Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences

Yixin Cao
National University of Singapore
caoyixin2011@gmail.com

Xiang Wang
National University of Singapore
xiangwang1223@gmail.com

Xiangnan He*
University of Science and Technology of China
xiangnanhe@gmail.com

Zikun Hu
National University of Singapore
zikunhu1016@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

Incorporating knowledge graph (KG) into recommender system is promising in improving the recommendation accuracy and explainability. However, existing methods largely assume that a KG is complete, transferring the "knowledge" in KG at the shallow level of entity raw data or embeddings. This may lead to suboptimal performance, since a practical KG can hardly be complete, and it is common that a KG has missing facts, relations, and entities. Thus, we argue that it is crucial to consider the incomplete nature of KG when incorporating it into recommender system.

In this paper, we jointly learn the model of recommendation and knowledge graph completion. Distinct from previous KG-based recommendation methods, we transfer the **relation** information in KG, so as to understand the reasons that a user likes an item. As an example, if a user has watched several movies *directed by* (relation) the *same person* (entity), we can get that the director relation plays a critical role when the user makes decision, understanding the user's preference in a finer granularity.

Technically, we contribute a new translation-based recommendation model, which specially accounts for various preferences in translating a user to an item, and then jointly train it with a KG completion model by combining several transfer schemes. Extensive experiments on two benchmark datasets show that our method outperforms state-of-the-art KG-based recommendation methods. Further analysis verifies the positive effect of joint training on both tasks of recommendation and KG completion, and the advantage of our model in understanding user preference. We publish our project at https://github.com/TaoMiner/joint-kg-recommender.

KEYWORDS

Item Recommendation;Knowledge Graph;Embedding;Joint Model;

*Corresponding author.

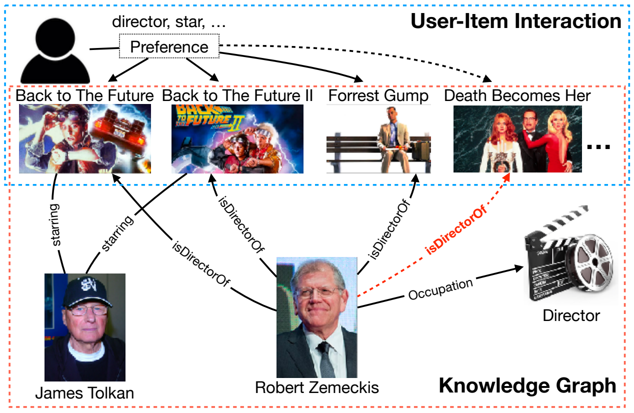


Figure 1: An illustrative example on the necessity of considering the missing relations in KG for recommendation.

1 INTRODUCTION

Knowledge Graph (KG) is a heterogeneous structure that stores the world's facts in the form of machine-readable graphs, where nodes denote entities, and edges denote the relations between entities. Since it's proposed, KG has attracted much attention in many research areas, ranging from recommendation [39], question answering [11], to information extraction [3]. Focusing on recommendation, the structural knowledge has shown great potential by providing rich information about items, being a promising solution to improve the accuracy and explainability of recommender systems.

Nevertheless, existing KGs (e.g., DBPedia [20]) are far from complete, which limits the benefits of the transferred knowledge. As illustrated in Figure 1, the red dashed line between *Robert Zemeckis* and *Death Becomes Her* indicates a missing relation *isDirectorOf*. Assuming that the user has chosen movies *Back to The Future I & II* and *Forrest Gump*, by using the KG, we can attribute the reason of the user's choices to the director *Robert Zemeckis*. In this case, although we have accurately captured the user's preference on movies, we may still fail to recommend *Death Becomes Her* (which is also of interest to the user), due to the missing relation in the KG (cf. the red dashed line). As such, we believe that it is critical to consider the incomplete nature of KG when using it for recommendation, and more interestingly, can the completion of KG benefit from the improved modeling of user-item interactions in recommender system?

In this paper, we propose to unify the two tasks of recommendation and KG completion in a joint model for mutually enhancements. The basic idea is twofold: 1) utilizing the facts in KG as auxiliary data to augment the modeling of user-item interaction, and 2) completing the missing facts in KG based on the enhanced user-item modeling. For example, we are able to understand the user's preference on director via the related entities and relations; meanwhile we can predict that *Robert Zemeckis* is the director of *Death Becomes Her*, if there exist some users who like the movie also have a preference of other movies directed by *Robert Zemeckis*.

Although many prior efforts have leveraged KG in recommender systems [17, 28, 29, 43, 45, 47], there is very few work jointly modeling the two tasks of knowledge graph learning and recommendation. CoFM [30] is the most similar work that aligned two latent vector spaces in each task together by regularizing or sharing entity and item embeddings if they refer to the same thing. However, it ignores the important role of entity relations in user-item modeling, and fail to provide the interpretation ability.

In this work, we propose a **T**ranslation-based **U**ser **P**reference model (TUP) to integrate with KG learning seamlessly. The key idea is that there are multiple (implicit) relations between users and items, which reveal the preferences (i.e., reasons) of users on consuming items. An example of the "preference" is the director information in Figure 1 that drives the user to decide watching movies *Back to Future I & II* and *Forest Gump*. While we can pre-define the number of preferences and train TUP from user-item interaction data, the preferences are expressed as latent vectors that are opaque to understand. To endow the preferences with explicit semantics, we align them with the relations in KG, capturing the intuition that the type of item attributes plays a crucial role in user decision-making process. Technically speaking, we transfer the relation embeddings as well as entity embeddings learned from KG to TUP, simultaneously training the KG completion and recommendation tasks. We term the method as **K**nowledge-enhanced TUP (KTUP) that jointly learns the representations of users, items, entities, and relations, which successfully captures complementary information from the two tasks to facilitate their mutual enhancements.

The main contributions of this work are summarized as follows:

- We propose a new translation-based model, which exploits the implicit preference representations to capture the relations between users and items.
- We emphasize the importance of jointly modeling item recommendation and KG completion to couple the preference representations with the knowledge-aware relations, empowering the model with explainability.
- We perform extensive experiments on two datasets for top-N recommendation and KG completion, verifying the rationality of joint learning. Experimental results demonstrate the effectiveness and explainability of our model.

2 RELATED WORK

Our proposed methods involve two tasks: item recommendation and KG completion. In this section, we first introduce previous work for each task, then discuss their relations.

2.1 Item Recommendation

In the early stage of item recommendation, researchers focus on recommending similar users or items to a target user using history interactions alone, such as collaborative filtering (CF) [34], factorization machines [32], matrix factorization techniques [19], BPRMF [33]. The key challenge lies in extracting features of users and items to compute their similarity, namely **similarity-based methods**.

With the surge of neural network (NN) models, a host of methods extend **similarity-based methods with NN** and present a more effective mechanism in automatically extracting latent features of users and items for recommendation [7, 13–15]. However, they still suffer from the data sparsity issue and cold-start problem. **Content-based methods** deal with the issues by introducing various side information, such as contextual reviews [9, 24], relational data [10, 35] and knowledge graphs [6]. Another advantage of additional contents is the explainable ability to understand why an item is recommended out of others, which is then proved to be important for the effectiveness, efficiency, persuasiveness, and user satisfaction of recommender systems [8, 38, 46].

Among the side information, knowledge graphs (e.g., DBPedia [20]) show great potentials on recommendations due to its well-defined structures and adequant resources. This type of methods mostly transfer structural knowledge of entities from KG to user-item interaction modeling based on the given mapping between entities and items. We roughly classify them into two groups: the methods augmenting the data of user-item pairs with KG triplets, and the methods combining item and entity embeddings learned from different sources. In the first group, Piao and Breslin [28] extracted lightweight features drived from KG (i.e. property-object, subject-property) for factorization machines. Zhang et al. [45] constructed a unified graph by adding a *buy* relation between users and items, then applied transE [2] to model relational data. On the other hand, the methods in second group usually improve the quality of item embeddings using entity embeddings if they refer to the same thing [17, 43]. Piao and Breslin [29] summerized the recommendation results using different entity embeddings (i.e. node2vec, doc2vec and transE), and found that node2vec improves the most. CoFM [30] first took into account the refinements of entity embeddings from user-item modeling as another transfering task. However, the above methods heavily rely on the alignments between items and entities. Zhou et al. [47] introduced entity concepts in KG to deal with the sparisity issue of the alignments, but still fail to consider the importance of entity relations in transfering knowledge from KG.

Another line of work is **translation-based recommendation**, inspired by KG representation learning. It assumes that the selection of items satisfies translational relations in the latent vector space, where the relations are either regarded as related to users in sequential recommendation [12], or modeled implicitly via Memory-based Attention [36]. We thus improve this type of method by considering the N-to-N issue[1] in modeling user preferences as translational relations, which shall be further enhanced by transfering knowledge of entities and their relations from KG.

[1]N-to-N issue here means that one user may like multiple items, and also, several users may like a single item, which will be detailed in Section 4.2.

2.2 KG Completion

External knowledge has been proved to be effective in many tasks of natural language processing, such as question answering [44], which accelerates the popularity of KGs. Although there are a host of methods for finding entities [4, 5] and their relations from texts [21], existing KGs are far from complete. Recent studies for KG completion show a great enthusiasm for learning low-dimensional represention of entities and relations while perserving structural knowledge of the graph. We roughly categorize such representation learning methods into two groups: translational distance models and semantic matching models.

TransE [2] first proposed the core idea of translational distance models that the relationship between two entities corresponds to a translation in their vector space. Although it is simple and effective, but sometimes also confusing because some relations shall translate one entity to various entities, namely 1-to-N problem, and similarly, N-to-1 and N-to-N problems. To address these issues, a host of methods extended TransE by introducing additional hyperplanes [41], vector spaces [22], textual information [40] and relational path [21].

The second group measures plausibility of facts by matching semantic representations of entities and relations through similarity-based scoring functions. RESCAL [26] represented each relation as a matrix to capture the compositional semantics between entities, and utilized a bilinear function as similarity metrics. To simplify the learning of relation matrices, DistMult [42] restricted them to be diagonal, HolE [25] defined a circular correlation [31] to compress relation matrices as vectors, and ComplEx [37] introduced complex-value for asymmetric relations. Instead of modeling the compositional relation, another line of methods directly introduce NNs for matching. SME [1] learned relation specific layers for head entity and tail entity, respectively, and then feeded them into final matching layer (e.g., dot production), while NAM [23] conducted semantic matching with a deep architecture.

2.3 Relationship between Two Tasks

Items usually correspond to entities in many fields, such as books, movies and musics, making it possible for transfering information between them. These information involving in two tasks are complementary revealing the connectivity among items or between users and items. In terms of models, the two tasks are both to rank candidates for a target according to either implict or explict relations. For example, KG completion is to find correct movies (e.g., *Death Becomes Her*) for the person *Robert Zemeckis* given the explicit relation *isDirectorOf*. Item recommendation aims at recommending movies for a target user satisfying some implicit preference. Therefore, we are to fill in the gap between item recommendation and KG completion via a joint model, and systematically investigate how the two tasks impact each other.

3 PRELIMINARY

Before introducing our proposed methods, let's first formally define the two tasks as well as TransH [41] as the component for KG completion in our model.

3.1 Tasks and Notations

Item Recommendation : Given a list of user-item interactions $\mathcal{Y} = \{(u, i)\}$, we use implicit feedback as the protocol so that each pair (u, i) implies the user $u \in \mathcal{U}$ consumes the item $i \in \mathcal{I}$. The goal is to recommend top-N items for a target user.

KG Completion : A **Knowledge Graph** $\mathcal{KG}$ is a directed graph composed of *subject-property-object* triple facts. Each triplet denotes that there is a relationship r from head entity e_h to tail entity e_t, formally defined by (e_h, e_t, r), where $e_h, e_t \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ are relations. Due to the incompleteness nature of KGs, KG completion is to predict the missing entity e_h or e_t for a triplet (e_h, e_t, r), which can also be regarded as to recommend top-N entities for a target (e_t, r) or (e_h, r).

TUP denotes the model for item recommendation. It takes a list of user-item pairs $\mathcal{Y}$ as input, and outputs a relevance score $g(u, i; p)$ indicating the likelihood that u likes i, given the preference $p \in \mathcal{P}$, where the number of the preference set $\mathcal{P}$ is predefined. For each user-item pair, we induce a preference, serving as a similar role with the relation for two entities. To deal with the N-to-N issue, we introduce preference hyperplanes, and assign each preference with two vectors: $\mathbf{w}_p$ for the projection to a hyperplane, $\mathbf{p}$ for the translation between users and items.

KTUP is a multi-task architecture. Given $\mathcal{KG}$, $\mathcal{Y}$, and a set of item-entity alignments $\mathcal{A} = \{(i, e) | i \in \mathcal{I}, e \in \mathcal{E}\}$, where each (i, e) means that i can be mapped to an entity e in the given KG. KTUP is able to output not only $g(u, i; p)$, but also a score $f(e_h, e_t, r)$ indicating how possible the fact is true, based on the jointly learned embeddings of users $\mathbf{u}$, items $\mathbf{i}$, preferences $\mathbf{p}, \mathbf{w}_p$, entities $\mathbf{e}$ and relations $\mathbf{r}, \mathbf{w}_r$.

Example 3.1. As shown in Figure 1, given a user, the interacted movies (e.g., *Back to The Future I & II* and *Forrest Gump*), and the related triplets, KTUP is able to (1) figure out the user preference of the *isDirectorOf* relation on movies, (2) recommend the movie *Death Becomes Her* based on the induced preference, and (3) predict the missing head or tail entity for the triplet (*Death Becomes Her-isDirectorOf-Robert Zemeckis*). The above three goals shall be achieved by considering not only the structural knowledge in KG but also the user-item interactions.

Next, we will briefly describe transH as the module of KG completion in our joint model.

3.2 TransH for KG Completion

Learning distributional representations of KG provides an effective and efficient way in manipulating entities while perserving their structural knowledge. TransE [2] is a widely used method due to its simplicity and remarkable effectiveness. Its basic idea is to learn embeddings for entities and relations, satisfying $\mathbf{e}_h + \mathbf{r} \approx \mathbf{e}_t$ if there is a triplet (e_h, e_t, r) in KG. However, a single relation type may correspond to multiple head entities or tail entities, leading to serious 1-to-N, N-to-1 and N-to-N issues [41].

Therefore, TransH [41] learns different representations for an entity conditioned on different relations. It assumes that each relation owns a hyperplane, and the translation between head entity and tail entity is valid only if they are projected to the hyperplane. It defines an energy score function for a triplet as follows:

$$f(e_h, e_t, r) = \| \mathbf{e}_h^{\perp} + \mathbf{r} - \mathbf{e}_t^{\perp} \| \tag{1}$$

where a lower score of $f(e_h, e_t, r)$ indicates that the triplet is possbile true, and otherwise not. $\mathbf{e}_h^{\perp}$ and $\mathbf{e}_t^{\perp}$ are projected entity vectors:

$$\mathbf{e}_h^{\perp} = \mathbf{e}_h - \mathbf{w}_r^{\mathrm{T}} \mathbf{e}_h \mathbf{w}_r \tag{2}$$

$$\mathbf{e}_t^{\perp} = \mathbf{e}_t - \mathbf{w}_r^{\mathrm{T}} \mathbf{e}_t \mathbf{w}_r \tag{3}$$

where $\mathbf{w}_r$ and $\mathbf{r}$ are two learned vectors of relation r, $\mathbf{w}_r$ denotes the projection vector of the corresponding hyperplane, and $\mathbf{r}$ is the translation vector. $\| \cdot \|$ denotes the L1-norm distance function used in the presented paper.

Finally, the training of TransH encourages the discrimination between valid triplets and incorrect ones using margin-based ranking loss:

$$\mathcal{L}_k = \sum_{(e_h, e_t, r) \in \mathcal{KG}} \sum_{(e'_h, e'_t, r') \in \mathcal{KG}^-} [f(e_h, e_t, r) + \gamma - f(e'_h, e'_t, r')]_+ \tag{4}$$

where $[\cdot]_+ \triangleq \max(0, \cdot)$, $\mathcal{KG}^-$ contains incorrect triplets constructed by replacing head entity or tail entity in a valid triplet randomly, and γ controls the margin between positive and negative triplets.

4 TUP FOR ITEM RECOMMENDATION

Inspired by the above translation assumption between two entities in KG, we propose TUP to explictly models user preferences and regards them as translational relationships between users and items. Given a set of user-item interactions $\mathcal{Y}$, it automatically induces a preference for a user-items pair, and learns the embeddings of preference $\mathbf{p}$, user $\mathbf{u}$ and item $\mathbf{i}$, satisfying $\mathbf{u} + \mathbf{p} \approx \mathbf{i}$.

Considering the implicity and variety of user preferences, we design two main components in TUP: Preference Induction and Hyperplane-based Translation.

4.1 Preference Induction

Given a user-item pair (u, i), this component is to induce a preference from a set of latent factors $\mathcal{P}$. These factors are shared by all users, and each $p \in \mathcal{P}$ denotes a different preference, which aims at capturing the commonness among users as global features complementary to user embeddings that focus on a single user locally. Similar with topic models, the number $P = |\mathcal{P}|$ is a hyperparameter, and we cannot nominate the exact meaning of each preference. With the help of KG, the number of preferences can be automatically set and each preference is assigned with explanations (Section 5).

We design two strategies for preference induction: a hard approach that selects one out of the P preferences, and a soft way that combines all preferences with attentions.

4.1.1 Hard Strategy. The intuition behind our hard strategy is that only a single preference takes effect when a user makes a decision over items. We use Straight-Through (ST) Gumbel SoftMax [18] for discretely sampling the preference given a user-item pair, which utilizes reparameterization trick for backpropagation, making it possible to calculate continuous gradients of model parameters during end-to-end training.

ST Gumbel SoftMax approximately samples one-hot vectors from a multi-classification distribution. Assuming that the probability of belonging to class p in a P-way classification distribution is defined as log softmax:

$$\phi(p) = \frac{\exp(\log(\pi_p))}{\sum_{j=1}^{P} \exp(\log(\pi_j))} \tag{5}$$

where π_p is the unnormalized output of a score function. Then, we sample a one-hot vector $\mathbf{z} = [z_1, \cdots, z_P] \in \mathbb{R}^P$ from the above distribution as follows:

$$z_p = \begin{cases} 1, & p = \arg\max_j (\log(\pi_j) + g_j) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $g = -\log(-\log(u))$ is the Gumbel noise and u is generated by a certain noise distribution (e.g., $u \sim \mathcal{N}(0, 1)$). The noise term increases the stochastic of $\arg\max$ function and makes the process become equivalent to drawing a sample accroding to a continuous probability distribution $\mathbf{y} = [y_1, \cdots, y_p, \cdots, y_P]$:

$$y_p = \frac{\exp((\log(\pi_p) + g_p)/\tau)}{\sum_{j=1}^{P} \exp((\log(\pi_j) + g_j)/\tau)} \tag{7}$$

This is called Gumbel-Softmax distribution, where τ is a temperature parameter. The related proofs can be found in original papers.

Straight-Through (ST) gumbel-Softmax takes different paths in the forward and backward propagation, so as to maintain sparsity yet support stochastic gredient descent (SGD). In the forward pass, it discretizes the continuous probability distribution for a one-hot vector as mentioned above. And in the backward pass, it simply follows the continuous $\mathbf{y}$, thus the error signal is still able to backpropagate.

In the hard strategy, we define the score function for π_p as the similarity between the user-item pair and preference:

$$\phi(u, i, p) = \text{Similarity}(\mathbf{u} + \mathbf{i}, \mathbf{p}) \tag{8}$$

We use dot product as similarity function in presented work.

4.1.2 Soft Strategy. Actually, a user might like an item according to various factors, which have no distinct boundary. Instead of selecting the most prominent preference, the soft strategy is to combine multiple preferences via the attention mechanism:

$$\mathbf{p} = \sum_{p' \in \mathcal{P}} \alpha_{p'} \mathbf{p}' \tag{9}$$

where $\alpha_{p'}$ is the attention weight of preference p', and defined as proportional to the similarity score:

$$\alpha_{p'} \propto \phi(u, i, p') \tag{10}$$

4.2 Hyperplane-based Translation

Inspired by TransH, we introduce the hyperplane to deal with the variety of preferences. That is, different users may share the same preference to different items (i.e., N-to-N issue), which is pretty common in practice. Obviously, it is confusing for the TransE-like transition: the embeddings of items are close as long as a user likes them both due to some preference (Figure 2(a)), leading to an incorrection conclusion that a user consuming one shall consume the other one, no matter with which preference.

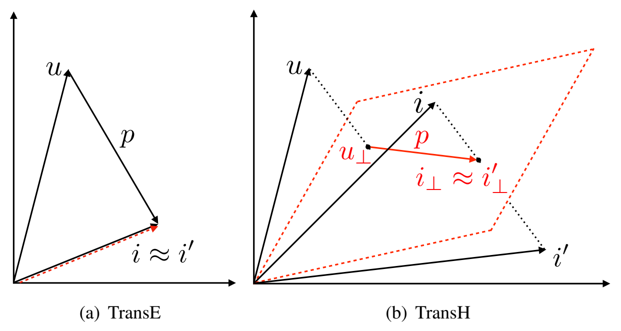


Figure 2: Illustration of the two translation schemes for item recommendation

Such limitations are alleviated by introducing perference hyperplanes as illustrated in Figure 2(b): i and i' have different representations, and are similar only when they are projected to a specific hyperplane. We thus define the hyperplane-based translation function as follows:

$$g(u, i; p) = \| \mathbf{u}^\perp + \mathbf{p} - \mathbf{i}^\perp \| \tag{11}$$

where $\mathbf{u}^\perp$ and $\mathbf{i}^\perp$ are projected vectors of the user and the item, and are obtained through the induced preference p that plays a similar role as relations in TransH:

$$\mathbf{u}^\perp = \mathbf{u} - \mathbf{w}_p^{\mathrm{T}} \mathbf{u} \mathbf{w}_p \tag{12}$$

$$\mathbf{i}^\perp = \mathbf{i} - \mathbf{w}_p^{\mathrm{T}} \mathbf{i} \mathbf{w}_p \tag{13}$$

where $\mathbf{w}_p$ is the projection vector that is obtained along with the induction process of preferences p: either to pick up the corresponding one using the hard strategy, or through attentive addition of all projection vectors based on the induced attention weights in the soft strategy:

$$\mathbf{w}_p = \sum_{p' \in \mathcal{P}} \alpha_{p'} \mathbf{w}_{p'} \tag{14}$$

We encourage that the translation distances of interacted items are smaller than random ones for each user through BPR Loss function:

$$\mathcal{L}_p = \sum_{(u,i) \in \mathcal{Y}} \sum_{(u,i') \in \mathcal{Y}'} -\log \sigma[g(u, i'; p') - g(u, i; p)] \tag{15}$$

where $\mathcal{Y}'$ contains negative interactions by randomly corrupting an interacted item to a non-interacted one for each user.

Tranditional methods (e.g., BPRMF [33]) recommend items for a user by computing a scalar score based on user and item embeddings, which indicates to which extent the user prefers to the item. Instead, we model preferences as vectors in order to (1) capture the commonness among users as global latent features, compared with the user embeddings which only concerns the local features of the user alone, and (2) reflect richer semantics for explainable ability.

5 JOINT LEARNING VIA KTUP FOR TWO TASKS

Unifying the two tasks of item recommendation and KG completion, KTUP extends the translation-based recommendation model, TUP, by incorporating KG knowledge of entities as well as relations. Intuitively, the auxiliary knowledge supplements the connectivity among items as constraints to model user-item pairs. On the other hand, the understanding of users' preferences to items shall reveal their commonness related to some relation types and entities, which may be missing in the given KG.

5.1 KTUP

Figure 3 presents the overall framework of KTUP. In the left side is the inputs: user-item interactions, knowledge graph and the alignments between items and entities. At the top-right corner is TUP for item recommendation, and TransH for knowledge graph completion is at the bottom-right corner. KTUP jointly learns the two tasks by enhancing the embeddings of items and preferences with that of entities and relations. we define the knowledge enhanced TUP translation function as follows:

$$g(u, i; p) = \| \mathbf{u}^\perp + \hat{\mathbf{p}} - \hat{\mathbf{i}}^\perp \| \tag{16}$$

where $\hat{\mathbf{i}}^\perp$ is the projected vector for enhanced item embedding $\hat{\mathbf{i}}$ by the corresponding entity embedding $\mathbf{e}$:

$$\hat{\mathbf{i}}^\perp = \hat{\mathbf{i}} - \hat{\mathbf{w}}_p^{\mathrm{T}} \hat{\mathbf{i}} \hat{\mathbf{w}}_p \tag{17}$$

$$\hat{\mathbf{i}} = \mathbf{i} + \mathbf{e}, \ (i, e) \in \mathcal{A} \tag{18}$$

And $\hat{\mathbf{p}}$ and $\hat{\mathbf{w}}_p$ are the translation vector and the projection vector enhanced by those of the corresponding relation embedding according a predefined one-to-one mapping $\mathcal{R} \to \mathcal{P}$. We obtain these two vectors as follows:

$$\hat{\mathbf{p}} = \mathbf{p} + \mathbf{r} \tag{19}$$

$$\hat{\mathbf{w}}_p = \mathbf{w}_p + \mathbf{w}_r \tag{20}$$

Thus, for entities and items, the enhanced item embeddings contain the relational knowledge among items that is complementary to user-item interactions, and improves item recommendation, since the entity embedding $\mathbf{e}$ perserves the structural knowledge in KG. Meanwhile, the entity embedding $\mathbf{e}$ shall be fine tuned by the additional connectivity through users and items during backpropagation. Note that we don't use the combined embeddings for both tasks, because it makes the embeddings of items the same as corresponding entities in two tasks, which actually degrades our model to share embeddings between items and entities.

For relations and preferences, the usage of relations not only offers explicit interpretation of explainability, but also further combines two tasks more sufficiently at model level. On one hand, through the one-to-one mapping, the meaning of each preference is revealed by the relation label. For instance, the relation *isDirectorOf* reveals a preference to director, or *starring* for a preference to movie stars. On the other hand, many items have no aligned entities due to the incompleteness of KG, which limits the mutual impacts to the alignments between entities and items in the models that only transfer

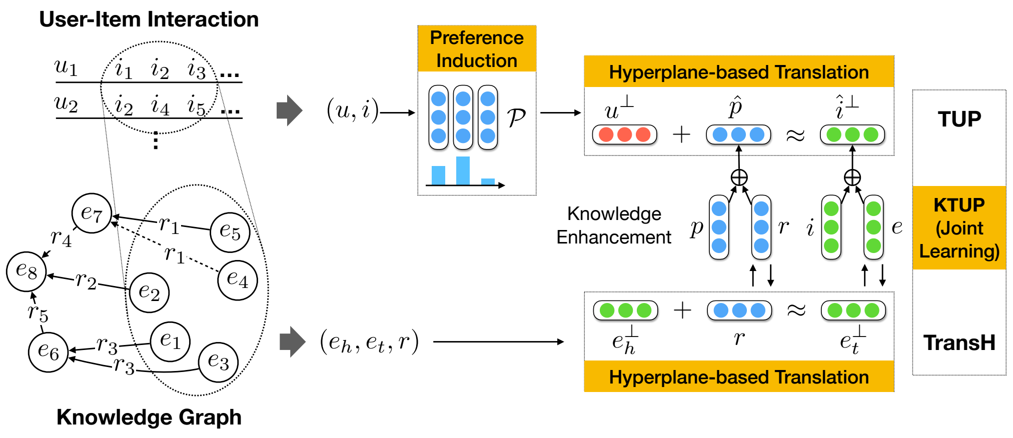


Figure 3: Framwork of KTUP. At the top is TUP for item recommendation including two components: preference induction and hyperplane-based translation. KTUP jointly learns TUP and TransH to enhance the item and preference modeling by transfering knowledge of entities as well as relations.

knowledge of entities. Considering that each user-item pair has a preference and so does the relations between two entities, KTUP optimizes all users, items and entities more thoroughly.

5.2 Training

We thus train KTUP using the overall objective function as follows:

$$\mathcal{L} = \lambda \mathcal{L}_p + (1 - \lambda)\mathcal{L}_k \tag{21}$$

where λ is a hyperparameter to balance the two tasks.

5.3 Relationship to SOTA Models

In this section, we give a discussion on the relationship between KTUP and the other state-of-the-art recommendation methods based on KG to facilite a deep understanding of the investigation between two tasks in Section 6. We choose three typical models that transfer knowledge of entities at data level (CFKG [45]), at embedding level (CKE [43]) and in both directions (CoFM [30]). We summerize the main difference and similarities from the following aspects:

Implicity of User Preference CKE and CoFM can be regarded as extentions of collaborative filtering. This type of methods consider the preferences from users to items implicitly and rely on their embeddings to compute a score (i.e., dot product) indicating in which degree the user likes the item. CFKG and KTUP model the preferences explictly and learn the vectoral representations instead of scalars to capture more comprehensive semantics.

Variety of User Preference CFKG defines the only *buy* preference between users and items, which obviously suffers from the serious N-to-N issue and fails to deal with it through the TransE-like scoring function. TKUP distinguishes different user perferences and introduces hyperplanes for each preference as well as each relation to learn various representations of items and entities.

Transfered Knowledge from KG CKE and CoFM only focus on transfering knowledge of entities. CFKG also transfers relations in the way of data integration through a unifed graph. Except for entities and items, KTUP combines the embeddings of relations and preferences according to predefined one-to-one mapping, which brings another byproduct of the explainable ability to recommendation mechanism.

6 EXPERIMENTS

In this section, we conduct quantitative experiments on separate tasks of item recommendation and KG completion. For each task, we use two datasets in different domains and give further evaluations on the influence of data sparsity as well as the N-to-N issue. We then investigate the mutual impacts between the two tasks during jointly training. Finally, we describe real examples for qualitative analysis to give an intuitive impression of the explainability. We publish our project at https://github.com/TaoMiner/joint-kg-recommender.

6.1 Datasets

Following CoFM [30], we use two public available datasets in the movie and book domains: MovieLens-1m [27] and DBbook2014 [2]. Both of them consist of users and their ratings on movies or books, which are then refined for LODRecSys [16, 27, 28] by mapping items into DBPedia entities if there is a mapping available. Following most item recommendation work that models implicit feedback [39], we treat existing ratings as positive interactions, and generate negative ones by randomly corrupting items.

To collect the related facts from DBPedia, we only consider those triplets that are directly related to the entities with mapped items, no matter which role (i.e. subject or object) the entity serves as. We then preprocess the two datasets by: filtering out low frequency users and items (i.e., lower than 10 in MovieLens and 5 in DBbook), filtering out infrequent entities (i.e., lower than 10 in both datasets), cutting off unrelated relations and merging similar relations manually.

Table 1 shows the statistics of MovieLens-1m and DBbook2014 datasets[3]. After preprocessing, there are 6,040 users and 3,230 items with 998,539 ratings in Movielens-1m, the average number of ratings

[2] http://2014.eswc-conferences.org/important-dates/call-RecSys.html
[3] Because we cannot find the released triplets for the two datasets, we collect them as our KG as mentioned above, which leads to a little difference with those papers using the same datasets (e.g., CoFM [30]).

Table 1: Statistics of MovieLens-1m and DBbook2014

		MovieLens-1m	**DBbook2014**
User-Item Interactions	# Users	6,040	5,576
	# Items	3,240	2,680
	# Ratings	998,539	65,961
	# Avg. ratings	165	12
	Sparsity	94.9%	99.6%
KG	# Entity	14,708	13,882
	# Relation	20	13
	# Triple	434,189	334,511
Multi-Tasks	# Item-Entity Alignments	2,934	2,534
	Coverage	90.6%	94.6%

per user is 165 and the sparisity rate is 94.9%. The data sparisity issue is more severe in DBbook2014. It consists of 5,576 users and 2,680 items with 65,961 ratings, where the average number of ratings per user is 12 and the sparisity rate reaches up to 99.6%. The triplets used in two datasets are at the same scale, where the subgraph for MovieLens-1m composes of 434,189 triplets with 14,708 entities and 20 relations, and the subgraph of DBbook has 334,511 triplets with 13,882 entities and 13 relations. Note that the alignments between items and entities for transfering are fewer in MovieLens-1m than that in DBbook2014.

6.2 Baselines

For item recommendation, we compare our proposed models with the following state-of-the-art baselines involving typical similarity-based methods and KG-based methods.

- Typical similarity-based methods: we choose the widely used collaborative filtering models, **FM** [32] and **BPRMF** [33], because they are the foundations of other baselines and also achieve the state-of-the-art performances on many datasets.
- KG-based methods: **CFKG** [45] integrated the data of two sources and applied TransE on a unifed graph including users, items, entities and relations. **CKE** [43] combined various item embeddings from different sources including TransR on KG. **CoFM** [30] jointly trained FM and TransE by sharing parameters or regularization of aligned items and entities. We mark the two schemes as CoFM (share) and CoFM (reg), respectively.

For KG completion, we choose the typical methods **TransE** [2], **TransH** [41] and **TransR** [22] that are widely used in this field. Also, we evaluate the above KG-based methods even if they have not been done in the original papers to investigate the impacts of different transfer schemes.

For fair comparison, we carefully reimplement them in our released codes because they didn't report the results on the same datasets and we cannot find their released codes. Note that we remove the components of side information modeling like reviews and visual information, since they are not available in the datasets and out of our scope in this paper.

6.3 Training Details

We construct training set, validation set and test set by randomly spliting the dataset with the ratio of $7:1:2$. For item recommendation, we split the items for each user and ensure at least one item exist in the test set.

For hyperparameters, we apply a grid search on BPRMF and TransE to find out the best settings for each task, and use them for all of other models since they contribute the basic learning ideas[4]. The learning rate is searched in $\{0.0005, 0.005, 0.001, 0.05, 0.01\}$, the coefficient of L_2 regularization is in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0\}$, and the optimization methods include Adaptive Moment Estimation (Adam), Adagrad and SGD. Finally, we set learning rate as 0.005 and 0.001 for item recommendation and KG completion, respectively, L_2 coefficient is set to 10^{-5} and 0, and the optimization methods is set to Adagrad and Adam. Particularly, for the models involving two tasks, we have tried both sets of parameters, and pick up the latter set of parameters due to its superior performance.

Other hypermeters are empirically set as follows: the batch size is 256, the embedding size is 100 and we perform early stopping strategy on validation sets.

We predefine the number of preferences in TUP as 20 and 13 for MovieLens-1m and DBbook2014, respectively, which are set according to the relations of the collected triplets. For the models involving two tasks (i.e., CFKG, CKE, CoFM and KTUP), we set the joint hyperparameter λ as 0.5 and 0.7 on two datasets after searching in $\{0.7, 0.5, 0.3\}$, so as to balance their impacts, and use pretrained embeddings of the basic model (i.e., BPRMF and TransE).

The main goal in this paper is to investigate the mutual impacts on each task during jointly training, rather than achieving best performance by tuning parameters. Thus, our proposed models as well as baseline methods are trained once for each dataset and evaluate for both tasks of item recommendation and KG completion.

6.4 Item Recommendation

In this section, we evaluate our models as well as baseline methods on the task of item recommendation. Given a user, we take all items in test sets as candidates and rank them according to the scores computed based on the embeddings of users and items. Thus, the N items ranked at top are recommended items.

6.4.1 Metrics. We use five evaluation metrics that have been widely used in previous work:

- Precision@N: Precision at rank N is the fraction of the items recommended that are relevant to the user. We compute the mean of all users as the final precision.
- Recall@N: Recall at rank N is the proportion of the items relevant to the user that have been successfully recommended. We compute the mean of all users as the final recall.
- F1 score@N: F1 score at rank N is the harmonic mean of precision at rank N and recall at rank N.
- Hit ratio@N: Hit at rank N is 1 if any gold items are recommended within the top N items, otherwise 0. We compute the mean of all users as the final hit ratio score.
- nDCG@N: Normalized Discounted Cumulative Gain (nDCG) is a standard measure of ranking quality, considering the

[4]We have tested other parameters for baselines and find performance drop.

Table 2: Overall Performances on Item Recommendation

	MovieLens-1m (@10, %)					DBbook2014 (@10, %)				
	Precision	**Recall**	**F1**	**Hit**	**NDCG**	**Precision**	**Recall**	**F1**	**Hit**	**NDCG**
FM	29.28	11.92	13.81	81.06	59.48	3.44	21.55	5.75	30.15	20.10
BPRMF	30.81	12.95	14.84	83.18	61.02	3.56	22.46	5.96	31.26	21.01
CFKG	29.45	12.49	14.23	82.24	58.97	3.17	19.69	5.30	28.09	19.87
CKE	38.67	16.65	18.94	88.36	67.05	3.92	23.41	6.51	33.18	**27.78**
CoFM (share)	32.08	13.02	15.12	83.30	58.69	3.41	20.78	5.67	29.84	20.92
CoFM (reg)	31.74	12.74	14.87	82.67	58.66	3.32	20.54	5.54	28.96	20.53
TUP (hard)	37.29	17.07	18.98	**89.60**	67.40	3.40	21.11	5.67	29.56	20.19
TUP (soft)	37.00	16.79	18.76	89.47	67.02	3.62	22.81	6.06	31.42	21.54
KTUP (hard)	40.87	17.24	19.79	88.97	69.65	4.04	24.48	6.71	34.49	27.38
KTUP (soft)	**41.03**	**17.25**	**19.82**	89.03	**69.92**	**4.05**	**24.51**	**6.73**	**34.61**	27.62

graded relevance among positive and negative items within the top N of the ranking list.

6.4.2 Overall Results. Table 2 shows the overall performances of our proposed models as well as the baseline methods, where *hard* and *soft* denote the two preference induction strategies in Section 4.1. We can see:

- Our proposed methods significantly outperform all the baseline methods on both datasets. Particularly, TUP performs competitively to other KG-based models, while it doesn't require any additional information. This is because TUP automatically infers the knowledge of preferences from user-item interactions, and performs better especially when the interaction data is sufficient, like MovieLens-1m. By incorporating KG, KTUP further improves it and presents more promising improvements on DBbook than MovieLens (i.e., 11.06% v.s. 4.43% gains in F1), which implies that the knowledge is more helpful for sparse data.
- The hard strategy performs better than the soft strategy only when it is used for TUP on MovieLens-1m, which implies that to induce a deterministic user perference requires sufficient data, and the soft strategy is more robust.
- CFKG and CoFM perform slightly better than the typical models (i.e., FM and BPRMF) on MovieLens-1m, but perform worse on the sparse dataset of DBbook2014. One possible reason is that they both transfer entities by forcing their embeddings to be similar with aligned items, leading to the loss of knowledge that has been perserved in the embeddings, and the loss becomes more serious when there is no sufficient training data.
- CKE achieves pretty good performances on two datasets mainly because it combines the embeddings of items and entities that perserve the information from both sources, instead of aligning them with similar positions in the latent space.
- All models preform much better on MovieLens-1m than on DBbook2014 due to the relatively sufficient training data and an easier test (a higher value even using random initializations). Interestingly, the improvement by utilizing KG is larger on dense dataset of MovieLens than that on sparse dataset of DBbook. This goes against our intuitions that the more sparse the dataset is, the more potentials it shall have for absorbing richer knowledge. Thus, we further split the test set according to different sparisity levels of training data, and investigate the impacts from KG on each subset in the next section.

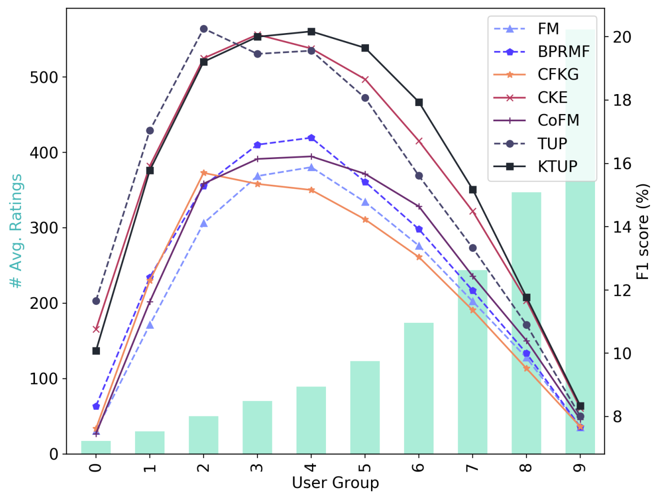


Figure 4: Influence of Different Sparsity on MovieLens-1m. The x-axis shows 10 user groups splited according to interaction number, the left y-axis corresponds to the bars indicating the number of interactions in each user group, and the right y-axis denotes F1-score of curves.

6.4.3 Influence of Training Data Sparsity. To investigate the influence of data sparsity on knowledge transfer, we split the test set of MovieLens-1m into 10 subsets according to the rating number of each user for training, meanwhile try to balance the number of users as well as ratings in each subset. The detailed results of F1 score are shown in Figure 4. Green bars represent the average rating number per user ranging from 17 to 563[5]. We denote the models without KG knowledge as dashed lines, and other models as solid lines.

We can see that (1) KG-based methods (i.e., CKE and KTUP) outperform the other models the most when the average ratings per

[5]Note that the sparisity of DBBook2014 can be concluded into the 0th user group in MovieLens-1m, in which we observe similar results.

Table 3: Performances on MovieLens by Relation Category

Task	Prediction Head (Hits@10, %)				Prediction Tail (Hits@10, %)			
Relation Category	**1-to-1**	**1-to-N**	**N-to-1**	**N-to-N**	**1-to-1**	**1-to-N**	**N-to-1**	**N-to-N**
TransE	59.62	56.76	64.55	24.56	**65.38**	62.16	78.52	46.25
TransH	61.54	48.65	65.73	25.51	57.69	78.38	75.62	46.73
TransR	17.31	29.73	32.88	18.50	17.31	43.24	53.12	38.88
CFKG	59.62	51.35	63.31	20.30	57.69	70.27	78.56	41.22
CKE	19.23	21.62	24.16	14.81	7.69	24.32	37.83	34.82
CoFM (share)	65.38	59.46	66.13	24.42	61.54	72.97	**81.05**	45.99
CoFM (reg)	69.23	**70.27**	66.09	24.30	48.08	**86.49**	80.72	45.79
KTUP (hard)	67.31	59.46	66.42	25.67	57.69	81.08	79.22	47.24
KTUP (soft)	**75.00**	56.76	**67.16**	**26.09**	63.46	81.08	78.34	**47.65**

user ranges from 100 to 200. (2) The gap between two types of models is getting closer along with the training data decreases, and the improvements become similar with that on DBbook when their training data are at the similar sparisity level. (3) Meanwhile, the gap almost disappears when the average ratings are 563 (the left most bar), which implies that the impacts of KG become negligible if there are sufficient training data. Note that the performances of all models are getting worse when the average ratings are larger than 89, the possible reason is the user likes so many items that the preferences are too general to capture. (4) TUP outperforms KTUP when the user preferences are relative simple to model (i.e. # rating < 50), demonstrating the effectiveness and necessity to fully utilize the user-item interactions for preference modeling.

Table 4: Overall Performances on KG Completion

	MovieLens-1m		**DBbook2014**	
	Hit@10 (%)	**Mean Rank**	**Hit@10 (%)**	**Mean Rank**
TransE	46.95	537	60.71	531
TransH	47.63	537	60.06	556
TransR	38.93	609	56.33	563
CFKG	41.56	523	58.83	547
CKE	34.37	585	54.66	593
CoFM (share)	46.62	515	57.01	529
CoFM (reg)	46.51	**506**	60.81	521
KTUP (hard)	48.39	525	60.53	501
KTUP (soft)	**48.90**	527	**60.75**	**499**

6.5 Knowledge Graph Completion

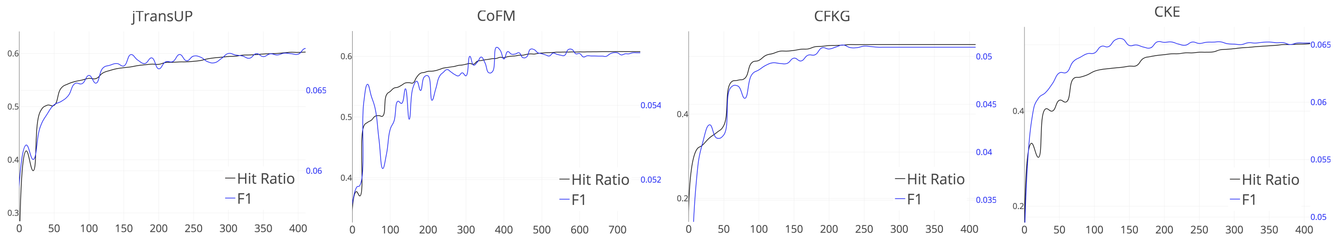
In this section, we evaluate on the task of KG completion. It is to predict the missing entity e_h or e_t for a given triplet (e_h, e_t, r). For each missing entity, we take all entities as candidates and rank them according to the scores computed based on entity and relation embeddings.

6.5.1 Metrics. We use two evaluation metrics that have been widely used in previous work [41]:

- Hit ratio@N: Hit at rank N is 1 if the miss entity are ranked within the top N candidates, otherwise 0. We compute the mean of all triplets as the final hit ratio score.
- Mean Rank : Mean Rank is the averaged rank of the missing entities, the smaller the better.

6.5.2 Overall Results. Table 4 shows the overall performances. We can see that KTUP outperforms all the other models on both datasets. Compared with TransH, it achieves a larger improvement on Hit Ratio for MovieLens-1m than that for DBbook2014 (2.67% v.s. 1.15%), because Movielens-1m contains more connectivities through users and items that are helpful for modeling structural knowledge between entities. Note that the mean rank metric is less important since it is easily reduced by an obstinate triple with a low rank [40]. We also observe that CFKG, CKE and CoFM show a performance drop than their basic KG components: TransE and TransR. One reason may be that these methods force the embeddings of aligned entities to satisfy the other task of item recommendation, while the aligned entities are only a small portion (i.e. 19.95% and 18.25% on two datasets), which actually disturbes the learning for KG completion. Another reason is that the N-to-N issues of user preferences have negative impacts on the representation learning of entities and relations, especially for the *buy* relation in CFKG. CKE takes this issue into account but TransR contains a large amount of trainable parameters and doesn't work well on such a small training set.

6.5.3 Capability to Handle N-to-N Relations. Table 3 shows the separate evaluation results on each relation category. Following [2], we divide relations into four types: 1-to-1, 1-to-N, N-to-1 and N-to-N. We can see that (1) TransR and its related models (i.e., CKE) perform the worst which is consistent with the above overall performances. (2) KTUP achieves the best performances on N-to-N issues, and also performs competitively with TransE and CoFM on 1-to-1, 1-to-N and N-to-1 problems, which indicates the capability of our methods in hanling complex relations and improves both tasks. (3) CFKG presents a lower value on N-to-N relations than TransE, which implies the unifed graph introduced more confusing relational semantics. (4) CoFM performs competitive in KG completion while worse in item recommendation, becuase their knowledge transfering schemes lead to unstable jointly training. That is, it's difficult to control the positive impacts of knowledge transfer on which task, and different parameters for separately training CoFM on each task is required, which is also concluded in the original paper [30].



(a) KTUP ($\rho = 0.97$) (b) CoFM ($\rho = 0.81$) (c) CFKG ($\rho = 0.97$) (d) CKE ($\rho = 0.94$)

Figure 5: Correlation of Training Curves between Two Tasks on DBbook2014, which is denoted by the Pearson's correlation coefficient ρ. The x-axis is training epoch, the left y-axis corresponds to KG completion via hit ratio, and the right y-axis is for item recommendation through F1.

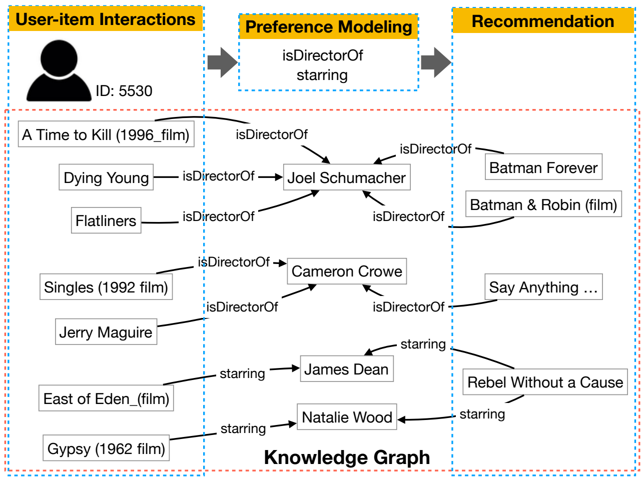


Figure 6: Real Example from MovieLens-1m

6.6 Mutual Benefits of Two Tasks

Although the evalutions on separate tasks have been conducted, it is still unclear that how different transfer schemes take effect. We thus investigate the correlations between the training curves of two tasks. Intuitively, a strong correlation implies more sufficient transfer learning, and better utilization of the complementary information from each other. Because KG completion has no F1 measures, so we present its most important curve of hit ratio corresponding to the left y-axis, and the right y-axis is for item recommendation through F1.

As shown in Figure 5, we can see that KTUP and CFKG present the strongest correlations between their curves, that is, the increase and decrease of one curve shall be reflected on the other curve simultaneously. This implies that the transfer of relations plays an important role in training two tasks together, but CFKG performs not well on both tasks mainly because it cannot deal with complex relations. Another reason is that it only increase the connectivity in the unified graph through the integration of relations and preferences, which are actually not transitional. Instead, KTUP combines the embeddings of relations and preferences that perserve two types of structural knowledge, and meanwhile introduces hyperplanes for the N-to-N issue. The curves of CoFM and CKE are obviously not correlated strongly due to the small portion of transfered entities. Concretely, CoFM forces the embeddings of aligned entities and items to be similar which may result in unstable training. CKE focuses on unidirectional enhancements by combining their embeddings, and thus performs well in item recommendation but worse in KG completion.

6.7 Case Study

In this section, we present a real example of Movielens to give an intuitive impression of our explainability. On the left is a user and his interacted 7 movies. KTUP first induces preference of these movies, and finds what he concerns about is the relations of *isDirectorOf* and *starring* (the preferences having highest attention in Section 4.1). Thus, we search the nearest items that the user is translated to using the induced preferences, and recommend the four movies in the right side. Particularly, *Batman Forever* and *Batman & Robin (film)* are recommended because the user concerns with their director *Joel Schumacher*, who has been captured in existing interactions. Similarly, the preference to director is also induced for the movie *Say Anything ...* of *Cameron Crowe* who becomes attractive due to *Singles (1992 film)* as well as *Jerry Maguire*. Besides, the user has various preferences like *starring*, such as *James Dean* in *East of Eden (film)* and *Natalie Wood* in *Gypsy (1962 film)* suggests another movie *Rebel Without a Cause*.

7 CONCLUSION

In this paper, we propose a novel translation-based recommender model, TUP, and extend it to integrate KG completion seamlessly, namely KTUP. TUP is capable of modeling various implicit relations between users and items which reveal the preferences of users on consuming items. KTUP further enhances the model explainability via aligned relations and perferences, and improves the preformances of both tasks via mutual transfering. In future, we are interested in inducing more complex user preferences over multi-hop entity relations and introducing KG reasoning (e.g., rule mining) techniques for unseen user preferences to deal with cold start problem.

ACKNOWLEDGMENTS

This research is part of NExT research which is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

REFERENCES

[1] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* (2014).

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

[3] Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural Collective Entity Linking. In *COLING*.

[4] Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Chengjiang Li, Xu Chen, and Tiansi Dong. 2018. Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision. In *EMNLP*.

[5] Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*.

[6] Rose Catherine and William Cohen. 2016. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *RecSys*.

[7] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*.

[8] Xu Chen, Yongfeng Zhang, Hongteng Xu, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Visually Explainable Recommendation. *arXiv preprint arXiv:1801.10288* (2018).

[9] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. In *WWW*.

[10] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal Relational Ranking for Stock Prediction. *ACM Transactions on Information Systems* (2019).

[11] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *ACL*.

[12] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*.

[13] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*.

[14] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[16] Benjamin Heitmann. 2012. An open framework for multi-source, cross-domain personalisation with semantic interest graphs. In *RecSys*.

[17] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*.

[18] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR* (2017).

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009).

[20] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* (2015).

[21] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *EMNLP*.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *AAAI*.

[23] Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704* (2016).

[24] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.

[25] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic Embeddings of Knowledge Graphs.. In *AAAI*.

[26] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *WWW*.

[27] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 2016. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2016).

[28] Guangyuan Piao and John G Breslin. 2017. Factorization machines leveraging lightweight linked open data-enabled features for top-N recommendations. In *WISE*.

[29] Guangyuan Piao and John G. Breslin. 2018. A Study of the Similarities of Entity Embeddings Learned from Different Aspects of a Knowledge Base for Item Recommendations. In *ESWC*.

[30] Guangyuan Piao and John G. Breslin. 2018. Transfer Learning for Item Recommendations and Knowledge Graph Completion in Item Related Domains via a Co-Factorization Model. In *ESWC*.

[31] Tony A Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural networks* (1995).

[32] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[34] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.

[35] Amit Sharma and Dan Cosley. 2013. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *WWW*.

[36] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*.

[37] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

[38] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*.

[39] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*.

[40] Zhigang Wang and Juan-Zi Li. 2016. Text-Enhanced Representation Learning for Knowledge Graph.. In *IJCAI*. 1293–1299.

[41] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*.

[42] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

[43] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *SIGKDD*.

[44] Mengdi Zhang, Tao Huang, Yixin Cao, and Lei Hou. 2015. Target Detection and Knowledge Learning for Domain Restricted Question Answering. In *NLPCC*.

[45] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over Knowledge-Base Embeddings for Recommendation. In *SIGIR*.

[46] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*.

[47] Zili Zhou, Shaowu Liu, Guandong Xu, Xing Xie, Jun Yin, Yidong Li, and Wu Zhang. 2018. Knowledge-Based Recommendation with Hierarchical Collaborative Embedding. In *PAKDD*.