

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 and simply transfer 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 infer that the director relation plays a critical role when the user makes the decision, thus help to understand the user's preference at 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>.

CCS CONCEPTS

• Information systems → Personalization.

KEYWORDS

Item Recommendation; Knowledge Graph; Embedding; Joint Model;

*Corresponding author.

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.

ACM ISBN 978-1-4503-6674-8/19/05.

<https://doi.org/10.1145/3308558.3313705>

ACM Reference Format:

Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *Proceedings of the 2019 World Wide Web Conference (WWW '19)*, May 13–17, 2019, San Francisco, CA, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3308558.3313705>

1 INTRODUCTION

Knowledge Graph (KG) is a heterogeneous structure that stores the world's knowledge in the form of machine-readable graphs, where nodes denote entities, and edges denote the relations between entities. Since proposed, KG has attracted much attention in many fields, ranging from recommendation [40], dialogue system [18, 21], to information extraction [3]. Focusing on recommendation, the structural knowledge has shown great potential in providing rich information about the items, offers a promising solution to improving 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 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 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 [16, 29, 30, 44, 46, 48], there are few works that jointly

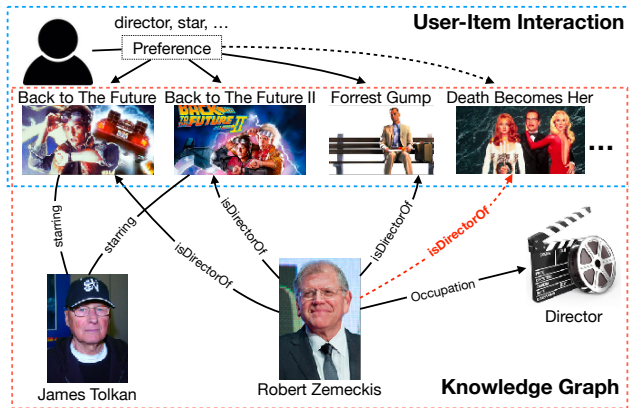


Figure 1: An illustrative example on the necessity of considering the missing relations in KG for recommendation.

model the two tasks of knowledge graph learning and recommendation. CoFM [31] is the most similar work that aligns the 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 fails to offer interpretation ability.

In this work, we propose a Translation-based User Preference model (TUP) to integrate with KG learning seamlessly. The key idea is that there exists 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 watch the 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 deeper understanding. 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 Knowledge-enhanced TUP (KTUP) that jointly learns the representations of users, items, entities, and relations. The main contributions are summarized:

- 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, thus empowering the model with explainability.
- We perform extensive experiments on two datasets for top-N recommendation and KG completion tasks, which verify 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 related work for each task, before discussing 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) [35], factorization machines [33], matrix factorization techniques [19], BPRMF [34]. The key challenge here 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, 12–14]. 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 the contextual reviews [9, 25], relational data [10, 36] and knowledge graphs [6]. Another advantage of additional contents is the improved explainable ability to understand why an item is being recommended out of others. This has been found to be important for the effectiveness, efficiency, persuasiveness, and user satisfaction of recommender systems [8, 39, 47].

Among the side information, knowledge graphs (e.g., DBPedia [20]) show great potentials on recommendations due to its well-defined structures and adequate 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 [29] extracted lightweight features derived from KG (i.e. property-object, subject-property) for factorization machines. Zhang et al. [46] 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 the second group usually improve the quality of item embeddings using entity embeddings if they refer to the same thing [16, 44]. Piao and Breslin [30] summarized the recommendation results using different entity embeddings (i.e. node2vec, doc2vec and transE), and found that node2vec improves the most. CoFM [31] first takes into account the refinements of entity embeddings from user-item modeling as another transferring task. However, the above methods heavily rely on the alignments between items and entities. Zhou et al. [48] introduced entity concepts in KG to deal with the sparsity issue of the alignments, but still fail to consider the importance of entity relations in transferring 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 [11], or modeled implicitly via Memory-based Attention [37]. We thus improve this type of method by considering the N-to-N issue¹ in modeling user preferences as translational relations, which shall be further enhanced by transferring knowledge of entities and their relations from KG.

¹N-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 found to be effective in many tasks of natural language processing, such as the question answering [45], which accelerates the popularity of KGs. Although there are a host of methods for finding entities [4, 5] and their relations from texts [22], existing KGs are far from complete. Recent studies for KG completion show a great enthusiasm for learning low-dimensional representations 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, it is sometimes confusing because some relations can translate one entity to various entities, namely the 1-to-N problem. Similarly, there are other N-to-1 and N-to-N problems. To address these problems, a host of methods extended TransE by introducing additional hyperplanes [42], vector spaces [23], textual information [41] and relational path [22].

The second group measures the plausibility of facts by matching semantic representations of entities and relations through similarity-based scoring functions. RESCAL [27] represents each relation as a matrix to capture the compositional semantics between entities, and utilizes a bilinear function as similarity metrics. To simplify the learning of relation matrices, DistMult [43] restricts them to be diagonal, HoE [26] defines a circular correlation [32] to compress relation matrices as vectors, and ComplEx [38] introduces complex-value for asymmetric relations. Instead of modeling the compositional relation, another line of methods directly introduce NNs for matching. SME [1] learns relation specific layers for head entity and tail entity, respectively, and then feeds them into the final matching layer (e.g., dot production), while NAM [24] conducts the 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 to transfer knowledge between fields. 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 both aim to rank candidates given a query (i.e., an entity or a user) as well as their implicit or explicit relatedness. For example, KG completion aims to find correct movies (e.g., *Death Becomes Her*) for the person *Robert Zemeckis* given the explicit relation *isDirectorOf*, while item recommendation aims at recommending movies for a target user satisfying some implicit preference. Therefore, we are able to fill in the gap between item recommendation and KG completion via a joint model, to 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 [42] 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 recommending 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 of 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 [42].

Therefore, TransH [42] 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 same hyperplane. It defines an energy score function for a triplet as follows:

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

where a lower score of $f(e_h, e_t, r)$ indicates that the triplet is possibly true, otherwise no. \mathbf{e}_h^\perp and \mathbf{e}_t^\perp are projected entity vectors:

$$\mathbf{e}_h^\perp = \mathbf{e}_h - \mathbf{w}_r^T \mathbf{e}_h \mathbf{w}_r \quad (2)$$

$$\mathbf{e}_t^\perp = \mathbf{e}_t - \mathbf{w}_r^T \mathbf{e}_t \mathbf{w}_r \quad (3)$$

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:

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

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 explicitly 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 implicit 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 commonality 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 [17] for discretely sampling the preference given a user-item pair, which utilizes reparameterization trick for backpropagation, making it possible to calculate the continuous gradients of model parameters during the 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:

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

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

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

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 according to a continuous probability distribution $\mathbf{y} = [y_1, \dots, y_p, \dots, y_P]$:

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

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

Straight-Through (ST) gumbel-Softmax takes different paths in the forward and backward propagation, so as to maintain sparsity yet support stochastic gradient 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 be backpropagated.

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

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

We use dot product as similarity function.

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:

$$\mathbf{p} = \sum_{p' \in \mathcal{P}} \alpha_{p'} \mathbf{p}' \quad (9)$$

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

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

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 who likes both of them are due to the some preference (Figure 2(a)),

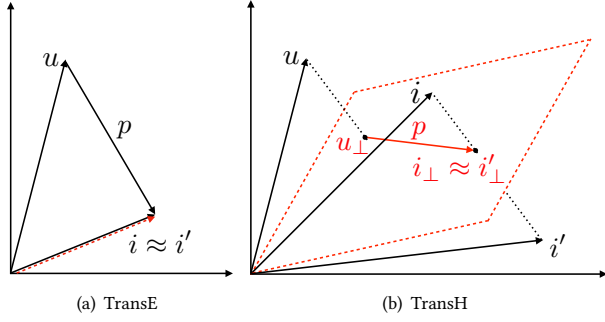


Figure 2: Illustration of the two translation schemes for item recommendation

leading to an incorrect conclusion that a user consuming one shall consume the other, no matter what the user’s preference is.

Such limitations are alleviated by introducing preference 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:

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

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:

$$\mathbf{u}^\perp = \mathbf{u} - \mathbf{w}_p^T \mathbf{u} \mathbf{w}_p \quad (12)$$

$$\mathbf{i}^\perp = \mathbf{i} - \mathbf{w}_p^T \mathbf{i} \mathbf{w}_p \quad (13)$$

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:

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

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

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

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

Traditional methods (e.g., BPRMF [34]) 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 commonality among users as global latent features, compared with user embeddings which only concerns with the local features of the user alone, and (2) reflect richer semantics for explainable ability.

5 JOINT LEARNING VIA KTUP FOR TWO TASKS

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 commonality 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. On 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, while 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:

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

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

$$\hat{\mathbf{i}}^\perp = \hat{\mathbf{i}} - \hat{\mathbf{w}}_p^T \hat{\mathbf{i}} \hat{\mathbf{w}}_p \quad (17)$$

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

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 $\mathcal{R} \rightarrow \mathcal{P}$. We obtain these two vectors as follows:

$$\hat{\mathbf{p}} = \mathbf{p} + \mathbf{r} \quad (19)$$

$$\hat{\mathbf{w}}_p = \mathbf{w}_p + \mathbf{w}_r \quad (20)$$

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} preserves the structural knowledge in KG. Meanwhile, the entity embedding \mathbf{e} shall be fine tuned by the additional connectivity through users and items during back-propagation. 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 the 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 knowledge of entities. Considering that each user-item pair

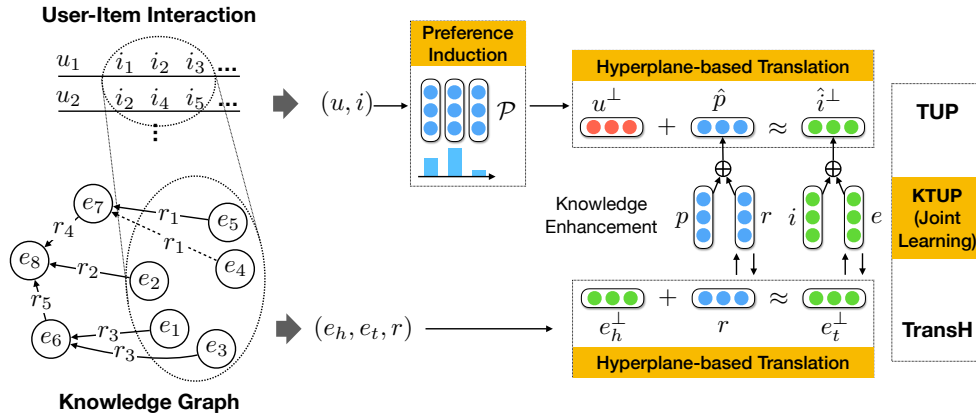


Figure 3: Framework of KTUP. At the top is TUP for item recommendation including two components: preference induction and hyperplane-based translation. KTUP jointly learns TUP and TransH to enhance the item and preference modeling by transferring knowledge of entities as well as relations.

has a preference and so does the relations between two entities, KTUP optimizes all users, items and entities more thoroughly.

5.2 Training

We train KTUP using the overall objective function as follows:

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

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 KG-based recommendation methods to facilitate a deep understanding between two tasks in Section 6. We choose three typical models that transfer knowledge of entities at data level (CFKG [46]), at embedding level (CKE [44]) and in both directions (CoFM [31]). We summarize the main differences and similarities from the following aspects:

Implicitly of User Preference CKE and CoFM can be regarded as extensions 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 explicitly 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. KTUP distinguishes different user preferences and introduces hyperplanes for each preference as well as each relation to learn the various representations of items and entities.

Transferred Knowledge from KG CKE and CoFM only focus on transferring knowledge of entities. CFKG also transfers relations in the way of data integration through a unified graph. Except for entities and items, KTUP combines the embeddings of relations and preferences according to the 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 joint training. Finally, we highlight real examples for qualitative analysis to give an intuitive impression of explainability. We publish our project at <https://github.com/TaoMiner/joint-kg-recommender>.

6.1 Datasets

Following CoFM [31], we use two publicly available datasets in the movie and book domains: MovieLens-1m [28] and DBbook2014². Both datasets consist of users and their ratings on movies or books, which are then refined for LODRecSys [15, 28, 29] by mapping items into DBpedia entities if there is a mapping available. Following most item recommendation work that models implicit feedback [40], 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 datasets³. After preprocessing, there are 6,040 users and 3,230 items with 998,539 ratings in MovieLens-1m, the average number of ratings per user is 165 and the sparsity rate is 94.9%. The data sparsity

²<http://2014.eswc-conferences.org/important-dates/call-RecSys.html>

³Because 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 [31]).

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%

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 sparsity rate reaches up to 99.6%. The triplets used in the 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, while the subgraph of DBbook has 334,511 triplets with 13,882 entities and 13 relations. Note that the alignments between items and entities for transferring 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** [33] and **BPRMF** [34], because they are the foundations of other baselines and also achieve the state-of-the-art performance on many benchmark datasets.
- **CFKG** [46] that integrates the data of two sources and applies TransE on a unified graph including users, items, entities and relations;
- **CKE** [44] that combines various item embeddings from different sources including TransR on KG;
- **CoFM** [31] that jointly trains 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** [42] and **TransR** [23] that are widely used in this field. Also, we evaluate the above KG-based methods even if they have not been done so 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 did not 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 are out of the scope of this paper.

6.3 Training Details

We construct training set, validation set and test set by randomly splitting 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 the best settings for each task, and use them for all of the other models since they share the basic learning ideas⁴. 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 the 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 hyperparameters are empirically set as follows: the batch size is 256, the embedding size is 100 and we perform early stopping strategy on the 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 the two datasets after searching in {0.7, 0.5, 0.3}, so as to balance their impacts, and use the 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 the 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 the recommended items.

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

- **Precision@N:** It 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:** It is the proportion of items relevant to the user that have been successfully recommended. We compute the mean of all users as the final recall.
- **F1 score@N:** It is the harmonic mean of precision at rank N and recall at rank N.
- **Hit ratio@N:** It 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.

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

Table 2: Overall performance 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

- nDCG@N: Normalized Discounted Cumulative Gain (nDCG) is a standard measure of ranking quality, considering the 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 performance 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 observe that:

- Our proposed methods perform the best compared with the baseline methods on the two 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 the user-item interactions, and performs much better especially when the amount of interaction data is sufficient, like MovieLens-1m. By incorporating KG, KTUP further 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 preference 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 the aligned items, leading to the loss of knowledge that has been perserved in the embeddings, and the loss becomes more serious when there is insufficient training data.
- CKE achieves pretty good performance on the 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 the sparse

dataset of DBbook. This goes against our intuitions that the more sparse the dataset is, the more potentials it shall have in absorbing richer knowledge. Thus, we further split the test set according to different sparsity levels of training data, and investigate the impacts from KG on each subset in the next section.

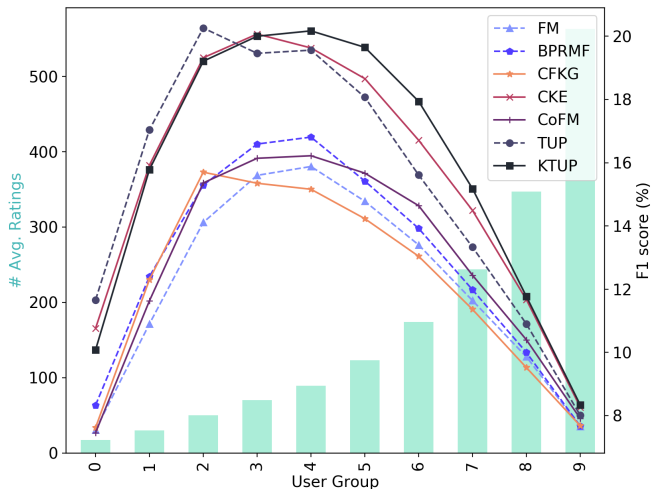


Figure 4: Influence of Different Sparsity on MovieLens-1m. The x-axis shows 10 user groups splitted according to interaction number, the left y-axis corresponds to the bars indicating the number of interactions in each user group, and the right y-axis denotes F1-score of curves.

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 we also 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 563⁵. We denote the models without KG knowledge as dashed lines, and other models as solid lines.

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

Table 3: Performance on MovieLens by Relation Category

Task Relation Category	Prediction Head (Hits@10, %)				Prediction Tail (Hits@10, %)			
	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

We can see that (1) KG-based methods (i.e., CKE and KTUP) outperform the other models the most when the average number of ratings per user ranges from 100 to 200. (2) The gap between the two types of models is getting closer as the amount of the training data decreases, and the improvements become similar with that on DBbook when their training data is at the similar sparsity 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 is sufficient training data. Note that the performance 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 user preferences are relative simple to model (i.e., #rating<50), showing the effectiveness and necessity to fully utilize user-item interactions for preference modeling.

Table 4: Overall performance 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 [42]:

- Hit ratio@N: It is 1 if the miss entity is ranked within the top N candidates, otherwise 0. We compute the mean of all triplets as the final hit ratio score.

- Mean Rank : It is the averaged rank of the missing entities, the smaller the better.

6.5.2 Overall Results. Table 4 shows the overall performance. We can see that KTUP almost outperforms all the other models on both datasets except the mean rank value on MovieLens-1m. We argue this metric is less important since it is easily reduced by an obstinate triple with a low rank [41]. Compared with TransH, it achieves a larger improvement on Hit Ratio for MovieLens-1m as compared to that for DBbook2014 (2.67% v.s. 1.15%), because MovieLens-1m contains more connectivities between users and items that are helpful for modeling structural knowledge between entities. We also observe that CFKG, CKE and CoFM show a performance drop as compared to 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 the two datasets), which actually degrades 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 lot of trainable parameters and does not 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 performance. (2) KTUP achieves the best performance 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 handling complex relations and improves both tasks. (3) CFKG presents a lower value on N-to-N relations than TransE, which implies that the unified graph may have introduced more confusing relational semantics. (4) CoFM performs competitively in KG completion task while worse in item recommendation, because their knowledge transferring schemes lead to unstable joint training. That is, it is 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 [31].

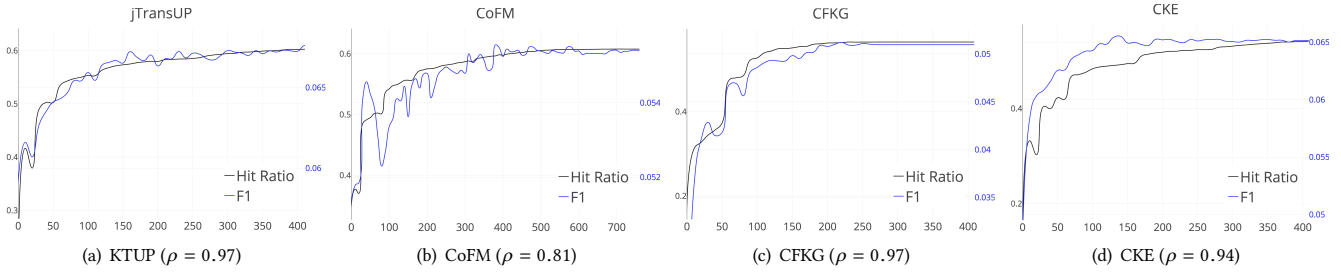


Figure 5: Correlation of Training Curves between Two Tasks on DBbook2014, which is denoted by the Pearson’s correlation coefficient ρ . The x-axis is training epoch, the left y-axis corresponds to KG completion via hit ratio, and the right y-axis is for item recommendation through F1. (Note that we scale the values of both F1 and Hit Ratio to the same magnitude.)

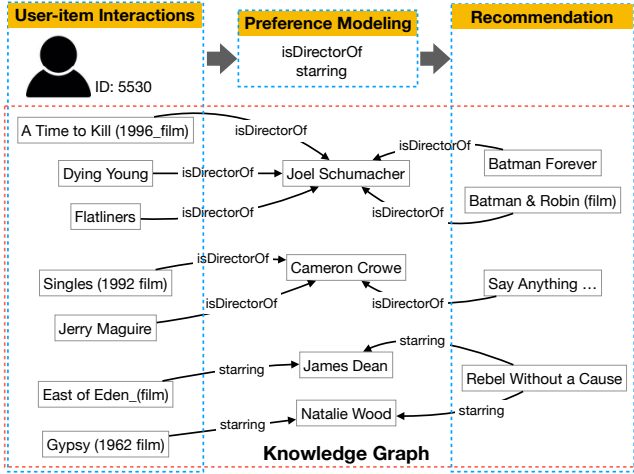


Figure 6: Real Example from MovieLens-1m

6.6 Mutual Benefits of Two Tasks

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

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 the two tasks together. However, CFKG does not perform well on both tasks (as shown in Table 2 and Table 4) mainly because of 2 reasons. First, it cannot deal with complex relations; second, 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 preserve 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 transferred 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 an example of MovieLens-1m to give an intuitive impression of our explainability. On the left is a user who has interacted with 7 movies. KTUP first induces user preferences to these movies, and finds that what the user is concerned is the relations of *isDirectorOf* and *starring* (the preferences having highest attention in Section 4.1). Thus, it searches the nearest items according to Equation 16 based on the induced preferences. We present the recommended four movies on the right side. In particular, *Batman Forever* and *Batman & Robin (film)* are recommended because the user shows preference with their director *Joel Schumacher*. Similarly, the preference to director also helps to induce the movie *Say Anything ...* directed by *Cameron Crowe*. Besides, the user also has preferences on *starring*, such as *James Dean* in *East of Eden (film)* and *Natalie Wood* in *Gypsy (1962 film)*; together, the system suggests another movie *Rebel Without a Cause*.

7 CONCLUSION

In this paper, we proposed a novel translation-based recommender model, TUP, and extended 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 preferences, and improves the performances of both tasks via joint learning.

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 the 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, Jiayi 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] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*.
- [12] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*.
- [13] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*.
- [14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.
- [15] Benjamin Heitmann. 2012. An open framework for multi-source, cross-domain personalisation with semantic interest graphs. In *RecSys*.
- [16] 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*.
- [17] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR* (2017).
- [18] Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Zhao, and Dawei Yin. 2018. Explicit State Tracking with Semi-Supervised Neural Dialogue Generation. In *CIKM*.
- [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] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures. In *ACL*.
- [22] 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*.
- [23] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.
- [24] 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).
- [25] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.
- [26] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic Embeddings of Knowledge Graphs. In *AAAI*.
- [27] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *WWW*.
- [28] 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).
- [29] Guangyuan Piao and John G Breslin. 2017. Factorization machines leveraging lightweight linked open data-enabled features for top-N recommendations. In *WISE*.
- [30] 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*.
- [31] 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*.
- [32] Tony A Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural networks* (1995).
- [33] Steffen Rendle. 2010. Factorization machines. In *ICDM*.
- [34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.
- [35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.
- [36] Amit Sharma and Dan Cosley. 2013. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *WWW*.
- [37] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*.
- [38] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.
- [39] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*.
- [40] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*.
- [41] Zhigang Wang and Juan-Zi Li. 2016. Text-Enhanced Representation Learning for Knowledge Graph. In *IJCAL* 1293–1299.
- [42] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.
- [43] 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*.
- [44] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *SIGKDD*.
- [45] Mengdi Zhang, Tao Huang, Yixin Cao, and Lei Hou. 2015. Target Detection and Knowledge Learning for Domain Restricted Question Answering. In *NLPCC*.
- [46] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over Knowledge-Base Embeddings for Recommendation. In *SIGIR*.
- [47] 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*.
- [48] 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*.