Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation
Hongwei Wang\textsuperscript{1,2}, Fuzheng Zhang\textsuperscript{3}, Miao Zhao\textsuperscript{4}, Wenjie Li\textsuperscript{4}, Xing Xie\textsuperscript{2}, Minyi Guo\textsuperscript{1,}\
\textsuperscript{1}Shanghai Jiao Tong University, Shanghai, China\textsuperscript{*}\textsuperscript{2}Microsoft Research Asia, Beijing, China\textsuperscript{3}Meituan-Dianping Group, Beijing, China\textsuperscript{4}The Hong Kong Polytechnic University, Hong Kong, China
wanghaiwei55@gmail.com,zhangfuzheng@meituan.com,[csmiaozhao,cswjli]@comp.polyu.edu.hk
xingx@microsoft.com,guo-my@cs.sjtu.edu.cn

ABSTRACT
Collaborative filtering often suffers from sparsity and cold start problems in real recommendation scenarios, therefore, researchers and engineers usually use side information to address the issues and improve the performance of recommender systems. In this paper, we consider knowledge graphs as the source of side information. We propose MKR, a Multi-task feature learning approach for Knowledge graph enhanced Recommendation. MKR is a deep end-to-end framework that utilizes knowledge graph embedding task to assist recommendation task. The two tasks are associated by cross&compress units, which automatically share latent features and learn high-order interactions between items in recommender systems and entities in the knowledge graph. We prove that cross&compress units have sufficient capability of polynomial approximation, and show that MKR is a generalized framework over several representative methods of recommender systems and multi-task learning. Through extensive experiments on real-world datasets, we demonstrate that MKR achieves substantial gains in multi-task learning. Through extensive experiments on real-world datasets, we demonstrate that MKR achieves substantial gains in movie, book, music, and news recommendation, over state-of-the-art baselines. MKR is also shown to be able to maintain a decent performance even if user-item interactions are sparse.

KEYWORDS
Recommender systems; knowledge graph; multi-task learning

ACM Reference Format:
Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation In Proceedings of The 2019 Web Conference (WWW 2019). ACM, New York, NY, USA, 11 pages. https://doi.org/xxxxxx

1 INTRODUCTION
Recommender systems (RS) aims to address the information explosion and meet users personalized interests. One of the most popular recommendation techniques is collaborative filtering (CF) [11], which utilizes users’ historical interactions and makes recommendations based on their common preferences. However, CF-based methods usually suffer from the sparsity of user-item interactions and the cold start problem. Therefore, researchers propose using side information in recommender systems, including social networks [10], attributes [30], and multimedia (e.g., texts [29], images [40]). Knowledge graphs (KGs) are one type of side information for RS, which usually contain fruitful facts and connections about items. Recently, researchers have proposed several academic and commercial KGs, such as NELL\textsuperscript{3}, DBpedia\textsuperscript{4}, Google Knowledge Graph\textsuperscript{5} and Microsoft Satori\textsuperscript{6}. Due to its high dimensionality and heterogeneity, a KG is usually pre-processed by knowledge graph embedding (KGE) methods [27], which embeds entities and relations into low-dimensional vector spaces while preserving its inherent structure.

Existing KG-aware methods
Inspired by the success of applying KG in a wide variety of tasks, researchers have recently tried to utilize KG to improve the performance of recommender systems [31, 32, 39, 40, 45]. Personalized Entity Recommendation (PER) [39] and Factorization Machine with Group lasso (FMG) [45] treat KG as a heterogeneous information network, and extract meta-path/meta-graph based latent features to represent the connectivity between users and items along different types of relation paths/graphs. It should be noted that PER and FMG rely heavily on manually designed meta-paths/meta-graphs, which limits its application in generic recommendation scenarios. Deep Knowledge-aware Network (DKN) [32] designs a CNN framework to combine entity embeddings with word embeddings for news recommendation. However, the entity embeddings are required in advance of using DKN, causing DKN to lack an end-to-end way of training. Another concern about DKN is that it can hardly incorporate side information other than texts. RippleNet [31] is a memory-network-like model that propagates users’ potential preferences in the KG and explores their hierarchical interests. But the importance of relations is weakly characterized in RippleNet, because the embedding matrix of a relation R can hardly be trained to capture the sense of importance in the quadratic form $v^T Rh$ (v and h are embedding vectors of two entities). Collaborative Knowledge base Embedding (CKE) [40] combines CF with structural knowledge, textual knowledge, and visual knowledge in a unified framework. However, the KGE module in CKE (i.e., TransR [13]) is more suitable for in-graph applications (such as KG completion and link prediction) rather than recommendation. In addition, the CF module and the KGE module are loosely coupled

\textsuperscript{\hspace{1em}}\textsuperscript{\textsuperscript{*}M. Guo is the corresponding author. This work was partially sponsored by the National Basic Research 973 Program of China under Grant 2015CB352403.}
in CKE under a Bayesian framework, making the supervision from KG less obvious for recommender systems.

The proposed approach
To address the limitations of previous work, we propose MKR, a multi-task learning (MTL) approach for knowledge graph enhanced recommendation. MKR is a generic, end-to-end deep recommendation framework, which aims to utilize KGE task to assist recommendation task\(^1\). Note that the two tasks are not mutually independent, but are highly correlated since an item in RS may associate with one or more entities in KG. Therefore, an item and its corresponding entity are likely to have a similar proximity structure in RS and KG, and share similar features in low-level and non-task-specific latent feature spaces [15]. We will further validate the similarity in the experiments section. To model the shared features between items and entities, we design a cross\&compress unit in MKR. The cross\&compress unit explicitly models high-order interactions between item and entity features, and automatically control the cross knowledge transfer for both tasks. Through cross\&compress units, representations of items and entities can complement each other, assisting both tasks in avoiding fitting noises and improving generalization. The whole framework can be trained by alternately optimizing the two tasks with different frequencies, which endows MKR with high flexibility and adaptability in real recommendation scenarios.

We probe the expressive capability of MKR and show, through theoretical analysis, that the cross\&compress unit is capable of approximating sufficiently high order feature interactions between items and entities. We also show that MKR is a generalized framework over several representative methods of recommender systems and multi-task learning, including factorization machines [22, 23], deep\&cross network [34], and cross-stitch network [18]. Empirically, we evaluate our method in four recommendation scenarios, i.e., movie, book, music, and news recommendations. The results demonstrate that MKR achieves substantial gains over state-of-the-art baselines in both click-through rate (CTR) prediction (e.g., 11.6\% AUC improvements on average for movies) and top-K recommendation (e.g., 66.4\% Recall@10 improvements on average for books). MKR can also maintain a decent performance in sparse scenarios.

Contribution
It is worth noticing that the problem studied in this paper can also be modelled as cross-domain recommendation [26] or transfer learning [21], since we care more about the performance of recommendation task. However, the key observation is that though cross-domain recommendation and transfer learning have single objective for the target domain, their loss functions still contain constraint terms for measuring data distribution in the source domain or similarity between two domains. In our proposed MKR, the KGE task serves as the constraint term explicitly to provide regularization for recommender systems. We would like to emphasize that the major contribution of this paper is exactly modeling the problem as multi-task learning: We go a step further than cross-domain recommendation and transfer learning by finding that the inter-task similarity is helpful to not only recommender systems but also knowledge graph embedding, as shown in theoretical analysis and experiment results.

2 OUR APPROACH
In this section, we first formulate the knowledge graph enhanced recommendation problem, then introduce the framework of MKR and present the design of the cross\&compress unit, recommendation module and KGE module in detail. We lastly discuss the learning algorithm for MKR.

2.1 Problem Formulation
We formulate the knowledge graph enhanced recommendation problem in this paper as follows. In a typical recommendation scenario, we have a set of \( M \) users \( U = \{ u_1, u_2, ..., u_M \} \) and a set of \( N \) items \( \mathcal{V} = \{ v_1, v_2, ..., v_N \} \). The user-item interaction matrix \( Y \in \mathbb{R}^{M \times N} \) is defined according to users’ implicit feedback, where \( y_{uv} = 1 \) indicates that user \( u \) engaged with item \( v \), such as behaviors of clicking, watching, browsing, or purchasing; otherwise \( y_{uv} = 0 \). Additionally, we also have access to a knowledge graph \( \mathcal{G} \), which is comprised of entity-relation-entity triples \((h, r, t)\). Here \( h, r, \) and \( t \) denote the head, relation, and tail of a knowledge triple, respectively. For example, the triple \((\text{Quentin Tarantino}, \text{film.director}, \text{Pulp Fiction})\) states the fact that Quentin Tarantino directs the film Pulp Fiction. In many recommendation scenarios, an item \( v \in \mathcal{V} \) may associate with one or more entities in \( \mathcal{G} \). For example, in movie recommendation, the item “Pulp Fiction” is linked with its namesake (Quentin Tarantino) and “Silicon Valley” in a KG.

Given the user-item interaction matrix \( Y \) as well as the knowledge graph \( \mathcal{G} \), we aim to predict whether user \( u \) has potential interest in item \( v \) with which he has had no interaction before. Our goal is to learn a prediction function \( \hat{y}_{uv} = \mathcal{F}(u, v|\Theta, Y, \mathcal{G}) \), where \( \hat{y}_{uv} \) denotes the probability that user \( u \) will engage with item \( v \), and \( \Theta \) is the model parameters of function \( \mathcal{F} \).

2.2 Framework
The framework of MKR is illustrated in Figure 1a. MKR consists of three main components: recommendation module, KGE module, and cross\&compress units. (1) The recommendation module on the left takes a user and an item as input, and uses a multi-layer perceptron (MLP) and cross\&compress units to extract short and dense features for the user and the item, respectively. The extracted features are then fed into another MLP together to output the predicted probability. (2) Similar to the left part, the KGE module in the right part also uses multiple layers to extract features from the head and relation of a knowledge triple, and outputs the representation of the predicted tail under the supervision of a score function \( f \) and the real tail. (3) The recommendation module and the KGE module are bridged by specially designed cross\&compress units. The proposed unit can automatically learn high-order feature interactions of items in recommender systems and entities in the KG.

2.3 Cross\&compress Unit
To model feature interactions between items and entities, we design a cross\&compress unit in MKR framework. As shown in Figure 1b,
for item \( v \) and one of its associated entities \( e \), we first construct \( d \times d \) pairwise interactions of their latent feature \( v_l \in \mathbb{R}^d \) and \( e_l \in \mathbb{R}^d \) from layer \( l \):

\[
C_l = v_l e_l^T = \begin{bmatrix}
    v_{l1}^{(1)}e_{l1}^{(1)} & \cdots & v_{l1}^{(d)}e_{l1}^{(d)} \\
    \vdots & \ddots & \vdots \\
    v_{l1}^{(d)}e_{l1}^{(1)} & \cdots & v_{l1}^{(d)}e_{l1}^{(d)} \\
\end{bmatrix},
\]

(1)

where \( C_l \in \mathbb{R}^{d \times d} \) is the cross feature matrix of layer \( l \), and \( d \) is the dimension of hidden layers. This is called the cross operation, since each possible feature interaction \( v_l^{(i)}e_l^{(j)}, v(i, j) \in \{1, \ldots, d\}^2 \) between item \( v \) and its associated entity \( e \) is modeled explicitly in the cross feature matrix. We then output the feature vectors of items and entities for the next layer by projecting the cross feature matrix into their latent representation spaces:

\[
v_{l+1} = C_l v_l EV + C_l e_l EE + b_l, \\
e_{l+1} = C_l v_l EV + C_l e_l EE + b_l, \]

(2)

where \( w_l \in \mathbb{R}^d \) and \( b_l \in \mathbb{R}^d \) are trainable weight and bias vectors. This is called the compress operation, since the weight vectors project the cross feature matrix from \( \mathbb{R}^{d \times d} \) space back to the feature spaces \( \mathbb{R}^d \). Note that in Eq. (2), the cross feature matrix is compressed along both horizontal and vertical directions (by operating on \( C_l \) and \( C_l^T \)) for the sake of symmetry, but we will provide more insights of the design in Section 3.2. For simplicity, the cross&compress unit is denoted as:

\[
[v_{l+1}, e_{l+1}] = C(v_l, e_l),
\]

(3)

and we use a suffix \( v \) or \( e \) to distinguish its two outputs in the following of this paper. Through cross&compress units, MKR can adaptively adjust the weights of knowledge transfer and learn the relevance between the two tasks.

It should be noted that cross&compress units should only exist in low-level layers of MKR, as shown in Figure 1a. This is because:

(1) In deep architectures, features usually transform from general to specific along the network, and feature transferability drops significantly in higher layers with increasing task dissimilarity [38]. Therefore, sharing high-level layers risks to possible negative transfer, especially for the heterogeneous tasks in MKR. (2) In high-level layers of MKR, item features are mixed with user features, and entity features are mixed with relation features. The mixed features are not suitable for sharing since they have no explicit association.

### 2.4 Recommendation Module

The input of the recommendation module in MKR consists of two raw feature vectors \( u \) and \( v \) that describe user \( u \) and item \( v \), respectively. \( u \) and \( v \) can be customized as one-hot ID [8], attributes [30], bag-of-words [29], or their combinations, based on the application scenario. Given user \( u \)’s raw feature vector \( u \), we use an \( L \)-layer MLP to extract its latent condensed feature:

\[
u_L = M(M(\cdots M(u))) = M^L(u), \]  

(4)

where \( M(x) = \sigma(Wx + b) \) is a fully-connected neural network layer\(^7\) with weight \( W \), bias \( b \), and nonlinear activation function \( \sigma(\cdot) \). For item \( v \), we use \( L \) cross&compress units to extract its feature:

\[
v_L = \mathbb{E}_{e \sim S(v)} [C^L(v, e)[v]] , \]

(5)

where \( S(v) \) is the set of associated entities of item \( v \).

After having user \( u \)’s latent feature \( u_L \) and item \( v \)’s latent feature \( v_L \), we combine the two pathways by a predicting function \( f_{RS} \) for example, inner product or an \( H \)-layer MLP. The final predicted probability of user \( u \) engaging item \( v \) is:

\[
\hat{y}_{uv} = \sigma(f_{RS}(u_L, v_L)). \]

\(^7\)We use the exponent notation \( L \) in Eq. (4) and following equations in the rest of this paper for simplicity, but note that the parameters of \( L \) layers are actually different.
2.5 Knowledge Graph Embedding Module

Knowledge graph embedding is to embed entities and relations into continuous vector spaces while preserving their structure. Recently, researchers have proposed a great many KGE methods, including translational distance models [2, 13] and semantic matching models [14, 19]. In MKR, we propose a deep semantic matching architecture for KGE module. Similar to the recommendation module, for a given knowledge triple \((h, r, t)\), we first utilize multiple cross&compress units and nonlinear layers to process the raw feature vectors of head \(h\) and relation \(r\) (including ID [13], types [36], textual description [35], etc.), respectively. Their latent features are then concatenated together, followed by a K-layer MLP for predicting tail \(t\):

\[
h_L = \mathbb{E}_{v \sim S(h)} C^L(v, h)[e],
\]

\[
r_L = M^L(r),
\]

\[
\hat{t} = M^K \begin{pmatrix} h_1 \\ r_1 \end{pmatrix},
\]

where \(S(h)\) is the set of associated items of entity \(h\), and \(\hat{t}\) is the predicted vector of tail \(t\). Finally, the score of the triple \((h, r, t)\) is calculated using a score (similarity) function \(f_{KG}\):

\[
score(h, r, t) = f_{KG}(t, \hat{t}),
\]

where \(t\) is the real feature vector of \(t\). In this paper, we use the normalized inner product \(f_{KG}(t, \hat{t}) = \sigma(t^T \hat{t})\) as the choice of score function [18], but other forms of (dis)similarity metrics can also be applied here such as Kullback-Leibler divergence.

2.6 Learning Algorithm

The complete loss function of MKR is as follows:

\[
\mathcal{L} = \mathcal{L}_{RS} + \mathcal{L}_{KG} + \mathcal{L}_{REG}
\]

\[
= \sum_{u \in \mathbb{U}, v \in \mathbb{V}} \mathcal{J}(\hat{y}_{uv}, y_{uv})
\]

\[
- \lambda_1 \left( \sum_{(h, r, t) \in \mathbb{G}} \text{score}(h, r, t) - \sum_{(h', r, t') \notin \mathbb{G}} \text{score}(h', r, t') \right)
\]

\[
+ \lambda_2 ||W||^2_2.
\]

In Eq. (9), the first term measures loss in the recommendation module, where \(u \) and \(v \) traverse the set of users and the items, respectively, and \(\mathcal{J} \) is the cross-entropy function. The second term calculates the loss in the KGE module, in which we aim to increase the score for all true triples while reducing the score for all false triples. The last item is the regularization term for preventing overfitting, \(\lambda_1\) and \(\lambda_2\) are the balancing parameters.\(^8\)

Note that the loss function in Eq. (9) traverses all possible user-item pairs and knowledge triples. To make computation more efficient, following [17], we use a negative sampling strategy during training. The learning algorithm of MKR is presented in Algorithm 1, in which a training epoch consists of two stages: recommendation task (line 3-7) and KGE task (line 8-10). In each iteration, we repeat training on recommendation task for \(t\) times (\(t\) is a hyperparameter and normally \(t > 1\)) before training on KGE task once in each epoch, since we are more focused on improving recommendation performance. We will discuss the choice of \(t\) in the experiments section.

3 THEORETICAL ANALYSIS

In this section, we prove that cross&compress units have sufficient capability of polynomial approximation. We also show that MKR is a generalized framework over several representative methods of recommender systems and multi-task learning.

3.1 Polynomial Approximation

According to the Weierstrass approximation theorem [25], any function under certain smoothness assumption can be approximated by a polynomial to an arbitrary accuracy. Therefore, we examine the ability of high-order interaction approximation of the cross&compress unit. We show that cross&compress units can model the order of item-entity feature interaction up to exponential degree:

**Theorem 1.** Denote the input of item and entity in MKR network as \(v = [v_1 \cdots v_d]^T\) and \(e = [e_1 \cdots e_d]^T\), respectively. Then the cross terms about \(v\) and \(e\) in \(\|v\|_1\) and \(\|e\|_1\) (the L1-norm of \(v\) and \(e\)) with maximal degree is \(k_{\alpha, \beta} v_1^{\alpha_1} \cdots v_d^{\alpha_d} e_1^{\beta_1} \cdots e_d^{\beta_d}\), where \(k_{\alpha, \beta} \in \mathbb{R}, \alpha_i, \beta_i \in \mathbb{N}\) for \(i \in \{1, \cdots, d\}\), \(\alpha_1 + \cdots + \alpha_d = 2^{L-1}\), and \(\beta_1 + \cdots + \beta_d = 2^{L-1}\) for \(L \geq 1, v_0 = v, e_0 = e\).

In recommender systems, \(\prod_{i=1}^d v_i^{\alpha_i} e_i^{\beta_i}\) is also called combinatorial feature, as it measures the interactions of multiple original features. Theorem 1 states that cross&compress units can automatically model the combinatorial features of items and entities for sufficiently high order, which demonstrates the superior approximation capacity of MKR as compared with existing work such as Wide&Deep [3], factorization machines [22, 23] and DCN [34]. The proof of Theorem 1 is provided in the Appendix. Note that Theorem 1 gives a theoretical view of the polynomial approximation.
ability of the cross&compress unit rather than providing guarantees on its actual performance. We will empirically evaluate the cross&compress unit in the experiments section.

3.2 Unified View of Representative Methods

In the following we provide a unified view of several representative models in recommender systems and multi-task learning, by showing that they are restricted versions of or theoretically related to MKR. This justifies the design of cross&compress unit and conceptually explains its strong empirical performance as compared to baselines.

3.2.1 Factorization machines. Factorization machines [22, 23] are a generic method for recommender systems. Given an input feature vector, FMs model all interactions between variables in the input vector using factorized parameters, thus being able to estimate interactions in problems with huge sparsity such as recommender systems. The model equation for a 2-degree factorization machine is defined as

\[ \hat{y}(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle v_i, v_j \rangle x_i x_j, \]

(10)

where \( x_i \) is the \( i \)-th unit of input vector \( x \), \( w \) is weight scalar, \( v \) is weight vector, and \( \langle \cdot, \cdot \rangle \) is dot product of two vectors. We show that the essence of FM is conceptually similar to an 1-layer cross&compress unit:

Proposition 1. The L1-norm of \( v_1 \) and \( e_1 \) can be written as the following form:

\[ \|v_1\|_1 \text{ (or } \|e_1\|_1\| = b + \sum_{i=1}^{d} \sum_{j=1}^{d} \langle w_i, w_j \rangle v_i e_j, \]

(11)

where \( \langle w_i, w_j \rangle = w_i + w_j \) is the sum of two scalars.

It is interesting to notice that, instead of factorizing the weight parameter of \( x_i x_j \) into the dot product of two vectors as in FM, the weight of term \( v_i e_j \) is factorized into the sum of two scalars in cross&compress unit to reduce the number of parameters and increase robustness of the model.

3.2.2 Deep&Cross Network. DCN [34] learns explicit and high-order cross features by introducing the layers:

\[ x_{l+1} = x_0 x_l^\top w_l + x_l + b_l, \]

(12)

where \( x_j, w_l, \) and \( b_l \) are representation, weight, and bias of the \( l \)-th layer. We demonstrate the link between DCN and MKR by the following proposition:

Proposition 2. In the formula of \( v_{l+1} \) in Eq. (2), if we restrict \( w_l^\top V \) in the first term to satisfy \( e_l^\top w_l^\top V = 1 \) and restrict \( e_l \) in the second term to be \( e_0 \) (and impose similar restrictions on \( e_{l+1} \)), the cross&compress unit is then conceptually equivalent to DCN layer in the sense of multi-task learning:

\[ v_{l+1} = e_0 v_l^\top w_l^\top V + v_l + b_l^V, \]

\[ e_{l+1} = e_0 e_l^\top w_l^\top V + e_l + b_l^E. \]

(13)

It can be proven that the polynomial approximation ability of the above DCN-equivalent version (i.e., the maximal degree of cross terms in \( v_1 \) and \( e_1 \)) is \( O(1) \), which is weaker than original cross&compress units with \( O(2^l) \) approximation ability.

3.2.3 Cross-stitch Networks. Cross-stitch networks [18] is a multi-task learning model in convolutional networks, in which the designed cross-stitch unit can learn a combination of shared and task-specific representations between two tasks. Specifically, given two activation maps \( x_A \) and \( x_B \) from layer \( l \) for both the tasks, cross-stitch networks learn linear combinations \( x_A \) and \( x_B \) of both the input activations and feed these combinations as input to the next layers’ filters. The formula at location \((i, j)\) in the activation map is

\[ \begin{bmatrix} x_{ij}^A \\ x_{ij}^B \end{bmatrix} = \begin{bmatrix} a_{AA} & a_{AB} \\ a_{BA} & a_{BB} \end{bmatrix} \begin{bmatrix} x_{ij}^A \\ x_{ij}^B \end{bmatrix}, \]

(14)

where \( a \)'s are trainable transfer weights of representations between task \( A \) and task \( B \). We show that the cross-stitch unit in Eq. (14) is a simplified version of our cross&compress unit by the following proposition:

Proposition 3. If we omit all biases in Eq. (2), the cross&compress unit can be written as

\[ \begin{bmatrix} v_{l+1} \\ e_{l+1} \end{bmatrix} = \begin{bmatrix} e_l^V w_l^V + v_l \\ e_l^E w_l^E + e_l \end{bmatrix} \begin{bmatrix} v_l \\ e_l \end{bmatrix}. \]

(15)

The transfer matrix in Eq. (15) serves as the cross-stitch unit \( [a_{AA} \ a_{AB}; a_{BA} \ a_{BB}] \) in Eq. (14). Like cross-stitch networks, MKR network can decide to make certain layers task specific by setting \( v_l^V w_l^V (a_{AB}) \) or \( e_l^E w_l^E (a_{BA}) \) to zero, or choose a more shared representation by assigning a higher value to them. But the transfer matrix is more fine-grained in cross&compress unit, because the transfer weights are replaced from scalars to dot products of two vectors. It is rather interesting to notice that Eq. (15) can also be regarded as an attention mechanism [1], as the computation of transfer weights involves the feature vectors \( v_l \) and \( e_l \) themselves.

4 EXPERIMENTS

In this section, we evaluate the performance of MKR in four real-world recommendation scenarios: movie, book, music, and news.

4.1 Datasets

We utilize the following four datasets in our experiments:

- **MovieLens-1M** is a widely used benchmark dataset in movie recommendations, which consists of approximately 1 million explicit ratings (ranging from 1 to 5) on the MovieLens website.
- **Book-Crossing** dataset contains 1,149,780 explicit ratings (ranging from 0 to 10) of books in the Book-Crossing community.
- **Last.FM** dataset contains musician listening information from a set of 2 thousand users from Last.fm online music system.
- **Bing-News** dataset contains 1,025,192 pieces of implicit feedback collected from the server logs of Bing News.

---

9 The source code is available at https://github.com/hwwang55/MKR.
10 https://grouplens.org/datasets/movielens/1m/
11 http://www2.informatik.uni-freiburg.de/~cziegler/BX/
12 https://grouplens.org/datasets/hetrec-2011/
13 https://www.bing.com/news
October 16, 2016 to August 11, 2017. Each piece of news has a title and a snippet.

Since MovieLens-1M, Book-Crossing, and Last.FM are explicit feedback data (Last.FM provides the listening count as weight for each user-item interaction), we transform them into implicit feedback where each entry is marked with 1 indicating that the user has rated the item positively, and sample an unwatched set marked as 0 for each user. The threshold of positive rating is 4 for MovieLens-1M, while no threshold is set for Book-Crossing and Last.FM due to their sparsity.

We use Microsoft Satori to construct the KG for each dataset. We first select a subset of triples from the whole KG with a confidence level greater than 0.9. For MovieLens-1M and Book-Crossing, we additionally select a subset of triples from the sub-KG whose relation name contains “film” or “book” respectively to further reduce KG size.

Given the sub-KGs, for MovieLens-1M, Book-Crossing, and Last.FM, we collect IDs of all valid movies, books, or musicians by matching their names with tail of triples (head, film.film.name, tail), (head, book.book.title, tail), or (head, type.object.name, tail), respectively. For simplicity, items with no matched or multiple matched entities are excluded. We then match the IDs with the head and tail of all KG triples and select all well-matched triples from the sub-KG. The constructing process is similar for Bing-News except that: (1) we use entity linking tools to extract entities in news titles; (2) we do not impose restrictions on the names of relations since the entities in news titles are not within one particular domain. The basic statistics of the four datasets are presented in Table 1. Note that the number of users, items, and interactions are smaller than original statistics of the four datasets.

Table 1: Basic statistics and hyper-parameter settings for the four datasets.

| Dataset            | # users | # items | # interactions | # KG triples | Hyper-parameters  |
|--------------------|---------|---------|----------------|--------------|------------------|
| MovieLens-1M       | 6,036   | 2,347   | 753,772        | 20,195       | L = 1, d = 8, t = 3, λ₁ = 0.5 |
| Book-Crossing      | 17,860  | 14,910  | 139,746        | 19,793       | L = 1, d = 8, t = 2, λ₁ = 0.1 |
| Last.FM            | 1,872   | 3,846   | 42,346         | 15,518       | L = 2, d = 4, t = 2, λ₁ = 0.1 |
| Bing-News          | 141,487 | 535,145 | 1,025,192      | 1,545,217    | L = 3, d = 16, t = 5, λ₁ = 0.2 |

We apply CKE as CF plus structural knowledge module in this paper. The dimension of user and item embeddings for the four datasets are set as 64, 128, 32, 64, respectively. The dimension of entity embeddings is 32.

Figure 2: The correlation between the number of common neighbors of an item pair in KG and their number of common raters in RS.

4.2 Baselines

We compare our proposed MKR with the following baselines. Unless otherwise specified, the hyper-parameter settings of baselines are the same as reported in their original papers or as default in their codes.

- **PER** [39] treats the KG as heterogeneous information networks and extracts meta-path based features to represent the connectivity between users and items. In this paper, we use manually designed user-item-attribute-item paths as features, i.e., “user-movie-director-movie”, “user-movie-genre-movie”, and “user-movie-star-movie” for MovieLens-20M; “user-book-author-book” and “user-book-genre-book” for Book-Crossing; “user-musician-genre-musician”, “user-musician-country-musician”, and “user-musician-age-musician” (age is discretized) for Last.FM. Note that PER cannot be applied to news recommendation because it’s hard to pre-define meta-paths for entities in news.

- **CKE** [40] combines CF with structural, textual, and visual knowledge in a unified framework for recommendation. We implement CKE as CF plus structural knowledge module in this paper. The dimension of user and item embeddings for the four datasets are set as 64, 128, 32, 64, respectively. The dimension of entity embeddings is 32.

- **DKN** [32] treats entity embedding and word embedding as multiple channels and combines them together in CNN for CTR prediction. In this paper, we use movie/book names and news titles as textual input for DKN. The dimension of word embedding and entity embedding is 64, and the number of filters is 128 for each window size 1, 2, 3.

- **RippleNet** [31] is a memory-network-like approach that propagates users’ preferences on the knowledge graph for recommendation. The hyper-parameter settings for Last.FM are d = 8, H = 2, λ₁ = 10−6, λ₂ = 0.01, η = 0.02.

- **LibFM** [23] is a widely used feature-based factorization model. We concatenate the raw features of users and items as well as the corresponding averaged entity embeddings learned from TransR [13] as input for LibFM. The dimension is 1, 1, 8 and the number of training epochs is 50. The dimension of TransR is 32.

- **Wide&Deep** [3] is a deep recommendation model combining a (wide) linear channel with a (deep) nonlinear channel. The input for Wide&Deep is the same as in LibFM. The dimension of user, item, and entity is 64, and we use a two-layer deep channel with dimension of 100 and 50 as well as a wide channel.
We conduct an empirical study to investigate the correlation of CTR prediction and top-K recommendation. To this end, we first randomly sample 1 million item pairs from MovieLens-1M. We then classify each pair into 5 categories based on the number of their common raters in RS, and count their average click probability. We use AUC and Accuracy to evaluate the performance of CTR prediction. In top-K recommendation, we use the trained model to select K items with highest predicted click probability for each user in the test set, and choose Precision@K and Recall@K to evaluate the recommended sets.

### 4.3 Experiments setup
In MKR, we set the number of high-level layers $K = 1$, $f_{RS}$ as inner product, and $\lambda_2 = 10^{-6}$ for all three datasets, and other hyper-parameters are given in Table 1. The settings of hyper-parameters are determined by optimizing AUC on a validation set. For each dataset, the ratio of training, validation, and test set is 6 : 2 : 2. Each experiment is repeated 3 times, and the average performance is reported. We evaluate our method in two experiment scenarios: (1) In click-through rate (CTR) prediction, we apply the trained model to each piece of interactions in the test set and output the predicted click probability. We use AUC and Accuracy to evaluate the performance of CTR prediction. (2) In top-K recommendation, we use the trained model to select K items with highest predicted click probability for each user in the test set, and choose Precision@K and Recall@K to evaluate the recommended sets.

### 4.4 Empirical study
We conduct an empirical study to investigate the correlation of items in RS and their corresponding entities in KG. Specifically, we aim to reveal how the number of common neighbors of an item pair in KG changes with their number of common raters in RS. To this end, we first randomly sample 1 million item pairs from MovieLens-1M. We then classify each pair into 5 categories based on the number of their common raters in RS, and count their average number of common neighbors in KG for each category. The result is presented in Figure 2a, which clearly shows that if two items have more common raters in RS, they are likely to share more common neighbors in KG. Figure 2b shows the positive correlation from an opposite direction. The above findings empirically demonstrate that items share the similar structure of proximity in KG and RS, thus the cross knowledge transfer of items benefits both recommendation and KGE tasks in MKR.

### 4.5 Results

#### 4.5.1 Comparison with baselines
The results of all methods in CTR prediction and top-K recommendation are presented in Table 2 and Figure 3, 4, respectively. We have the following observations:

- PER performs poor on movie, book, and music recommendation because the user-defined meta-paths can hardly be optimal in reality. Moreover, PER cannot be applied to news recommendation.
- CKE performs better in movie, book, and music recommendation than news. This may be because MovieLens-1M, BookCrossing, and Last.FM are much denser than Bing-News, which is more favorable for the collaborative filtering part in CKE.
- DKN performs best in news recommendation compared with other baselines, but performs worst in other scenarios. This is because movie, book, and musician names are too short and ambiguous to provide useful information.
- RippleNet performs best among all baselines, and even outperforms MKR on MovieLens-1M. This demonstrates that RippleNet can precisely capture user interests, especially in the case where user-item interactions are dense. However, RippleNet is more sensitive to the density of datasets, as it performs worse than MKR in Book-Crossing, Last.FM, and Bing-News. We will further study their performance in sparse scenarios in Section 4.5.3.
- In general, our MKR performs best among all methods on the four datasets. Specifically, MKR achieves average Accuracy gains of 11.6%, 11.5%, 12.7%, and 8.7% in movie, book, music, and news recommendation, respectively, which demonstrates the efficacy of the multi-task learning framework in MKR. Note that the top-K metrics are much lower for Bing-News because the number of news is significantly larger than movies, books, and musicians.

#### 4.5.2 Comparison with MKR variants
We further compare MKR with its three variants to demonstrate the efficacy of cross & compress unit:

- MKR-1L is MKR with one layer of cross & compress unit, which corresponds to FM model according to Proposition 1. Note that MKR-1L is actually MKR in the experiments for MovieLens-1M.
- MKR-DCN is a variant of MKR based on Eq. (13), which corresponds to DCN model.
- MKR-stitch is another variant of MKR corresponding to the cross-stitch network, in which the transfer weights in Eq. (15) are replaced by four trainable scalars.

Table 2: The results of AUC and Accuracy in CTR prediction.

| Model   | MovieLens-1M | Book-Crossing | Last.FM | Bing-News |
|---------|--------------|---------------|---------|-----------|
|         | AUC | ACC | AUC | ACC | AUC | ACC | AUC | ACC |
| MKR     | 0.917 | 0.843 | 0.743 | 0.704 | 0.797 | 0.752 | 0.689 | 0.645 |
| MKR-1L  | - | - | - | - | 0.795 | (0.3%) | 0.749 | (0.4%) |
| MKR-DCN | 0.883 | (-3.7%) | 0.802 | (-4.9%) | 0.705 | (-4.3%) | 0.676 | (-4.2%) |
| MKR-stitch | 0.905 | (-1.3%) | 0.830 | (-1.5%) | 0.721 | (-2.2%) | 0.682 | (-3.4%) |

- MKR achieves average Accuracy gains of 11.6%, 11.5%, 12.7%, and 8.7% in movie, book, music, and news recommendation, respectively, which demonstrates the efficacy of the multi-task learning framework in MKR. Note that the top-K metrics are much lower for Bing-News because the number of news is significantly larger than movies, books, and musicians.
Figure 3: The results of Precision@K in top-K recommendation.

Figure 4: The results of Recall@K in top-K recommendation.

Table 3: Results of AUC on MovieLens-1M in CTR prediction with different ratios of training set r.

| Model    | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|
| PER      | 0.598 | 0.607 | 0.621 | 0.638 | 0.647 | 0.662 | 0.675 | 0.688 | 0.697 | 0.710 |
| CKE      | 0.674 | 0.692 | 0.705 | 0.716 | 0.739 | 0.754 | 0.768 | 0.775 | 0.797 | 0.801 |
| DKN      | 0.579 | 0.582 | 0.589 | 0.601 | 0.612 | 0.620 | 0.631 | 0.638 | 0.646 | 0.655 |
| RippleNet| 0.843 | 0.851 | 0.859 | 0.862 | 0.870 | 0.878 | 0.890 | 0.901 | 0.912 | 0.920 |
| LibFM    | 0.801 | 0.810 | 0.816 | 0.829 | 0.837 | 0.850 | 0.864 | 0.875 | 0.886 | 0.892 |
| Wide&Deep| 0.788 | 0.802 | 0.809 | 0.815 | 0.821 | 0.840 | 0.858 | 0.876 | 0.884 | 0.898 |
| MKR      | **0.868** | **0.874** | **0.881** | **0.882** | **0.889** | **0.897** | **0.903** | **0.908** | **0.913** | **0.917** |

From Table 2 we observe that MKR outperforms MKR-1L and MKR-DCN, which shows that modeling high-order interactions between item and entity features is helpful for maintaining decent performance. MKR also achieves better scores than MKR-stitch. This validates the efficacy of fine-grained control on knowledge transfer in MKR compared with the simple cross-stitch units.

4.5.3 Results in sparse scenarios. One major goal of using knowledge graph in MKR is to alleviate the sparsity and the cold start problem of recommender systems. To investigate the efficacy of the KGE module in sparse scenarios, we vary the ratio of training set of MovieLens-1M from 100% to 10% (while the validation and test set are kept fixed), and report the results of AUC in CTR prediction for all methods. The results are shown in Table 3. We observe that the performance of all methods deteriorates with the reduce of the training set. When $r = 10\%$, the AUC score decreases by 15.8%, 13.9%, 11.6%, 8.4%, 10.2%, 12.2% for PER, CKE, DKN, RippleNet, LibFM, and Wide&Deep, respectively, compared with the case when full training set is used ($r = 100\%$). In contrast, the
AUC score of MKR only decreases by 5.3%, which demonstrates that MKR can still maintain a decent performance even when the user-item interaction is sparse. We also notice that MKR performs better than RippleNet in sparse scenarios, which is in accordance with our observation in Section 4.5.1 that RippleNet is more sensitive to the density of user-item interactions.

### 4.5.4 Results on KGE side

Although the goal of MKR is to utilize KG to assist with recommendation, it is still interesting to investigate whether the RS task benefits the KGE task, since the principle of multi-task learning is to leverage shared information to help improve the performance of all tasks [42]. We present the result of RMSE (rooted mean square error) between predicted and real vectors of tails in the KGE task in Table 4. Fortunately, we find that the RS module can indeed reduce the prediction error by 1.9% ~ 6.4%. The results show that the cross&compress units are able to learn general and shared features that mutually benefit both sides of MKR.

### 5 RELATED WORK

#### 5.1 Knowledge Graph Embedding

The KGE module in MKR connects to a large body of work in KGE methods. KGE is used to embed entities and relations in a knowledge-into low-dimensional vector spaces while still preserving the structural information [33]. KGE methods can be classified into the following two categories: (1) Translational distance models exploit distance-based scoring functions when learning representations of entities and relations, such as TransE [2], TransH [35], and TransR [13]; (2) Semantic matching models measure plausibility of knowledge triples by matching latent semantics of entities and relations, such as RESCAL [20], ANALOGY [19], and HolE [14]. Recently, researchers also propose incorporating auxiliary information, such as entity types [36], logic rules [24], and textual descriptions [46] to assist KGE. The above KGE methods can also be incorporated into MKR as the implementation of the KGE module, but note that the cross&compress unit in MKR needs to be redesigned accordingly. Exploring other designs of KGE module as well as the corresponding bridging unit is also an important direction of future work.

#### 5.2 Multi-Task Learning

Multi-task learning is a learning paradigm in machine learning and its aim is to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks [42]. All of the learning tasks are assumed to be related to
each other, and it is found that learning these tasks jointly can lead to performance improvement compared with learning them individually. In general, MTL algorithms can be classified into several categories, including feature learning approach [34, 41], low-rank approach [7, 16], task clustering approach [47], task relation learning approach [12], and decomposition approach [6]. For example, the cross-stitch network [41] determines the inputs of hidden layers in different tasks by a knowledge transfer matrix; Zhou et al. [47] aims to cluster tasks by identifying representative tasks which are a subset of the given m tasks, i.e., if task T_j is selected by task T_j as a representative task, then it is expected that model parameters for T_j are similar to those of T_j. MTL can also be combined with other learning paradigms to improve the performance of learning tasks further, including semi-supervised learning, active learning, unsupervised learning, and reinforcement learning.

Our work can be seen as an asymmetric multi-task learning framework [37, 43, 44], in which we aim to utilize the connection between RS and KG to help improve their performance, and the two tasks are trained with different frequencies.

5.3 Deep Recommender Systems

Recently, deep learning has been revolutionizing recommender systems and achieves better performance in many recommendation scenarios. Roughly speaking, deep recommender systems can be classified into two categories: (1) Using deep neural networks to process the raw features of users or items [5, 28–30, 40]; For example, Collaborative Deep Learning [29] designs autoencoders to extract short and dense features from textual input and feeds the features into a collaborative filtering module; DeepFM [5] combines factorization machines for recommendation and deep learning for feature learning in a neural network architecture. (2) Using deep neural networks to model the interaction among users and items [3, 4, 8, 9]. For example, Neural Collaborative Filtering [8] replaces the inner product with a neural architecture to model the user-item interaction. The major difference between these methods and ours is that MKR deploys a multi-task learning framework that utilizes the knowledge from a KG to assist recommendation.

6 CONCLUSIONS AND FUTURE WORK

This paper proposes MKR, a multi-task learning approach for knowledge graph enhanced recommendation. MKR is a deep and end-to-end framework that consists of two parts: the recommendation module and the KGE module. Both modules adopt multiple nonlinear layers to extract latent features from inputs and fit the complicated interactions of user-item and head-relation pairs. Since the two tasks are not independent but connected by items and entity features and transfer knowledge between the two tasks. We conduct extensive experiments in four recommendation scenarios. The results demonstrate the significant superiority of MKR over strong baselines and the efficacy of the usage of KG.

For future work, we plan to investigate other types of neural networks (such as CNN) in MKR framework. We will also incorporate other KGE methods as the implementation of KGE module in MKR by redesigning the cross&compress unit.

APPENDIX

A Proof of Theorem 1

Proof. We prove the theorem by induction:

Base case: When l = 1,

\[ v_1 = \mathbf{e}^T \mathbf{w}_0^V + \mathbf{e}^T \mathbf{w}_0^E + b_0^V \]

\[ = \left[ \begin{array}{c} v_1 \sum_{i=1}^{d} e_i^V w_0^{V(i)} \cdots v_d \sum_{i=1}^{d} e_i^V w_0^{V(i)} \end{array} \right]^T + \left[ \begin{array}{c} e_1 \sum_{i=1}^{d} v_i^E w_0^{V(i)} \cdots e_d \sum_{i=1}^{d} v_i^E w_0^{V(i)} \end{array} \right]^T + \left[ \begin{array}{c} v_0^V(0) \cdots b_0^V (d) \end{array} \right]^T. \]

Therefore, we have

\[ ||v_1||_1 = \sum_{i,j=1}^{d} (\sum_{i=1}^{d} e_i^V w_0^{V(i)}) + \sum_{i=1}^{d} (\sum_{i=1}^{d} v_i^E w_0^{V(i)}) + \sum_{i=1}^{d} b_0^V (d) \]

It is clear that the cross terms about v and e with maximal degree is \( k_{x^a, y^a} e(v, e) \), so we have \( a_1 + \cdots + a_d = 1 = 2^{l-1} \), and \( b_1 + \cdots + b_d = 1 = 2^{l-1} \) for \( v_1 \). The proof for \( e_1 \) is similar.

Induction step: Suppose \( a_1 + \cdots + a_d = 2^{l-1} \) and \( b_1 + \cdots + b_d = 2^{l-1} \) hold for the maximal-degree term x and y in ||\( v_l \)||_1 and ||\( e_l \)||_1. Since ||\( v_l \)||_1 = \( \sum_{i=1}^{d} v_l^{(i,j)} \) and ||\( e_l \)||_1 = \( \sum_{i=1}^{d} e_l^{(i,j)} \) without loss of generality, we assume that x and y exist in \( v_l^{(a)} \) and \( e_l^{(b)} \), respectively. Then for \( l + 1 \), we have

\[ ||v_{l+1}||_1 = \sum_{i=1}^{d} d_{l+1} (w_0^{V(i)}) + w_0^{V(j)} v_l^{(i,j)} + b_0^V (d). \]

Obviously, the maximal-degree term in ||\( v_{l+1} \)||_1 is the cross term \( x y \) in \( v_l^{(a)} (b) \). Since we have \( a_1 + \cdots + a_d = 2^{l-1} \) and \( b_1 + \cdots + b_d = 2^{l-1} \) for both x and y, the degree of cross term \( x y \) therefore satisfies \( a_1 + \cdots + a_d = 2^{(l+1)-1} \) and \( b_1 + \cdots + b_d = 2^{(l+1)-1} \). The proof for ||\( e_{l+1} \)||_1 is similar. \( \square \)

B Proof of Proposition 1

Proof. In the proof of Theorem 1 in Appendix A, we have shown that

\[ ||v_1||_1 = \sum_{i=1}^{d} d_{l+1} (w_0^{V(i)}) + w_0^{V(j)} v_l^{(i,j)} + b_0^V (d). \]

It is easy to see that \( w_i = w_0^{V(i)} \), \( w_j = w_0^{V(j)} \), and \( b = \sum_{i=1}^{d} b_0^V (d) \). The proof is similar for ||\( e_1 \)||_1. \( \square \)

We omit the proofs for Proposition 2 and Proposition 3 as they are straightforward.
