AKUPM: Attention-Enhanced Knowledge-Aware User Preference Model for Recommendation

Xiaoli Tang, Tengyun Wang, Haizhi Yang, Hengjie Song*
South China University of Technology
Guangzhou, China
{semiertang,setywang,sehzyang}@mail.scut.edu.cn,sehjsong@scut.edu.cn

ABSTRACT

Recently, much attention has been paid to the usage of knowledge graph within the context of recommender systems to alleviate the data sparsity and cold-start problems. However, when incorporating entities from a knowledge graph to represent users, most existing works are unaware of the relationships between these entities and users. As a result, the recommendation results may suffer a lot from some unrelated entities. In this paper, we investigate how to explore these relationships which are essentially determined by the interactions among entities. Firstly, we categorize the interactions among entities into two types: *inter-entity-interaction* and *intra-entity-interaction*. *Inter-entity-interaction* is the interactions among entities that affect their importances to represent users. And *intra-entity-interaction* is the interactions within an entity that describe the different characteristics of this entity when involved in different relations. Then, considering these two types of interactions, we propose a novel model named ***A**ttention-enhanced **K**nowledge-aware **U**ser **P**reference **M**odel* (AKUPM) for click-through rate (CTR) prediction. More specifically, a self-attention network is utilized to capture the *inter-entity-interaction* by learning appropriate importance of each entity w.r.t the user. Moreover, the *intra-entity-interaction* is modeled by projecting each entity into its connected relation spaces to obtain the suitable characteristics. By doing so, AKUPM is able to figure out the most related part of incorporated entities (i.e., filter out the unrelated entities). Extensive experiments on two real-world public datasets demonstrate that AKUPM achieves substantial gains in terms of common evaluation metrics (e.g., AUC, ACC and Recall@top-K) over several state-of-the-art baselines.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Collaborative filtering*.

KEYWORDS

recommender system; knowledge graph; attention mechanism;

*Hengjie Song is the corresponding author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330705

ACM Reference Format:
Xiaoli Tang, Tengyun Wang, Haizhi Yang, Hengjie Song. 2019. AKUPM: Attention-Enhanced Knowledge-Aware User Preference Model for Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330705

1 INTRODUCTION

As item recommendation is one of the most effective and widely used information filtering and discovery method, extensive research efforts have been devoted to Recommendation Systems (RS) which aim at providing users with a small set of items to meet the personalized interests [12]. Among different recommendation strategies, collaborative filtering (CF) based methods which make use of historical user preferences have made significant success [7]. However, CF based methods generate poor performance when user-item interactions are sparse. A common idea to alleviate these problems is to import auxiliary information to enrich the description of users and items within the RS, so as to effectively compensate for the sparse or the missing users' interests. Among various types of auxiliary information (e.g., user profiles [15], images [21] and so on), Knowledge Graph (KG) as an emerging one has gradually gained attention of researchers in recent years because of the ability to support a bulk of structured data effectively [18].

A knowledge graph is a type of multi-relational directed graph composed of a large number of entities and relations [18]. More specifically, each edge in the knowledge graph is represented as a triple in the form of *(head entity, relation, tail entity)*, also called a *fact*, indicating that the *head entity* and the *tail entity* is connected through the *relation*. For example, as shown on the top of Figure 1, triple *(City Light, film.director, Charles Chaplin)* indicates that *Charles Chaplin* is the director of film *City Light*. Recently, a large number of knowledge graphs, such as DBpedia [1] and Microsoft's Satori [2], have been created and successfully applied to many real-world applications, e.g., information extraction [4], community detection [13] and so on.

Over the past years, many approaches [16, 17, 21] have been studied and designed to enrich the sparse information of the users/items in RS by incorporating a set of related entities from a knowledge graph. For example, Deep Knowledge-Aware Network [16] is a convolutional neural network (CNN) based framework to make news recommendation in which the information of news is enriched by associated entities in the knowledge graph. RippleNet [17] aims at item recommendation in which users' interests are enriched by entities they have interacted with. However, in these existing

[1] http://wiki.dbpedia.org/
[2] http://searchengineland.com/library/bing/bing-satori

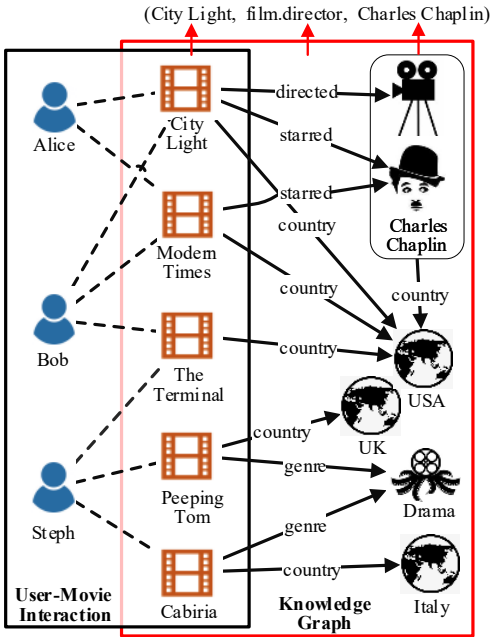

Figure 1: Illustration of knowledge graph enhanced movie recommendation system. The knowledge graph provides amounts of entities which can be used to enrich the sparse user-movie interactions.

approaches, the relationships between the set of incorporated entities and items/users are not explored so that the recommendation results may suffer a lot from some unrelated incorporated entities.

Take movie recommendation for example, where each user is represented by incorporating entities from a knowledge graph. Since the user is represented by entities, the interactions among these entities essentially determine the relationships between the user and these entities. Interactions among these entities are two-fold: 1) *Inter-entity-interaction*: the importance of an entity varies a lot when included in different entity sets due to the interactions among entities. As shown in Figure 1, with regard to Bob, all incorporated movies are from *USA*, whereas to Steph the incorporated movies are from various countries. In this case, *USA* is of great importance to identify Bos's interests rather than to identify Steph's interests. 2) *Intra-entity-interaction*: for a certain user, an entity may show different characteristics when involved in different relations. For example in Figure 1, Alice may like *City Lights* since *Charles Chaplin* is the director of this film, whereas she may like *Modern Time* since *Charles Chaplin* acts the leading role in this film.

To face these issues, we propose a novel recommendation model named ***A**ttention-enhanced **K**nowledge-aware **U**ser **P**reference **M**odel (AKUPM)*. AKUPM is designed to predict click-through rate (CTR) for a given user-item pair, in which information of the user is enriched by entities related to the user's clicked items. More specifically, firstly, entities to be incorporated are initialized as the user's clicked items, and then propagate along relations in the knowledge graph from near to distant to introduce rich entities. By doing so, AKUPM reaches a great deal of entities that can be used to infer the user's potential interests. Then, to capture the *intra-entity-interaction*, each entity is represented by being projected into its connected relation spaces, so as to distinguish characteristics involved in different relations. Moreover, to depict the *inter-entity-interaction*, a self-attention network [14] is utilized to learn appropriate importance of each incorporated entity. Finally, to characterize user's diverse interests on different items, relationships between the input item and incorporated entities are also explored through an attention network [1], so as to make better CTR predictions. In contrast to existing approaches [16, 17, 21], our contributions can be summarized as follows:

(1) We propose a knowledge-aware model to alleviate the sparsity problem in recommendation systems based on a novel adaptive self-attention modeling design. To the best of our knowledge, this is the first work fully exploring the relationships between users and incorporated entities based on categorizing the interactions of entities into two types: *intra-entity-interaction* and *inter-entity-interaction*.
(2) By projecting entities into relation spaces and utilizing self-attention mechanism, the *intra-entity-interaction* and *inter-entity-interaction* are naturally depicted, respectively. By doing so, AKUPM is able to figure out the most related part of the incorporated entities for each user, so as to make better CTR predictions.

Based on two real-world public recommendation datasets, we have conducted extensive experiments to evaluate the effectiveness of AKUPM. The results reveal that the improvements are significant compared with other state-of-the-art methods in terms of common evaluation metrics (e.g., AUC, ACC and Recall@top-K).

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 provides some preliminary concepts and presents our recommendation problem. Section 4 delves into our proposed approach, followed by learning algorithm in Section 5. Then, the experimental results on benchmark datasets and discussions are provided in Section 6. Finally we list concluding remarks and discuss the future work in Section 7.

2 RELATED WORK

In this section, we present several concepts and models related to our works, including knowledge graph embedding, attention mechanism and knowledge-aware recommendation.

2.1 Knowledge Graph Embedding

Knowledge graph embedding aims at embedding entities and relations of a knowledge graph into low-dimensional continuous vector spaces in which the inherent structure of the knowledge graph is preserved. Recently, due to the superior performance, translation-based embedding methods have received great attention, in which entities are usually represented as vectors and relations are typically abstracted as operations on the entity vectors. As the most representative translation-based method, TransE [2] represents both entities and relations as vectors in the same space so that the embedded entities $\mathbf{h}$ and $\mathbf{t}$ can be connected by relation $\mathbf{r}$ when triple (h, r, t) holds, i.e., $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Despite its simplicity and efficiency, TransE is unable to model *intra-entity-interactions* in our scenario since TransE has difficulty in dealing with 1-to-N relations [20]. To this

end, TransH [20] allows an entity to have distinct representations when involved in different relations by introducing relation-specific hyperplanes. Moreover, TransR [8] first projects entities from entity space to corresponding relation spaces and then builds translations between projected entities. In this paper, to facilitate appropriate characteristics of an entity when involved in different relations, each entity is represented by applying TransR.

2.2 Attention Mechanism

Attention mechanism shares the same intuition with the visual attention found in humans, which assigns distinct importance to different parts of data to allow focusing on the most important data. Attention mechanism is first introduced by [1] to handle sequence to sequence translation task based on an encoder-decoder framework, in which the encoder encoded source sequence into a vector and a decoder then output the translated sequence from the encoded vector. By introducing attention mechanism, the decoder could concentrate on the most relevant part of the source sequence instead of treating all parts of source sequence equally when performing translation.

Different from attention mechanism trying to relate two sequences, self-attention focuses on relating different parts of a single sequence in order to compute a representation of the sequence within its own context. Self-attention has started to gain exposure due to its recent successful application on machine translation [14]. [9] proposed a self-attention based LSTM (Long Short-Term Memory) to generate sentence embeddings, in which the self-attention is utilized to determine a linear combination of LSTM hidden states rather than just pick up the final state of LSTM as embedding output.

In this paper, we propose to use attention mechanism to explore relationships between the input item and incorporated entities. In addition, self-attention mechanism is introduced to generate representations of incorporated entities under associated entity set by assigning each entity an appropriate weight.

2.3 Knowledge-Aware Recommendation

The usage of knowledge graph within the context of recommender systems to address the sparsity and cold-start problems are attracting increasing attention. For example, in [21], the items' structural representations learned from a knowledge graph were combined with items' textual representations and visual representations to jointly represent the items. And then, item recommendations were given by collaborative filtering based on the items' representations. To make news recommendation, [16] proposed to represent news by fusing the representation learned from a knowledge graph and representation obtained by word-level embeddings. More recently, in [17], users' potential interests were explored by propagating users' historical clicked items along the links (relations) in the knowledge graph, so as to enrich users' interests and to improve recommendation performance.

However, in these existing approaches, the relationships between incorporated entities and items/users are not well explored. As a consequence, the recommendation results may suffer a lot from some unrelated entities.

3 PRELIMINARIES

In this section, we will first clarify some terminologies used in this paper, and then explicitly formulate our problem.

3.1 Implicit Feedback

Compared to explicit feedback, the recommendation task considered in this paper is for more abundant implicit feedback which indirectly reflects opinion through observed user behavior [5]. In a typical recommender system, user set and item set are denoted by $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$ and $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$ respectively. The user-item interaction (i.e., users' click history) matrix is denoted as $Y = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\}$, where

$$y_{uv} = \begin{cases} 1, & \text{if } u \text{ has clicked } v; \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

denotes whether user u has clicked item v. Note that a value 1 in Y represents that interaction between the user and corresponding item has been observed. Although such interactions do not assure that users actually like the items, they are source of information on what the users might like. Likewise, a value 0 in Y does not mean that the user is not interested in the item [21]. In this paper, the observed interactions (i.e., 1s in Y) are regarded as positive instances to model users' potential interests, whereas the unobserved interactions (i.e., 0s in Y) are regarded as negative ones.

3.2 Knowledge Graph

Apart from the user-item interaction matrix Y, a knowledge graph $\mathcal{G}$, which consists of massive entity-relation-entity triples (h, r, t), is also a good source to provide extra information about users' interests. Let entity set and relation set are denoted by ξ and $\mathcal{R}$, respectively. For each triple (h, r, t), $h \in \xi$, $t \in \xi$ and $r \in \mathcal{R}$ denote the head entity, tail entity, and relation between these two entities, respectively.

In this paper, to make the problem clearly, we assume that every item $v \in \mathcal{V}$ in interaction matrix Y can be linked to a corresponding entity $h_v \in \xi$ in knowledge graph $\mathcal{G}$ by the technique of entity linking [10].

3.3 Problem Formulation

In this paper, our goal is to learn a CTR prediction model to predict the probability $\hat{y}_{uv}$ that user u will click item v for each input user-item pair (u, v). To achieve the goal, we use a latent vector $\mathbf{u}$ as the representation for the input user u, and a latent vector $\mathbf{v}$ as the representation for the input item v. Then, since the user's click history y_{uv} is binary, we formulate the CTR prediction as binary classification and utilize a sigmoid function $\sigma(\cdot)$ as activation function as follows:

$$\hat{y}_{uv} = \sigma(\mathbf{u}^T \mathbf{v}). \tag{2}$$

4 THE PROPOSED METHOD

In this section, we give an overview of the proposed AKUPM followed by detailed descriptions of main components.

We introduce the architecture of AKUPM as illustrated in Figure 2 from the bottom up. For the input user, to incorporate rich entities from a knowledge graph potentially exhibiting his interests,

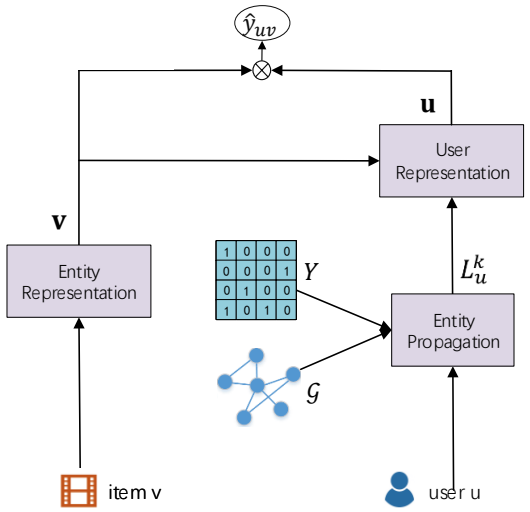

Figure 2: Illustration of AKUPM.

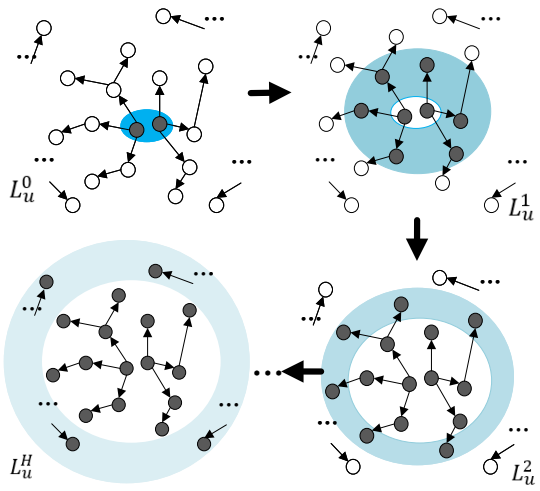

Figure 3: Illustration of entity propagation. All the grey circles denote the incorporated entities. And entities falling into different blues belong to different corresponding entity set.

a specially designed entity propagation (Section 4.1) is used to incorporate sets of entities based on the user's click history. Next, to depict the *intra-entity-interactions* among these incorporated entities, we propose to embed the entities into relation spaces (Section 4.2). To model the *inter-entity-interactions* (Section 4.3), we propose to first apply a self-attention network to obtain a representation for each set of entities. And then, we propose to apply an attention network to represent the user with respect to the input item by aggregating entity representations with proper weights. Finally, CTR prediction is made based on the latent representations of the input user and input item.

4.1 Entity Propagation

A knowledge graph carries various entities and complicated connections among entities that provide a latent perspective to explore user interests. However, for a certain user, not all entities share same importance to recognize user's interests and there are many entities unrelated to this user (called *noises*). To filter out noises, we propose to incorporate entities propagated from the user's click history along relations in knowledge graph, so that each incorporated entity is relevant to the user.

More specifically, for a given user u, we denote his click history as $S_u = \{v_{u,1}, \cdots, v_{u,m}, \cdots, v_{u,|S|}\}$, where $v_{u,m}$ is the identification of m^{th} item clicked by user u. The origin of entity propagation is defined as

$$L_u^0 = \{h^0_{v_{u,1}}, \cdots, h^0_{v_{u,m}} \cdots, h^0_{v_{u,|S|}}\}, \tag{3}$$

where $h^0_{v_{u,m}}$ denotes the entity linked to item $v_{u,m}$. Then, entities in L^0 can be iteratively propagated along relations to reach more connected entities and we define the k^{th} set of entities as follows:

$$L_u^k = \{t^k | (h^{k-1}, r^k, t^k) \in \mathcal{G}, h^{k-1} \in L_u^{k-1}\} \quad \text{with } k = 1, 2, \cdots, H, \tag{4}$$

where H is the maximal number of propagations. Such propagation process is illustrated in Fig. 3. After H times propagation, we have $H+1$ sets of entities $L_u^k (k = 0, 1, ..., H)$.

4.2 Entity Representation

It is necessary to obtain a suitable representation for each entity, before exploring the interactions between entities contained in $L_u^k (k = 0, 1, ..., H)$. To depict the *intra-entity-interaction* discussed in Section 1, we propose to apply TransR [12] and obtain proper representations for entities as follows. For each triple $(h, r, t) \in \mathcal{G}$, entity embeddings and relation embedding are set as d-dimensional vectors, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$ respectively. Note that, although we assume the entity embeddings and relation embeddings are both d-dimensional in this paper, the dimensions of these two embeddings are not necessarily identical. For this relation r, we set a projection matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$, which projects entities from entity spaces to the corresponding relation space as:

$$\mathbf{h}_r = \mathbf{R}\mathbf{h}, \quad \mathbf{t}_r = \mathbf{R}\mathbf{t}. \tag{5}$$

For m^{th} entity $h^0_{v_{u,m}} \in L_u^0$, since this entity is directly clicked by user u and no relation is involved in, this entity can be represented as:

$$\mathbf{e}^0_{u,m} = \mathbf{h}^0_{v_{u,m}}, \tag{6}$$

where $\mathbf{h}^0_{v_{u,m}}$ is the embedding vector of $h^0_{v_{u,m}}$. For m^{th} entity $t^k_{u,m} \in L_u^k (k = 1, 2..., H)$, this entity is represented as:

$$\mathbf{e}^k_{u,m} = \mathbf{R}^k_{u,m}\mathbf{t}^k_{u,m}, \tag{7}$$

where $\mathbf{t}^k_{u,m}$ is the embedding vector of $t^k_{u,m}$, and $\mathbf{R}^k_{u,m}$ is projection matrix associated with the relation connected to $t^k_{u,m}$.

In addition, the input item v's representation $\mathbf{v}$ is also adopted without projecting into relation space:

$$\mathbf{v} = \mathbf{h}_v, \tag{8}$$

where $\mathbf{h}_v$ is the embedding vector of entity h_v linked to v.

4.3 Attention-based User Representation

For the $H+1$ sets of entities $L_u^k (k = 0, 1, ..., H)$, a common way to generate user's representation $\mathbf{u}$ is to take the average of the representations in these sets of entities, i.e., $\mathbf{u} = \frac{1}{H+1} \sum_{k=0}^{H} \frac{1}{|L_u^k|} \sum_m \mathbf{e}_{u,m}^k$. However, this method ignores the *inter-entity-interactions* discussed in Section 1. To this end, we first propose to learn a latent representation $\mathbf{a}_u^k$ for each L_u^k by applying a self-attention network, which captures the interactions among entities. Then, with regard to input item v, we propose to adopt an attention network [19, 22] to model the different impacts of $\mathbf{a}_u^k (k = 0, 1, ..., H)$ on $\mathbf{u}$, which characterizes user's diverse interests on different input items.

In general, attention mechanism can be described as mapping a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Furthermore in self-attention, the query, key and value are all the same [14]. In our context, we adopt a scaled dot-product attention and the whole calculation process for $\mathbf{a}_u^k$ is illustrated in Fig. 4. Specifically, for each L_u^k, the query $\mathbf{Q}_u^k$, key $\mathbf{K}_u^k$, and value $\mathbf{V}_u^k$ are all concatenated from $\mathbf{e}_{u,m}^k$ as follows:

$$\mathbf{V}_u^k = \mathbf{Q}_u^k = \mathbf{K}_u^k = [\mathbf{e}_{u,1}^k, \mathbf{e}_{u,2}^k, \cdots, \mathbf{e}_{u,N}^k], \tag{9}$$

where N is number of entities to be concatenated. If $\left|L_u^k\right| \geq N$, N entities are randomly picked out to perform concatenation. Otherwise, $N - \left|U^k\right|$ null entities (i.e., zero vectors) are appended to L_u^k to form $\mathbf{Q}_u^k$, $\mathbf{K}_u^k$ and $\mathbf{V}_u^k$. Then, the compatibility between $\mathbf{Q}_u^k$ and $\mathbf{K}_u^k$ is calculated as follows:

$$\mathbf{C}_u^k = \mathbf{Q}_u^{k\,T} \mathbf{K}_u^k, \tag{10}$$

where $\mathbf{C}_u^k \in \mathbb{R}^{N \times N}$ and the entry $c_{u,i,j}^k \in \mathbf{C}_u^k$ at row i, column j describes the compatibility between i^{th} entity $\mathbf{e}_{u,i}^k \in \mathbf{Q}_u^k$ and j^{th} entity $\mathbf{e}_{u,j}^k \in \mathbf{K}_u^k$. Finally, a *softmax* function is applied to derive $\mathbf{a}_u^k$:

$$\mathbf{a}_u^k = \mathbf{V}_u^k softmax_{\boldsymbol{\beta}_1}(\frac{\mathbf{C}_u^k}{\sqrt{d}}), \tag{11}$$

where $\sqrt{d}$ is applied to scale the compatibility matrix to avoid the dot product in Eq. 10 getting too large. The *softmax* function essentially calculates weight for each column $\mathbf{e}_{u,m}^k \in V^k (m = 1, ..., N)$, and the weight of $\mathbf{e}_{u,m}^k$ is described as

$$softmax_{\boldsymbol{\beta}_1}(\mathbf{X}_{u,m}) = \frac{\exp(\boldsymbol{\beta}_1^T \mathbf{X}_{u,m})}{\sum_{n=1}^{N} \exp(\boldsymbol{\beta}_1^T \mathbf{X}_{u,n})}, \tag{12}$$

where $\mathbf{X}_{u,m} \in \mathbb{R}^{N \times 1}$ is the m^{th} column of $\mathbf{C}_u^k$ and $\boldsymbol{\beta}_1 \in \mathbb{R}^{N \times N}$ is the *softmax* function's parameter. Note that a masking operation (i.e., the diagonal entries of $\mathbf{C}_u^k$ are set as zero) is applied before the *softmax* function, which avoids high compatibility scores between identical vectors of $\mathbf{Q}_u^k$ and $\mathbf{K}_u^k$. After performing weighted sum according to Eq. 11, we obtain the latent representations $\mathbf{a}_u^k \in \mathbb{R}^d, (k = 0, ..., H)$. Next, we will present how to obtain $\mathbf{u}$ from $\mathbf{a}_u^k$.

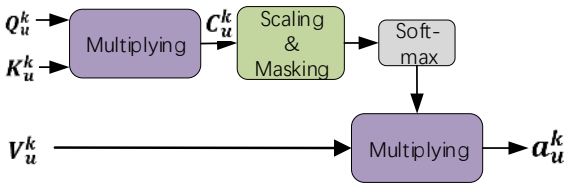

Figure 4: Illustration of Self-Attention network.

Similar to that $\mathbf{a}_u^k$ is described as a weighted sum of $\mathbf{e}_{u,m}^k$, the user's final representation $\mathbf{u}$ is described as a weighted sum of $\mathbf{a}_u^k$. In this context, the query $\mathbf{Q}_u$, key $\mathbf{K}_u$ and value $\mathbf{V}_u$ are described as:

$$\begin{aligned} \mathbf{Q}_u &= \mathbf{v}, \\ \mathbf{K}_u = \mathbf{V}_u &= [\mathbf{a}_u^0, \mathbf{a}_u^1, \cdots, \mathbf{a}_u^H], \end{aligned} \tag{13}$$

where $\mathbf{v}$ is the representation of input item v according to Eq. 8. And $\mathbf{u}$ is calculated by

$$\mathbf{u} = \mathbf{V}_u \; softmax_{\boldsymbol{\beta}_2}(\frac{\mathbf{Q}_u^T \mathbf{K}_u}{\sqrt{d}}). \tag{14}$$

Similar to Eq. 12, the $softmax_{\boldsymbol{\beta}_2}(\cdot)$ in Eq. 14 is a *softmax* function whose paremeter is $\boldsymbol{\beta}_2$. Note that, we do not perform masking operation here, since $\mathbf{Q}_u$ is not identical to $\mathbf{K}_u$.

Finally, based on the user's representation $\mathbf{u}$ and item's representation $\mathbf{v}$, the predicted CTR is calculated by

$$\hat{y}_{uv} = \sigma(\mathbf{u}^T \mathbf{v}), \tag{15}$$

where $\sigma(\cdot)$ is the sigmoid function.

5 LEARNING ALGORITHM

In this section, we present the details of how we can learn the optimal parameters of AKUPM given user-item interaction matrix Y and knowledge graph $\mathcal{G}$.

Suppose that Θ denotes all parameters in AKUPM, including embeddings of all entities $\mathbf{h}, \mathbf{t}$, embeddings of all relations $\mathbf{r}$, all relation projection matrices $\mathbf{R}$, self-attention network's parameter $\boldsymbol{\beta}_1$ and attention network's parameter $\boldsymbol{\beta}_2$. We intend to maximize the following posterior probability of model parameters Θ after observing the $\mathcal{G}$ and Y:

$$\Theta = \underset{\Theta}{\operatorname{argmax}}\, p(\Theta | \mathcal{G}, Y), \tag{16}$$

which is equivalent to maximizing

$$\begin{aligned} p(\Theta|\mathcal{G}, Y) &= \frac{p(\Theta, \mathcal{G}, Y)}{p(\mathcal{G}, Y)} \propto p(\Theta, \mathcal{G}, Y) \\ &= p(\Theta) p(\mathcal{G}|\Theta) p(Y|\Theta, \mathcal{G}) \\ &= p(\Theta) p(\mathcal{G}|\Theta) p(Y|\Theta) \\ &= p(\mathbf{h}, \mathbf{t}, \mathbf{r}, \mathbf{R}) p(\mathcal{G}|\mathbf{h}, \mathbf{t}, \mathbf{r}, \mathbf{R}) * \\ &\quad p(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) p(Y|\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) \\ &\propto p(\mathcal{G}|\mathbf{h}, \mathbf{t}, \mathbf{r}, \mathbf{R}) p(Y|\boldsymbol{\beta}_1, \boldsymbol{\beta}_2), \end{aligned} \tag{17}$$

$p(\mathcal{G}|\cdot)$ is the likelihood of the observed $\mathcal{G}$ given corresponding parameters and $p(Y|\cdot)$ is the likelihood of the observed Y given corresponding parameters. $p(\mathcal{G}, Y)$, $p(\mathbf{h}, \mathbf{t}, \mathbf{r}, \mathbf{R})$ and $p(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2)$ are priori probabilities which can be ignored in maximizing problem. To

maximize Eq. 17, we provide the details to estimate the likelihood of observed knowledge graph and implicit feedback in the following sections.

5.1 Likelihood of Observed Knowledge Graph and Implicit Feedback

For each observed triple $(h, r, t) \in \mathcal{G}$, we follow the existing work [8] and define the score function as

$$f_r(h, t) = \|\mathbf{Rh} + \mathbf{r} - \mathbf{Rt}\|^2 . \tag{18}$$

For this triple, by randomly replacing tail entity we can get a new triple which is not included in the knowledge graph, i.e., $(h, r, t') \notin \mathcal{G}$. By doing do, we construct a new quadruple set $\mathcal{G}'$ in which every quadruple $(h, r, t, t') \in \mathcal{G}'$ satisfying both $(h, r, t) \in \mathcal{G}$ and $(h, r, t') \notin \mathcal{G}$. We follow the previous work [21], and define the likelihood of observing the quadruple (h, r, t, t') as follows:

$$p((h, r, t, t')|\mathbf{h}, \mathbf{r}, \mathbf{t}, \mathbf{R}) = \sigma(f_r(h, t) - f_r(h, t')), \tag{19}$$

where $\sigma(\cdot)$ is the sigmoid function. As a result, the likelihood of observed knowledge graph can be described as

$$p(\mathcal{G}|\mathbf{h}, \mathbf{r}, \mathbf{t}, \mathbf{R}) = \prod_{(h,r,t,t') \in \mathcal{G}'} \sigma(f_r(h, t) - f_r(h, t')). \tag{20}$$

For each entry $y_{uv} \in Y$, the likelihood can be defined as the product of Bernoulli distributions:

$$p(Y|\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) = \prod_{y_{uv} \in Y} \hat{y}_{uv}^{y_{uv}} * (1 - \hat{y}_{uv})^{1-y_{uv}}, \tag{21}$$

where $\hat{y}_{uv}$ is predicted CTR for the input pair (u, v) according to Eq. 15.

5.2 Loss Function

Taking Eq. 20 and Eq. 21 into Eq. 16 gives:

$$\begin{aligned} \Theta = \underset{\Theta}{\operatorname{argmax}} & \prod_{(h,r,t,t') \in \mathcal{G}'} \sigma(f_r(h, t) - f_r(h, t')) * \\ & \prod_{y_{uv} \in Y} \hat{y}_{uv}^{y_{uv}} * (1 - \hat{y}_{uv})^{1-y_{uv}}. \end{aligned} \tag{22}$$

After taking the negative logarithm of Eq. 22 and adding regularization terms, we have the following loss function for AKUPM:

$$\begin{aligned} & \mathcal{L}(\mathbf{h}, \mathbf{t}, \mathbf{r}, \mathbf{R}_r, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2) = \\ & - \sum_{y_{uv} \in Y} \Big(y_{uv} \log \hat{y}_{uv} + (1 - y_{uv}) \log(1 - \hat{y}_{uv}) \Big) \\ & - \sum_{(h,r,t,t') \in \mathcal{G}'} \log \Big(\sigma(f_r(h, t) - f_r(h, t')) \Big) \\ & + \frac{\lambda_{\mathcal{G}}}{2} \Big(\|\mathbf{h}\|_2^2 + \|\mathbf{t}\|_2^2 + \|\mathbf{r}\|_2^2 + \|\mathbf{R}_r\|_2^2 \Big) \\ & + \frac{\lambda_{\boldsymbol{\beta}_1}}{2} \|\boldsymbol{\beta}_1\|_2^2 + \frac{\lambda_{\boldsymbol{\beta}_2}}{2} \|\boldsymbol{\beta}_2\|_2^2 . \end{aligned} \tag{23}$$

To minimize the objective in Eq. 23, we employ a batch gradient descent algorithm through the entire observed training set Y and $\mathcal{G}'$ and update each parameter using the corresponding gradient of the loss function.

Table 1: Detailed statistics of the two datasets (# means 'the number of ').

	MovieLens-1M	Book-Crossing
# users	6,036	17,860
# items	2,445	14,967
# positive feedback	376,886	69,873
# negative feedback	376,886	69,873
# entities	182,011	77,903
# relation types	12	25
# facts	1,241,995	198,771

6 EXPERIMENTS

In this section, we demonstrate the performance of AKUPM on two popular real-world public datasets[3] compared to some state-of-the-art baselines. We first introduce the experiment setup, including datasets, compared baselines and evaluation schema. Then, we compare the performance of AKUPM with baselines and analyze the results in detail. Finally, to demonstrate the necessity of depicting *intra-entity-interaction* and *inter-entity-interaction*, we compare the performance of AKUPM with several variants.

6.1 Experiment Setup

6.1.1 Datasets. To verify AKUPM's effectiveness in different application scenarios, two datasets from different domains (movie and book) are utilized in our experiments.

- **MovieLens-1M**[4] is a widely used benchmark dataset in movie recommendations, which contains 1,000,209 explicit ratings from 6,040 users on 3,900 movies. Each rating is an integer between 1 and 5.
- **Book-Crossing**[5] is a widely used benchmark dataset in book recommendations, which consists of 1,149,780 explicit ratings from 278,858 users on 271,379 books. Each rating is an integer between 0 and 10.

Since AKUPM is designed to make recommendations based on implicit feedback, in order to be consistent with the implicit feedback setting, we transform these two explicit feedback datasets into implicit feedback datasets by setting threshold on ratings (4 for MovieLens-1M and 1 for Book-Crossing)[6] similar to [23]. For each user, when we transform all his/her ratings which are no less than threshold into positive (i.e., 1) implicit feedback, we also sample same amount of his/her unrated movies as negative (i.e., 0) implicit feedback. Besides, users containing no positive implicit feedback are removed from the datasets, since there is no positive implicit feedback that can be used to verify the recommendation results.

The knowledge graphs used in this paper are released by [17], which are constructed to match the items in MovieLens-1M and Book-Crossing, respectively. In summary, the knowledge graph for MovieLens-1M contains 182,011 entities, 12 relations and 1,241,995

[3] Experiment code is provided at https://github.com/gegetang/akupm
[4] https://grouplens.org/datasets/movielens/1m/
[5] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[6] The threshold of Book-Crossing is set much lower than that of MovieLens-1M because the sparsity of Book-Crossing is more than 99.99%.

facts; and the knowledge graph for Book-Crossing contains 77,903 entities, 25 relations and 198,771 facts.

For clarity, detailed statistics of the two processed datasets are presented in Table 1.

6.1.2 Baselines. We choose the following 5 state-of-the-art models as the baselines to compare with AKUPM:

- **CKE** [21] combines a CF module with knowledge embedding, text embedding, and image embedding of items in a unified Bayesian framework. Here, each item is only represented as the embedding learned by TransR, since the textual description and the visual image is not available in our datasets.
- **DKN** [16] generates a knowledge-aware embedding vector for each input news by fusing its word-level embeddings and entity-level embeddings. Then, an attention module is utilized to dynamically model the user's diverse interests for news recommendations. Here, the title of movie or book is used to generate word-level embeddings, and entity-level embeddings are learned by TransD [6] which was reported to have the best performance in [16].
- **RippleNet** [17] represents a user as plenty of entities related to the user's historically clicking items. Then, the user's representation is combined with item's representation to predict the CTR of this user-item pair. Note that, in this method, the entity embedding is learned by TransE [2].
- **LibFM** [11] is a state-of-the-art feature-based factorization model and is widely used in CTR predictions. Here, we concatenate user ID, item ID and the item embeddings learned by TransR as features to feed into LibFM.
- **DeepWide** [3] is a general deep neural network for recommender system in which a wide linear model component is combined with a deep non-linear neural network component to achieve both memorization and generalization. Similar to LibFM, we concatenate user ID, item ID and the item embeddings learned by TransR as the input for DeepWide.

Note that the hyper-parameters of baselines are set as best ones reported in original papers.

6.1.3 Evaluation Schema. For each dataset, similar to [21], 80% of items associated with each user are randomly selected for training; 10% of items are for evaluation; and all the remaining constitutes the validation set which is used to find the optimal hyper-parameters for our proposed AKUPM.

We evaluate our method in two application scenarios: CTR prediction and top-K recommendation. For each evaluation scenario, all models are repeatedly trained 5 times and we report the averaged evaluation metrics of each model. More specifically, for CTR prediction, we adopt *ACC (Accuracy)* and *AUC (Area Under Curve)* to evaluate the performance. For top-K recommendation, we adopt *Precision@K*, *Recall@K* and *F1@K* to evaluate the performance.

6.2 Performance Comparisons with Baselines

In this section, we present the results of performance comparisons among AKUPM and baselines. The detailed hyper-parameter settings are shown in Table 2.

Table 2: Detailed hyper-parameter settings of AKUPM. d denotes the dimension of entity embeddings; H is the maximal number of propagations; N is number of entities picked out to perform concatenation; $\lambda_{\mathcal{G}}$, λ_{β_1} and λ_{β_2} are all regularization coefficients.

	d	H	N	$\lambda_{\mathcal{G}}$	λ_{β_1}	λ_{β_2}
MovieLens-1M	16	2	32	10^{-2}	10^{-4}	10^{-5}
Book-Crossing	4	3	32	2×10^{-2}	10^{-3}	10^{-3}

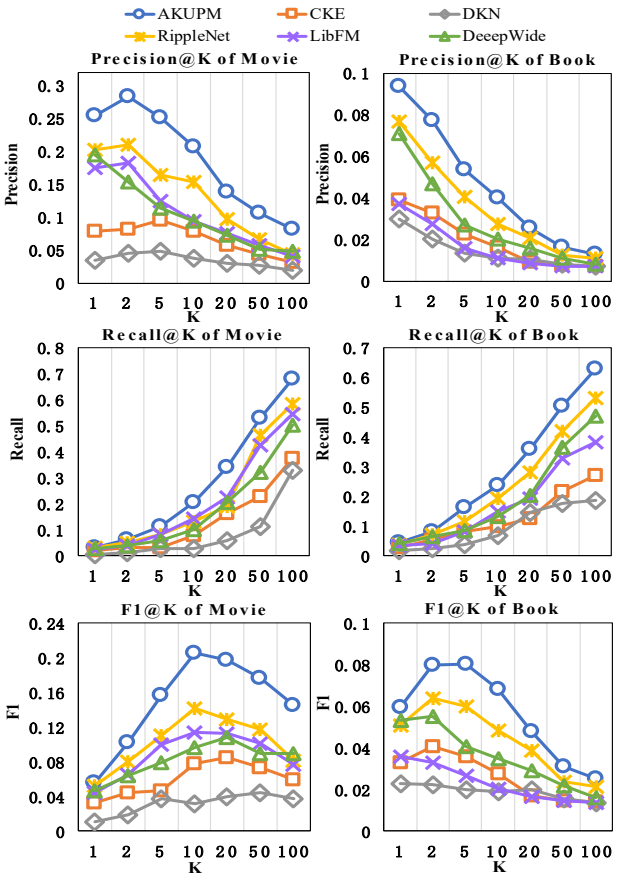

Figure 5: Comparison of different models in top-K recommendation scenarios on MovieLens-1M (the left column) and Book-Crossing (the right column).

The results of all methods in CTR prediction and top-K recommendation are presented in Table 3 and Figure 5, respectively. It can be observed that:

(1) AKUPM performs best among all methods in both two scenarios on the two datasets. Specifically, as shown in Table 3, AKUPM outperforms the best of baselines by 6.868% and 10.169% on the MovieLens-1M dataset in terms of AUC and ACC, respectively; AKUPM outperforms the best of baselines by 7.116% and 11.310% on the Book-Crossing dataset in terms of AUC and ACC, respectively. AKUPM also achieves

Table 3: Comparison of different models in CTR prediction scenario (impr. means 'improvements').

Model	MovieLens-1M				Book-Crossing			
	AUC ↑	AKUPM impr. on AUC	ACC ↑	AKUPM impr. on ACC	AUC ↑	AKUPM impr. on AUC	ACC ↑	AKUPM impr. on ACC
AKUPM	**0.918**	-	**0.845**	-	**0.843**	-	**0.807**	-
RippleNet	0.843	8.897%	0.767	10.169%	0.771	9.339%	0.725	11.310%
CKE	0.791	16.056%	0.729	15.912%	0.638	32.13%	0.592	36.318%
DKN	0.679	35.198%	0.607	39.209%	0.621	35.749%	0.591	36.548%
LibFM	0.805	14.037%	0.736	14.809%	0.782	7.801%	0.723	11.618%
DeepWide	0.859	6.868%	0.762	10.892%	0.787	7.116%	0.717	12.552%

significant improvements in terms of *Precision@K*, *Recall@K* and *F1@K* as shown in Figure 5.

(2) All models' performance on the Book-Crossing dataset are poorer than that on the MovieLens-1M dataset. The main reason is that the average positive feedback per user of Book-Crossing dataset (3.91) is much smaller than that of MovieLens-1M dataset (62.44). As a result, there is not enough information on the Book-Crossing dataset for models to learn users' interests.

(3) DKN performs worst compared with others. The reason may be that the title of a movie or a book is much shorter than news, making the word-level embeddings and entity-level embeddings containing insufficient information to make recommendations.

(4) CKE performs poorly in our experiments. The reason may be two-fold: the textual description and visual image is not available in our datasets; compared to AKUPM and RippleNet which incorporate many entities potentially related to the input item, CKE only use the one entity directly related to the input item for recommendation.

(5) Both LibFM and DeepWide achieve satisfactory performance, which are widely used in recommender systems. This implies that the additional usage of entity embedding can boost the recommendation performance.

(6) RippleNet nearly achieves best performance among all baselines. AKUPM and RippleNet are similar in incorporating entities to represent the user's hierarchical preferences. However, RippleNet does not explore the relationships between the user and the incorporating entities, so that the results may suffer a lot from unrelated entities.

6.3 Performance Comparisons with Variants

Table 4: Comparison among AKUPM and its two variants in CTR prediction scenario.

Model	MovieLens-1M		Book-Crossing	
	AUC↑	ACC↑	AUC↑	ACC↑
AKUPM	**0.918**	**0.845**	**0.843**	**0.807**
AKUPM-intra	0.909	0.837	0.827	0.788
AKUPM-inter	0.862	0.799	0.803	0.771

In this section, to demonstrate the necessity of capturing *intra-entity-interaction* and *inter-entity-interaction*, we compare the performance of AKUPM with the following two variants:

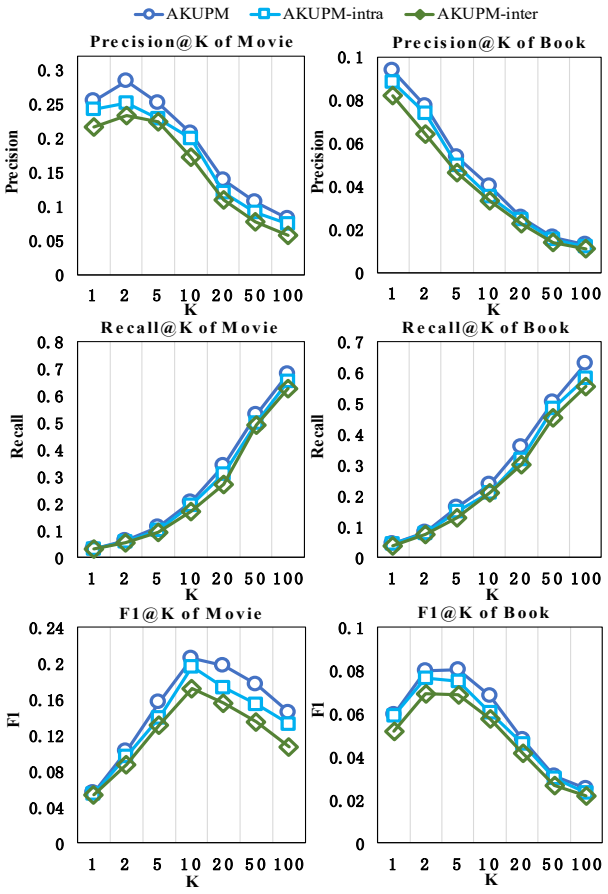

Figure 6: Comparison among AKUPM and its variants in top-*K* recommendation scenarios on MovieLens-1M (the left column)and Book-Crossing (the right column).

- **AKUPM-intra**. This model is one variant of AKUPM without depicting *intra-entity-interaction*, termed as AKUPM-intra. In particular, the representation of entity $t^k_{u,m} \in L^k_u (k = 1, 2..., H)$ is simply in the form of $\mathbf{t}^k_{u,m}$, discarding projection from the entity spaces to corresponding relation spaces abstracted in Eq. 7.

- **AKUPM-inter**. In this model, user embeddings are generated according to $\mathbf{a}_u^k = \frac{1}{|L_u^k|} \sum_{\mathbf{e}_{u,i}^k \in L_u^k} \mathbf{e}_{u,i}^k$, i.e., AKUMP without considering *inter-entity-interaction*.

The results of AKUPM-intra, AKUPM-inter and AKUPM in CTR prediction and top-K scenarios are presented in Table 4 and Figure 6, respectively. It can be observed that:

(1) AKUPM performs better than AKUPM-inter, because the self-attention network not only allows to capture the inter-entity-interactions among entities, but also allows to inspect how much each entity contributes to the user embedding.
(2) AKUPM performs better than AKUPM-intra. Because without projecting entity into relation spaces, AKUPM-intra is unable to obtain appropriate entity embedding when this entity is involved in different relations.

The results demonstrate that it is necessary to capture *intra-entity-interaction* and *inter-entity-interaction* when incorporating entities from knowledge graph to make recommendations.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel model termed as AKUPM, which aims at addressing the sparse problem in recommender systems by incorporating entities from a knowledge graph to infer users' potential interests. Following this idea, AKUPM first iteratively extends a user's historical interacted entities along relations in the knowledge graph to introduce much external knowledge. Next, AKUPM figures out the most related part of the incorporated entities based on depicting *intra-entity-interaction* and *inter-entity-interaction* among entities, so as to make better CTR prediction for a given user-item pair. Moreover, we conduct extensive experiments on two real-world public datasets and the results show that AKUPM achieves substantial gains in terms of common evaluation metrics (e.g., AUC, ACC and Recall@top-K) over several state-of-the-art baselines.

In the future, we plan to further evaluate the effectiveness of our model on more real-world datasets. Moreover, since the number of entities related to a user is variable, we plan to learn a better latent representation by introducing recurrent neural network which has ability to handle the variable-length input, rather than randomly pick out a certain number of entities in this paper.

8 ACKNOWLEDGMENTS

We thank the anonymous reviewers for taking time to read and make valuable comments on this paper. This work was supported by the National Natural Science Foundation of China (71671069), and National Key R&D Program of China (2018YFC0830900).

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* abs/1409.0473 (2014), 501–510.
[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*. Curran Associates, Inc., Lake Tahoe, Nevada, United States, 2787–2795.
[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, Boston, MA, USA, 7–10.
[4] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*. ACM, ACM, Graz, Austria, 121–124.
[5] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, IEEE Computer Society, Washington, DC, USA, 263–272.
[6] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. The Association for Computer Linguistics, Beijing, China, 687–696.
[7] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.
[8] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *AAAI*, Vol. 15. AAAI Press, Austin Texas, USA, 2181–2187.
[9] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* arXiv:1703.03130 (2017), 505–514.
[10] David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, New York, NY, USA, 509–518.
[11] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.
[12] Kai Shu, Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2018. Crossfire: Cross media joint friend and item recommendations. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, Marina Del Rey, CA, USA, 522–530.
[13] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. 2015. Using linked data traversal to label academic communities. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, Florence, Italy, 1029–1034.
[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. Curran Associates, Inc., Long Beach, CA, USA, 5998–6008.
[15] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, Marina Del Rey, CA, USA, 592–600.
[16] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, Lyon, France, 1835–1844.
[17] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, Torino, Italy, 417–426.
[18] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.
[19] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Halifax, NS, Canada, 2051–2059.
[20] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*, Vol. 14. AAAI Press, Québec City, Québec, Canada, 1112–1119.
[21] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, San Francisco, California, USA, 353–362.
[22] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, ACM, London, United Kingdom, 1059–1068.
[23] Ke Zhou and Hongyuan Zha. 2012. Learning binary codes for collaborative filtering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Beijing, China, 498–506.