B00ET9YYS2	7.20%
B00HNGB1YS	5.48%
B00H3OYSHW	4.86%
B00ICZL3KG	3.24%

B008RDI0TU	26.57%
B00KFVM6K6	4.90%
B00ET9YYS2	4.44%
B00HNGB1YS	3.41%
B00H86YO6U	2.84%

Bluetooth Headsets	3.67%
Headsets	3.28%
Chargers	2.53%
Material Handling Products	1.22%
Watches	1.13%

Travel Chargers	48.58%
Chargers	30.90%
Respiratory Aids & Accessories	0.33%
Power Tool Accessories	0.19%
Lawn Mowers & Outdoor Power Tools	0.18%

Figure 4. Example explanation paths between the user *A1P27BGF8NAI29* and the item *B009RXU59C* (B9C) in *Cell Phones*.

As we can see in Figure 4, there are three explanation paths between the user *A1P27BGF8NAI29* and the item *B9C*. The first and second paths are constructed by *Purchase* and *Bought_together*. According to our model, the user is linked to *B008RDI0TU* (*BTU*) and *B00HNGB1YS* (*BYS*) through *Purchase+Bought_together* with probabilities as 27.80% and 5.48%. The item *B9C* is linked to *BTU* and *BYS* through *Bought_together* directly with probabilities as 26.57% and 3.41%. The third path is constructed by *Purchase* and *Belongs_to*. The user is linked to the category *Chargers* with probability as 2.53% and *B9C* is linked to *Chargers* with probability as 30.90%. Therefore, we can create three natural language explanations for the recommendation of *B9C* by describing these explanation paths with simple templates. Example sentences and the corresponding confidences are listed below:

- *B9C* is recommended because the user often purchases items that are bought with *BTU* together, and *B9C* is also frequently bought with *BTU* together (27.80% $\times$ 26.57% $=$ 7.39%).
- *B9C* is recommended because the user often purchases items that are bought with *BYS* together, and *B9C* is also frequently bought with *BYS* together (5.48% $\times$ 3.41% $=$ 0.19%).
- *B9C* is recommended because the user often purchases items related to the category *Chargers*, and *B9C* belongs to the category *Chargers* (2.53% $\times$ 30.90% $=$ 0.78%).

Among the explanation sentences, the best explanation should be the one with the highest confidence, which is the first sentence in this case.

To better evaluate the quality of these recommendation explanations, we look at the details of each product shown in Figure 4. On Amazom.com, *B9C* is a *High-speed Wall Charger by New Trend* for tablets, *BTU* is an *iPad mini Keyboard Case*, and *BYS* is an *iPad Air Keyboard Case*. If the generated explanations are reasonable, this means that the user has purchased some items that were frequently co-purchased with iPad accessories. Also, this indicates that there is a high probability that the user has an iPad. For validation proposes, we list the five training reviews written by the user in Table 4.

As we can see in Table 4, the user has purchased several tablet accessories such as Bluetooth headsets and portable charger. The second review even explicitly mentions that the user has possessed an iPad and expresses concerns about the "running-out-of-juice" problem. Therefore, it is reasonable to believe that the user is likely to purchase stuff that are frequently co-purchased with iPad accessories such as *iPad mini Keyboard Case* (*BTU*) or *iPad Air Keyboard Case* (*BYS*), which are recommended as top items in our algorithm, and are also well-explained by the explanation paths in the knowledge graph.

Table 4. The reviews written by the user *A1P27BGF8NAI29* in the training data of *Cell Phones*.

Review of Jabra VOX Corded Stereo Wired **Headsets** ... *I like to listen to music at work, but I must wear some sort of* ***headset*** *so that I do not create a disturbance. So, I have a broad experience in* ***headsets*** *...*
Review of OXA Juice Mini M1 2600mAh ... *I recently had an experience, where I was about town and need to recharge my* ***iPad****, and so I tried this thing out. I plugged in the* ***iPad****, and it quickly charged it up, and at my next destination it was ready to go ...*
Review of OXA 8000mAh Solar External Battery Pack Portable ... *This amazing gadget is a solar powered* ***charger*** *for your small electronic device. This* ***charger*** *is (according to my ruler) 5-1/4 inches by 3 inches by about 6 inches tall. So, it is a bit big to place in the pocket ...*
Review of OXA Bluetooth Wristwatch Bracelet ... *I was far from thrilled with* ***Bluetooth headset*** *that I had, so I decided to give this device a try. Pros: The bracelet is not bad looking, ...* Review of OXA Mini Portable Wireless **Bluetooth** Speaker ... *This little gadget is a* ***Bluetooth*** *speaker. It's fantastic! This speaker fits comfortably in the palm of your hand, ...*

8. Conclusions and Outlook

In this paper, we propose to learn over heterogenous KBE for personalized explainable recommendation. To do so, we construct the user-item knowledge graph to incorporate both user behaviors and our knowledge about the items. We further learn the KBE with the heterogenous relations collectively, and leverage the user and item embeddings to generate personalized recommendations. To explain the recommendations, we devise a soft matching algorithm to find explanation paths between a user and the recommended items in the latent KBE space. Experimental results on real-world datasets verified the superior performance of our approach, as well as its flexibility to incorporate multiple relation types.

After years of success at integrating machine learning into recommender systems, we believe that equipping the systems with knowledge (again) is important to the future of recommender systems—or information systems in a broader sense—which can help to improve both the performance and explainability in the future.

Author Contributions: Y.Z. and Q.A.: come up with the idea and design the model. Q.A.: Model implementation and finishing the first version of the paper. Y.Z., V.A, and X.C.: Proof reading and revise the paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Y.; Chen, X. Explainable Recommendation: A Survey and New Perspectives. *arXiv* **2018**, arXiv:1804.11192.
2. Zhang, Y.; Lai, G.; Zhang, M.; Zhang, Y.; Liu, Y.; Ma, S. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, Gold Coast, Australia, 6–11 July 2014; pp. 83–92.
3. Herlocker, J.L.; Konstan, J.A.; Riedl, J. Explaining collaborative filtering recommendations. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, PA, USA, 2–6 December 2000; pp. 241–250.
4. Tintarev, N.; Masthoff, J. A survey of explanations in recommender systems. In Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop, Istanbul, Turkey, 11–15 April 2007; pp. 801–810.
5. Bilgic, M.; Mooney, R.J. Explaining recommendations: Satisfaction vs. promotion. In Proceedings of the Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research at the 2005 International Conference on Intelligent User Interfaces, San Diego, CA, USA, 9 January 2005; p. 153.
6. Cramer, H.; Evers, V.; Ramlal, S.; Van Someren, M.; Rutledge, L.; Stash, N.; Aroyo, L.; Wielinga, B. The effects of transparency on trust in and acceptance of a content-based art recommender. *UMUAI* **2008**, *18*, 455.
7. Tintarev, N.; Masthoff, J. Designing and Evaluating Explanations for Recommender Systems; Springer Press: Boston, MA, USA, 2011; pp. 479–510.
8. Burke, R. Integrating knowledge-based and collaborative-filtering recommender systems. In Proceedings of the Workshop on AI and Electronic Commerce, Orlando, FL, USA, 18–19 July 1999; pp. 69–72.
9. Shari, T. Knowledge-based recommender systems. *Encycl. Libr. Inf. Sci.* **2000**, *69*, 180.
10. Ghani, R.; Fano, A. Building recommender systems using a knowledge base of product semantics. In Proceedings of the Workshop on Recommendation and Personalization in ECommerce at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web based Systems, Malaga, Spain, 29–31 May 2002, pp. 27–29.
11. Heitmann, B. An open framework for multi-source, cross-domain personalisation with semantic interest graphs. In Proceedings of the Sixth ACM Conference on Recommender Systems, Dublin, Ireland, 9–13 September 2012; pp. 313–316.
12. Musto, C.; Lops, P.; Basile, P.; de Gemmis, M.; Semeraro, G. Semantics-aware graph-based recommender systems exploiting linked open data. In Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, Halifax, NS, Canada, 13–17 July 2016; pp. 229–237.
13. Noia, T.D.; Ostuni, V.C.; Tomeo, P.; Sciascio, E.D. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Trans. Intell. Syst. Technol.* **2016**, *8*, 9.
14. Oramas, S.; Ostuni, V.C.; Noia, T.D.; Serra, X.; Sciascio, E.D. Sound and music recommendation with knowledge graphs. *ACM Trans. Intell. Syst. Technol.* **2017**, *8*, 21.
15. Catherine, R.; Mazaitis, K.; Eskenazi, M.; Cohen, W. Explainable Entity-based Recommendations with Knowledge Graphs. *arXiv* **2017**, arXiv:1707.05254.
16. Nickel, M.; Tresp, V.; Kriegel, H.P. A Three-Way Model for Collective Learning on Multi-Relational Data. In Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011; pp. 809–816.
17. Singh, A.P.; Gordon, G.J. Relational learning via collective matrix factorization. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 650–658.

18. Miller, K.; Jordan, M.I.; Griffiths, T.L. Nonparametric latent feature models for link prediction. In Proceedings of the Advances in neural information processing systems, Vancouver, BC, Canada, 7–10 December 2009; pp.1276–1284.
19. Zhu, J. Max-margin nonparametric latent feature models for link prediction. In Proceedings of the 29th International Conference on Machine Learning, Edinburgh, UK, 27 June–3 July 2012.
20. Bordes, A.; Weston, J.; Collobert, R.; Bengio, Y.; others. Learning Structured Embeddings of Knowledge Bases. In Proceedings of the 25th AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 9–10 August 2011; p. 6.
21. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 2787–2795.
22. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the 28th AAAI Conference on Artificial Intelligence, Quebec, QC, Canada, 27–31 July 2014; pp. 1112–1119.
23. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; p. 2181.
24. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W.Y. Collaborative knowledge base embedding for recommender systems. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, USA, 13–17 August 2016; pp. 353–362.
25. He, R.; Kang, W.C.; McAuley, J. Translation-based Recommendation. In Proceedings of the 11th ACM Conference on Recommender Systems, Como, Italy, 27 –31 August 2017; pp. 161–169.
26. Li, S.; Kawale, J.; Fu, Y. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In Proceedings of the 24th ACM Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October 2015; pp. 811–820.
27. Wang, H.; Wang, N.; Yeung, D.Y. Collaborative Deep Learning for Recommender Systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; pp. 1235–1244.
28. Li, X.; She, J. Collaborative Variational Autoencoder for Recommender Systems. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 305–314.
29. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
30. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1188–1196.
31. Zhang, Y.; Ai, Q.; Chen, X.; Croft, W.B. Joint representation learning for top-n recommendation with heterogeneous information sources. In Proceedings of the 2017 ACM Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1449–1458.
32. Zheng, L.; Noroozi, V.; Yu, P.S. Joint deep modeling of users and items using reviews for recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 425–434.
33. Ai, Q.; Yang, L.; Guo, J.; Croft, W.B. Analysis of the paragraph vector model for information retrieval. In Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval, Newark, DE, USA, 12–16 September 2016; pp. 133–142.
34. Ai, Q.; Zhang, Y.; Bi, K.; Chen, X.; Croft, W.B. Learning a hierarchical embedding model for personalized product search. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, 7–11 August 2017; pp. 645–654.
35. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
36. Levy, O.; Goldberg, Y. Neural word embedding as implicit matrix factorization. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2177–2185.

37. McAuley, J.; Targett, C.; Shi, Q.; Van Den Hengel, A. Image-based recommendations on styles and substitutes. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 43–52.
38. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
39. McAuley, J.; Leskovec, J. Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; pp. 165–172.
40. He, R.; McAuley, J. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 144–150.
41. Smucker, M.D.; Allan, J.; Carterette, B. A comparison of statistical significance tests for information retrieval evaluation. In Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, Lisbon, Portugal, 6–10 November 2007; pp. 623–632.