

# Jointly Learning Explainable Rules for Recommendation with Knowledge Graph

Weizhi Ma<sup>†</sup>, Min Zhang<sup>†\*</sup>, Yue Cao<sup>‡</sup>, Woojeong Jin<sup>‡</sup>, Chenyang Wang<sup>†</sup>,

Yiqun Liu<sup>†</sup>, Shaoping Ma<sup>†</sup>, Xiang Ren<sup>‡\*</sup>

<sup>†</sup> Department of Computer Science and Technology, Institute for Artificial Intelligence  
Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China

<sup>‡</sup> Department of Computer Science, University of Southern California, Los Angeles, CA, USA  
mawz14@mails.tsinghua.edu.cn, {z-m, yiqunliu, msp}@tsinghua.edu.cn,  
{cao517, woojeong.jin, xiangren}@usc.edu, thuwangcy@gmail.com

## ABSTRACT

Explainability and effectiveness are two key aspects for building recommender systems. Prior efforts mostly focus on incorporating side information to achieve better recommendation performance. However, these methods have some weaknesses: (1) prediction of neural network-based embedding methods are hard to explain and debug; (2) symbolic, graph-based approaches (e.g., meta path-based models) require manual efforts and domain knowledge to define patterns and rules, and ignore the item association types (e.g. substitutable and complementary). In this paper, we propose a novel joint learning framework to integrate *induction of explainable rules from knowledge graph* with *construction of a rule-guided neural recommendation model*. The framework encourages two modules to complement each other in generating effective and explainable recommendation: 1) inductive rules, mined from item-centric knowledge graphs, summarize common multi-hop relational patterns for inferring different item associations and provide human-readable explanation for model prediction; 2) recommendation module can be augmented by induced rules and thus have better generalization ability dealing with the cold-start issue. Extensive experiments<sup>1</sup> show that our proposed method has achieved significant improvements in item recommendation over baselines on real-world datasets. Our model demonstrates robust performance over “noisy” item knowledge graphs, generated by linking item names to related entities.

## ACM Reference Format:

Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *Proceedings of the 2019 World Wide Web Conference (WWW’19)*, May 13–17, 2019, San Francisco, CA, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3308558.3313607>

## 1 INTRODUCTION

Recommender systems play an essential part in improving user experiences on online services. While a well-performed recommender system largely reduce human efforts in finding things of interests,

<sup>1</sup>Code and data can be found at: <https://github.com/THUIR/RuleRec>

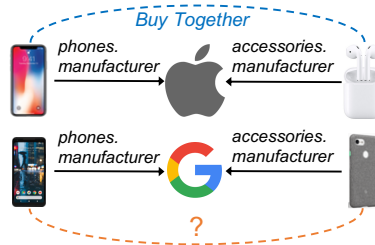
This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

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

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

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

<https://doi.org/10.1145/3308558.3313607>



**Figure 1: Illustration of Item-item Associations in a Knowledge Graph.** Given items, relations and item associations (e.g. *Buy Together*), our goal is to induce rules from them and recommend items from rules. These rules are used to infer associations between new items, recommend items, and explain the recommendation.

often times there may be some recommended items that are unexpected for users and cause confusion. Therefore, explainability becomes critically important for the recommender systems to provide convincing results—this helps to improve the effectiveness, efficiency, persuasiveness, transparency, and user satisfaction of recommender systems [45].

Though there are many powerful neural network-based recommendation algorithms proposed these years, most of them are unable to give explainable recommendation results [12, 14, 19]. Existing explainable recommendation algorithms are mainly two types: user-based [25, 33] and review-based [11, 46]. However, both of them are suffering from data sparsity problem, it is very hard for them to give clear reasons for the recommendation if the item lacks user reviews or the user has no social information.

On another line of research, some recommendation algorithms try to incorporate knowledge graphs, which contain lots of structured information, to introduce more features for the recommendation. There are two types of works that utilize knowledge graphs to improve recommendation: meta-path based methods [32, 43, 48] and embedding learning-based algorithms [24, 31, 44]. However, meta-path based methods require manual efforts and domain knowledge to define patterns and paths for feature extraction. Embedding based algorithms use the structure of the knowledge graph to learn users’ and items’ feature vectors for the recommendation, while the recommendation results are unexplainable. Besides, both types of algorithms ignore item associations.

We find that associations between items/products can be utilized to give accurate and explainable recommendation. For example,

\*Corresponding author

if a user buys a cellphone, it makes sense to recommend him/her some cellphone chargers or cases (as they are complementary items of the cellphone). But it may cause negative experiences if the system shows him/her other cellphones immediately (substitute items) because most users will not buy another cellphone right after buying one. So we can use this signal to tell users why we recommend an item for a user with explicit reasons (even for cold items). Furthermore, we propose that an idea to make use of item associations: After mapping the items into a knowledge graph, there will be multi-hop relational paths between items. Then, We can summarize explainable rules from for predicting association relationships between each two items and the induced rules will also be helpful for the recommendation.

To shed some light on this problem, we propose a novel joint learning framework to give accurate and explainable recommendations. The framework consists of a rule learning module and a recommendation module. We exploit knowledge graphs to induce explainable rules from item associations in the rule learning module and provide rule-guided recommendations based on the rules in the recommendation module. Fig. 1 shows an example of items with item associations in a knowledge graph. Note the knowledge graph here is constructed by linking items into a real knowledge graph, but not a heterogeneous graph that only consists of items and their attributes. The rule learning module leverage relations in a knowledge graph to summarize common rule patterns from item associations, which is explainable. The recommendation module combines existing recommendation models with the reduced rules, thus have a better ability to deal with the cold-start problem and give explainable recommendations. Our proposed framework outperforms baselines on real-world datasets from different domains. Furthermore, it gives an explainable result with the rules.

Our main contributions are listed as follows:

- We utilize a large-scale knowledge graph to derive rules between items from item associations.
- We propose a joint optimization framework that induces rules from knowledge graphs and recommends items based on the rules at the same time.
- We conduct extensive experiments on real-world datasets. Experimental results prove the effectiveness of our framework in accurate and explainable recommendation

## 2 PRELIMINARIES

We firstly introduce concepts and give a formal problem definition. Then, we briefly review BPRMF [27] and NCF [14] algorithms.

### 2.1 Background and Problem

**Item recommendation.** Given users  $U$  and items  $I$ , the task of item recommendation aims to identify items that are most suitable for each user based on historical interactions between users and items (e.g. purchase history). A user expresses his or her preferences by purchasing or rating items. These interactions can be represented as a matrix. One of the promising approaches is a matrix factorization method which embeds users and items into a low dimensional latent space. This method decomposes the user-item interaction matrix into the product of two lower dimensional rectangular matrices  $U$  and  $I$  for a user and an item, respectively. From these matrices, we can recommend new items to users.

**Knowledge graph.** A knowledge graph is a multi-relational graph that composed of entities as nodes and relations  $r$  as different types

edges  $e$ . We can use many triples (head entity  $E_1$ , relation type  $r_1$ , tail entity  $E_2$ ) to represent the facts in the knowledge graph [38].

**Inductive rules on knowledge graph.** There are several paths between two entities in the knowledge graph, and a path is consisted of entities with the relation types (e.g.  $P_k = E_1 r_1 E_2 r_2 E_3$  is a path between  $E_1$  and  $E_3$ ). A rule  $R$  is defined by the relation sequence between two entities, e.g.  $R = r_1 r_2$  is a rule. The difference between paths and rules is that rules focus on the relation types, not entities.

**Problem Definition.** Our study focus on jointly learning rules in a knowledge graph and a recommender system with the rules. Formally, our problem is defined as follows:

*Definition 2.1 (Problem Definition).* **Given** users  $U$ , items  $I$ , user-item interactions, item associations, and a knowledge graph, our framework **aims to** jointly (1) learn rules  $R$  between items based on item associations and (2) learn a recommender system to recommend items  $I'_u$  to each user  $u$  based on the rules  $R$  and his/her interaction history  $I_u$ . This framework outputs a set of rules  $R$  and recommended item lists  $I'$ .

### 2.2 Base Models for Recommendation

The framework proposed in our study is flexible to work with different recommendation algorithms. As BPRMF is a widely used classical matrix factorization algorithm and NCF is a state-of-the-art neural network based recommendation algorithm, we choose to modify them to verify the effectiveness of our framework.

**Bayesian Personalized Ranking Matrix Factorization (BPRMF).**

Matrix Factorization based algorithms play a vital role in recommender systems. The idea is to represent each user/item with a vector of latent features.  $U$  and  $I$  are user feature matrix and item feature matrix respectively, and we use  $U_u$  to denote the feature vector of user  $u$  ( $I_i$  for item  $i$ ). The dimensions of them are the same. In BPRMF algorithm [27], the preference score  $S_{u,i}$  between  $u$  and  $i$  is computed by the inner product of  $U_u$  and  $I_i$ :

$$S_{u,i} = U_u^T \cdot I_i \quad (1)$$

The objective function of BPRMF algorithm is defined as a pair-wise function as follows:

$$O_{BPRMF} = \sum_{u \in U} \sum_{p \in I_u, n \notin I_u} (S_{u,p} - S_{u,n}) \quad (2)$$

where  $p$  is a positive item that user  $u$  interacted before, and  $n$  is a negative item sampled randomly from the items user  $u$  has never interacted ( $n$  should not be in test set too).

**Neural Collaborative Filtering (NCF).** NCF [14] is a neural based matrix factorization algorithm. Similar to BPRMF, each user  $u$  and each item  $i$  has a corresponding feature vector  $U_u$  and  $I_i$ , respectively. NCF propose a generalized matrix factorization (GMF) (Eq (3)) and a non-linear interaction part via a multi-layer perception (MLP) (Eq (4)) between user and item to extraction.

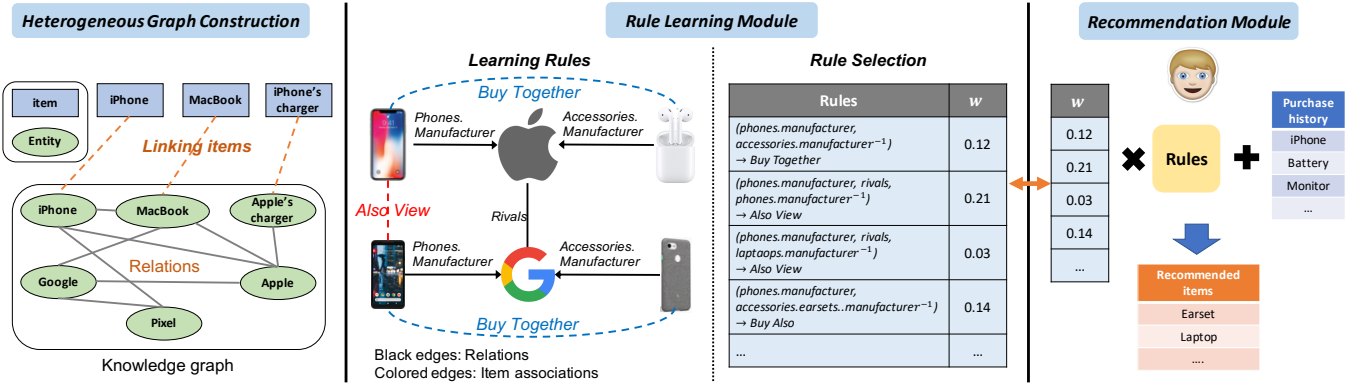
$$h_{u,i} = U_u^T \cdot I_i \quad (3)$$

$$g_{u,i} = \phi_n(\dots\phi_2(\phi_1(z_1)))$$

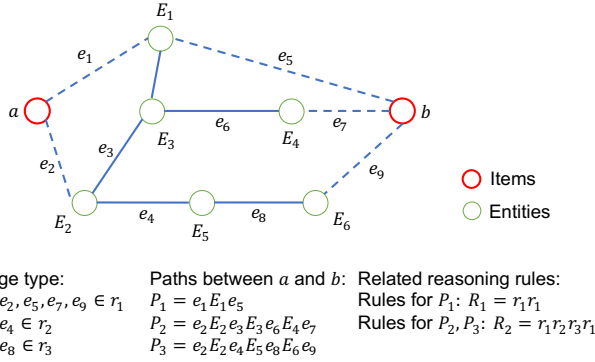
$$z_1 = \phi_0(U_u \oplus I_i) \quad (4)$$

$$\phi_k(z_{k-1}) = \phi_k(W_k^T z_{k-1} + b_{k-1}),$$

where  $n$  is the number of hidden layers.  $W_k$ ,  $b_l$ , and  $z_k$  are weight matrices, bias vector, and output of each layer.  $\oplus$  is vector concatenation and  $\phi$  is a non-linear activation function. Both  $h_{u,i}$  and  $g_{u,i}$  are user-item interaction feature vectors for GMF and MLP,



**Figure 2: Overview of the Proposed RuleRec Framework.** First, we build a heterogeneous graph from items and a knowledge graph. The rule learning module learns the importance of rules and the recommendation module learns the importance at the same time by sharing a parameter vector  $w$ .



**Figure 3:** An example of a heterogeneous graph which consists of items and entities in a knowledge graph. The dashed lines are links between items and entities generated by an entity linking algorithm.

respectively. The prediction equation of NCF is defined in Eq (5), in which the outputs of GMF and MLP parts are concatenated to get the final score. And we modified the objective function of NCF into Eq (6) in this paper.

$$S_{u,i} = \phi(\alpha \cdot h_{u,i} \oplus (1 - \alpha) \cdot g_{u,i}) \quad (5)$$

$$O_{NCF} = \sigma\left(\sum_{u \in U} \sum_{p \in I_u, n \notin I_u} (S_{u,p} - S_{u,n})\right) \quad (6)$$

### 3 THE RULEREC FRAMEWORK

**Framework Overview.** Recommendation with rule learning consists of two sub-tasks: 1) rule learning in a knowledge graph based on item associations; 2) recommending items for each user  $u$  with his/her purchase history  $I_u$  and the derived rules  $R$ .

To cope with these tasks, we design a multi-task learning framework. The framework consists of two modules, a rule learning module and a recommendation module. The rule learning module aims to derive useful rules through reasoning rules with ground-truth item associations in the knowledge graph. Based on the rule set, we can generate an item-pair feature vector whose each entry is an encoded value of each rule. The recommendation module takes the item-pair feature vector as additional input to enhance

recommendation performances and give explanations for the recommendation. We introduce a shared rule weight vector  $w$  which indicates the importance of each rule in predicting user preference, and shows the effectiveness of each rule in predicting item pair associations. Besides, based on the assume that useful rules perform consistently in both modules with higher weights, we design a objective function to conduct jointly learning:

$$\min_{V, W} O = \min_{V, W} \{O_r + \lambda O_l\} \quad (7)$$

where  $V$  denotes the parameters of the recommendation module, and  $W$  represents the shared parameters of the rule learning and the recommendation module. The objective function consists of two terms:  $O_r$  is the objective of the recommendation module, which recommends items based on the induced rules.  $O_l$  is the objective of the rule learning module, in which we leverage the given item associations to learn useful rules.  $\lambda$  is a trade-off parameter.

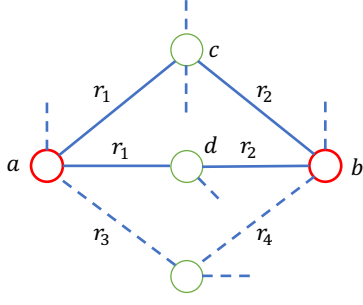
#### 3.1 Heterogeneous Graph Construction

First, we build a heterogeneous graph containing items for the recommendation and a knowledge graph. For some items, we can conduct exactly mapping between the item and the entity, such as "iPhone", "Macbook". For other items, it is hard to find an entity that represents the items, such iPhone's charger. Thus, we adopt entity linking algorithm [6] to find the related entities of an item from its title, brand, and description in the shopping website. In this way, we can add new nodes to the knowledge graph that represents items and add some edges for it according to entity linking results. Then, we get a heterogeneous graph which contains the items and the original knowledge graph. Fig. 3 is an example.

#### 3.2 Rule Learning Module

The rule learning module aims to find the reliable rule set  $R_A$  associated with given item associations  $A$  in the heterogeneous graph.

**Rule learning.** For any item pair  $(a, b)$  in the heterogeneous graph, we use a random walk based algorithm to compute the probabilities of finding paths which follow certain rules between the item pair, similar to [16, 17]. Then, we obtain feature vectors for item pairs. Each entry of the feature vector is the probability of a rule between the item pair. Here, we focus on relation types between the item pair to obtain rules such as  $R_1$  in Fig. 3, because it is general to the entities to capture the rules between items.



**Figure 4:** An example of a graph between items  $a$  and  $b$ .  $r$  represents a edge type or a relation type.

First, we define the probability of a rule between an item pair. Given a rule  $R = r_1 \dots r_k$ , probability  $P$  with the rule from  $a$  to  $b$  is defined as:

$$P(b|a, R) = \sum_{e \in N(a, R')} P(e|a, R') \cdot P(b|e, r_k), \quad (8)$$

where  $R' = r_1 \dots r_{k-1}$ , and  $P(b|e, r_k) = \frac{I(r_k(e, b))}{\sum_i I(r_k(e, i))}$  is the probability of reaching node  $b$  from node  $e$  with a one-step random walk with relation  $r_k$ .  $I(r_k(e, b))$  is 1 if there exists a link with relation  $r_k$  from  $e$  to  $b$ , otherwise 0. If  $b = e$ , then  $P(b|e, r_k) = 1$  for any  $r_k$ .  $N(a, R')$  denotes a node set that can be reached with rule  $R'$  from node  $a$ . For example,  $P(b|a, R)$  with a rule  $R = r_1 r_2$  in Fig. 4 is computed as follows:

$$P(b|a, R) = P(c|a, r_1) \cdot P(b|c, r_2) + P(d|a, r_1) \cdot P(b|d, r_2)$$

Second, we define a feature vector between an item pair. Given a set of rules, a rule feature vector for an item pair  $(a, b)$  is defined as  $x_{(a, b)} = [P(b|a, R_1), \dots, P(b|a, R_n)]^T$ . Each entry in the feature vector  $x_{(a, b)}$  represents a encoded value of rule  $R_i$  between  $a$  and  $b$ .

**Rule selection.** To select the most useful rules from the derived rules, we will introduce two types of selection methods: hard-selection and soft-selection.

**Hard-selection method.** Hard-selection method set a hyper parameter to decide how many rules we want to select with a selection algorithm firstly. Then we use a chi-square method and a learning based method to choose  $n$  rules in this study:

(1) Chi-square method. In statistics, the chi-square test is applied to measures dependence between two stochastic variables  $A$  and  $B$  (9) (to test if  $P(AB) = P(A)P(B)$ ).  $N_{A, B}$  is the observed occurrence of two events from a dataset and  $E_{A, B}$  is the expected frequency. In feature selection, as the features that have lower chi-square scores are independent of prediction target are likely to be useless for classification, chi-square scores between each column of feature vector  $(x_{(a, b)})$  and prediction target  $(y_{a, b|A})$  are used to select the top  $n$  useful features [29].

$$\chi_{A, B}^2 = \sum \frac{(N_{A, B} - E_{A, B})^2}{E_{A, B}} \quad (9)$$

(2) Learning based method. Another way to conduct feature selection is to design a objective function  $O_I$  that compute importance of each rule and try to minimize it. In the objective function, we introduce a weight vector  $w$  whose each entry represents importance of each rule. For an item pair  $(a, b)$ , we use  $y_{a, b|A}$  to denote

whether  $a$  and  $b$  have association  $A$  ( $y_{a, b|A}$  is 1 if they have, and 0 otherwise.). We define the following objective functions:

- Chi-square objective function

$$\sum_{\text{all pairs } s \in A} \sum_{i=0}^{|x_{(a, b)}|} w_i \cdot (x_{(a, b)}(i) + b - y_{a, b|A})^2 \quad (10)$$

- Linear regression objective function

$$\sum_{\text{all pairs } s \in A} \sum_{i=0}^{|x_{(a, b)}|} (w_i \cdot x_{(a, b)}(i) + b - y_{a, b|A})^2 \quad (11)$$

- Sigmoid objective function

$$\sum_{\text{all pairs } s \in A} \sum_{i=0}^{|x_{(a, b)}|} \frac{w_i}{1 + e^{-|x_{(a, b)}(i) + b - y_{a, b|A}|}} \quad (12)$$

where  $x_{(a, b)}(i)$  is  $i$ -th entry of  $x_{(a, b)}$ . To make the objective function reasonable, we constrain that  $\sum_i w_i = 1$  and  $w_i > 0$ . In training steps, if  $x_{(a, b)}(i)$  shows positive correlation with  $y_{a, b|A}$ , then rule  $i$  is likely to be useful for item association classification and will get higher weight according to the loss functions. So similar to the chi-square method, the top weighted rules will be selected.

**Soft-selection method.** Besides the hard-selection method, another way to make use of the learning based objective functions is to take the weight of each rule as a constrain on the rules weights in the recommendation module. No rule will be removed from rule set in this way and it will not introduce extra hyper-parameter. Due to this method is flexible to be combined into other part, we introduce the soft-selection method with learning based objective functions to the recommendation module as a multi-task learning. In such condition, there is no extra constrain on rule weight ( $\sum_i w_i = 1$  or  $w_i > 0$ ). The detail of the multi-task learning method will be shown in Section 3.5.

As the rule set is derived from an item association in rule learning module. To apply different item associations at the same time, we can combine the rule sets from different item associations together to get a global rule set  $R$ .

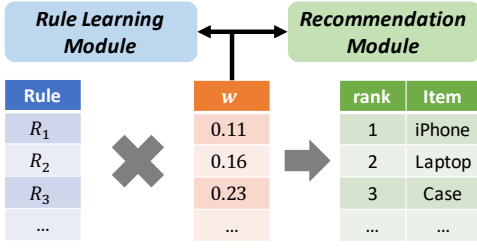
### 3.3 Item Recommendation Module

We propose a general recommendation module than can be combined with existing methods. This module utilizes the derived rule features to enhance recommendation performances.

The goal of this module is to predict an item list for user  $u$  based on the item set  $I_u$  s/he interacted (e.g. purchased) before. Previous works calculate the preference score  $S_{u, i}$  of user  $u$  purchase candidate item  $i$ , and then rank all candidate items with their scores to get the final recommendation list. As shown in Eq (13), we propose a function  $f_w$  parameterized by the shared weight vector  $w$  to combine the score  $S_{u, i}$  with rule features between candidate item  $i$  and items user interacted (e.g. purchased) under rule set  $R$ . A score  $S'_{u, i}$  for our method is defined as:

$$S'_{u, i} = f_w(S_{u, i}, \sum_{k \in I_u} F_{(i, k|R)}) \quad (13)$$

The feature vector for item pair  $(a, b)$  under rule set  $R$  is denoted by  $F_{(a, b|R)}$ . Note that  $F_{(a, b|R)}$  is different from  $x_{(a, b)}$  and calculated by  $F_{(a, b|R)} = \sum_{e \in N(a, R')} P(e|a, R') \cdot I(b|e, r_k)$ .  $I(b|e, r_k)$  is an indicator function: if there is a edge in relation type  $r_k$  between  $b$  and  $e$ ,  $I(b|e, r_k) = 1$ ; otherwise 0. The reason why we adopt another feature generation method is that in recommendation module, we



**Figure 5:** Multi-task learning of the rule learning module and the recommendation module. These two modules share the parameter  $w$ .

concerns more about if there exists a path in this rule between two items. The weight of each rule will be used in explaining the recommendation result, so we should make the comparing between rules fair. While longer rules are more likely to get lower score (more random walk steps so lower probability). If the feature vector is still  $x$ , it will hurt the explainable of our module. Thus we use  $F_{(a,b|R)}$  as the feature vector here, which represents the frequency of each rule between the two items.

To consider the global item associations between candidate item  $i$  and the item set  $I_u$ , we add the rule features between  $i$  and each item  $I_k$  in  $I_u$  together. For convenience, the new feature vector is named as  $F_{(i,I_u|R)}$ . So Eq (13) can be rewrite as the following:

$$S'_{u,i} = f_w(S_{u,i}, F_{(i,I_u|R)}) \quad (14)$$

We define the objective function for the recommendation module as follows:

$$\begin{aligned} O_r &= \sum_{u \in U} \sum_{p \in I_u, n \notin I_u} (S'_{u,p} - S'_{u,n}) \\ &= \sum_{u \in U} \sum_{p \in I_u, n \notin I_u} (f_w(S_{u,p}, F_{(p,I_u|R)}) - f_w(S_{u,n}, F_{(n,I_u|R)})), \end{aligned} \quad (15)$$

where  $p$  is a positive item ( $\in I_u$ ) and  $n$  is a random sampled negative item ( $\notin I_u$ ) for user  $u$ . Note that the rule weight vector  $w$  gives explanations for item pairs with rules in recommendation module. If a candidate item  $i$  gets a higher score than other candidate items, the rule which contributes the highest score for  $i$  and the corresponding items the user bought can be used to explain why the algorithm recommends  $i$  to the user. In other words, the introduction of rule features make the recommendation results explainable. There are some case studies in Section 5.5.

This combination method is flexible and easy to introduce rule features to many previous recommendation models (use the algorithm's prediction function to calculate  $S_{u,i}$ ). In this study, we implement this recommendation module with BPRMF (traditional MF algorithm) and NCF (neural network based MF algorithm). Since it is a two step algorithm (to learn rules firstly and then conduct recommendation), we denote them as RuleRec<sub>tw0</sub>(BPRMF) and RuleRec<sub>tw0</sub>(NCF). The prediction function and objective function of them are Eq (14) and Eq (15), where  $S_{u,i}$  is replaced by the prediction function of BPRMF (Eq (1)) and NCF (Eq (5)), respectively.

### 3.4 Multi-task Learning

In Sections 3.2 and 3.3, we introduced the two modules respectively. We can train the modules one by one to get the recommendation results. The shortcoming of training two modules separately is that

the usefulness of rules in prediction item association is ignored. Instead, we share the rule weight  $w$ , and this weight can capture the importance of the rule in both the recommendation and item association prediction simultaneously as shown in Fig. 5. Thus, we propose a multi-task learning objective function defined as follows:

$$O = O_r + \lambda O_l \quad (16)$$

where  $O_l$  and  $O_r$  are the objective functions for the rule learning module and the recommendation module, respectively. Note that both objective functions share  $w$ .

The multi-task learning combination method is able to conduct rule selection and recommendation model learning together. Similar to the two step combination method, it is also flexible to multiple recommendation models too. BPRMF and NCF are enhanced with this idea, and the modified algorithms are named as RuleRec<sub>multi</sub>(BPRMF) and RuleRec<sub>multi</sub>(NCF).

## 4 RULE SELECTION DETAILS

This section introduces the implementation details and results of the rule selection component in RuleRec.

### 4.1 Dataset and Implementation Details

We introduce item association datasets, a knowledge graph, and recommendation datasets for experiments.

**Item association datasets.** A open dataset with item associations is used in our experiments<sup>2</sup>. The item associations are extracted from user log on Amazon (same as [20]). Four types of item associations are considered: 1) Also view (ALV), users who viewed  $x$  also viewed  $y$ ; 2) Buy after view (BAV), users who viewed  $x$  eventually bought  $y$ ; 3) Also buy (ALB), users who bought  $x$  also bought  $y$ ; 4) Buy together (BT), users frequently bought  $x$  and  $y$  together. ALV and BAV are substitute associations, and ALB and BT are complementary associations. The statistics of Cellphone and Electronics datasets with different item associations are shown in Table 1. Since the data is crawled from Amazon<sup>3</sup>, the number of link is nearly ten times as large as the number of involved items in each association type. Besides, as shown in the table, over 37% items do not have any association with other items in this dataset.

**Knowledge graph dataset.** Freebase [2] is used to learn rules. It is the largest open knowledge graph<sup>4</sup>, containing more than 224M entities, 784K relation types, and over 1.9 billion links.

The link prediction algorithm<sup>5</sup> [6] is used to connect items (with their titles, brands, and descriptions) and entities in DBpedia<sup>6</sup> firstly. Then the linked entities in DBpedia are mapped to the entities in Freebase with a entity dictionary<sup>7</sup>. As there is a probability score of each linked entity with the algorithm, which represents the confidence of this linking. So if the probability of a word links to a entity is lower than 0.6, we will ignore it to make the link result more accurate.

Due to the large scale of the knowledge graph, it is infeasible to enumerate all possible rules in this step. Following the idea in [17], we require that all derived rule needs to be supported by at least a fraction  $\alpha$  of the training item pairs, as well as being of length no more than  $\beta$  (there will be huge number of rules without the

<sup>2</sup><http://jmcauley.ucsd.edu/data/amazon/>

<sup>3</sup>[www.amazon.com](http://www.amazon.com)

<sup>4</sup><https://developers.google.com/freebase/>

<sup>5</sup><http://model.dbpedia-spotlight.org/en/annotate>

<sup>6</sup><https://wiki.dbpedia.org/>

<sup>7</sup><https://drive.google.com/file/d/0Bw2KHcvHhx-gQ2RJVJLShJGYlk/view>

**Table 1: The statistics of item association pairs in different domain.** #Involved item means the number of items that have at least one type of association with any other items. #Item is the number of involved items with association and #Pair is the number of item pairs with association.

Dataset	#Item	#Involved Item	Also View		Buy After Viewing		Also Buy		Buy Together	
			#item	#Pair	#item	#Pair	#item	#Pair	#item	#Pair
Cellphone	346,793	214,692	103,845	1,038,090	181,935	1,818,990	71,660	716,240	29,372	293,360
Electronic	498,196	318,922	123,959	1,239,230	250,409	2,503,730	159,562	1,595,260	31,040	310,040

**Table 2: The number of derived rules from different associations.**

Dataset	#ALV	#BAV	#ALB	#BT
Cellphone	700	948	735	675
Electronic	46	66	70	50

length constraint). In the experiments, we set  $\alpha$  to 0.01 (the same as [17]), and  $\beta$  to 4, which means the maximum number of edges between entities in a path is 4.

## 4.2 Results of Rule Selection

**Item linking to the Knowledge Graph.** In this step, we link the items from different domains to the entities in the knowledge graph. Items in the Cellphone domain and the Electronic domain are connected with 33,542 entities and 55,180 entities in Freebase respectively. Due to the item-entity linking method is not in a one-by-one accurate linking but based on items’ titles, brands, and descriptions, each item will be linked into several entities and each entity will be linked with several items. With the random walk strategy introduced in Section 3.2, we find that the four hop routes in the knowledge graph from these entities will pass over 10 million entities. To avoid introducing unrelated entities in random walk step, the type of entities are constrained on pre-defined entity types (e.g.: entities in “ns.base.brand”, “ns.computer” and some other types are maintain), then the involved entity amount is reduced to around 100K in each domain.

**Rule Learning.** The derived rules of different associations in cellphone domain are summarized in Table 2. There are hundreds of rules derived from Cellphone domain in each association, while only around 46-70 rules are in Electronic domain. The possible reason is that comparing with Cellphone domain, Electronic domain contains more items and the items are more diversity. Most rules are supported by less than 0.01 of the training item pairs. so less general rules are derived.

**Rule Selection.** To select useful rules from the large rule set, we use the learning based (LR, Eq (11)) and chi-square based feature selection methods in Section 3.2. The idea of selection methods is to choose the rules by which any items in a specific association are followed. E.g. if any item pairs in the BT association follows a rule  $R_k$ , then  $R_k$  is a useful rule for the BT association.

We choose the ALB association in the Cellphone dataset to verify the selection ability of the two methods. Because the derived rules will be used to extract item-item pair feature for the recommendation, a good rule should be able to indicate the associations between item  $i$  and user’s purchase history  $I_u$ . So the recommendation dataset (Section 5.1) in the Cellphone domain is used for evaluation, we calculate the recall of whether there is at least one path satisfied rule  $r_k$  between the last item  $i_l$  user interacted and user’s previous purchase history  $I_u$ . Due to not always exist at least one rule between  $i_l$  and  $I_u$ , there is a upper bound for the recall.

**Table 3: Rule selection results on ALB association in the Cellphone domain.**

LR		Chi-square		All	Upper Bound
Top 50	Top 100	Top 50	Top 100		
20.1%	40.1%	87.0%	88.5%	89.2%	90.7%

Table 3 shows the rule selection results ALB association in Cellphone domain and its upper bound. Chi-square based method outperforms linear-regression based method in rule selection. The reason is that rules with higher weight in linear regression model cannot fully represent usefulness of rules in the recommendation. However, Chi-square method is able to find the most useful rules, and the selected 50 rules cover 87.0% of user purchase history (only 2.2% percentage lower than using all rules). It is reasonable to choose only the subset of derived rules for the recommendation. Besides, we find that the upper bound in Electronic domain is only about 65%, indicating that the combination between rules in Electronic dataset is not as tightly as in the Cellphone dataset.

Besides, for multi-task learning framework, it is unnecessary to conduct rule selection because the model takes the effect of each rule in predicting item associations through the combined loss function Eq ((10), (11), or (12)).

## 5 RECOMMENDATION EXPERIMENTS

This section introduces dataset and experiment settings for comparing RuleRec with other baseline methods, as well as providing case study on analyzing different components of RuleRec.

### 5.1 Recommendation Dataset

The recommendation datasets are open datasets that extracted from Amazon [10, 21]. Each user’s purchase histories in Amazon are recorded with the purchased items and times. We conduct experiments using two datasets: Amazon Cellphone and Amazon Electronic. Each user has at least 5 interactions with items. The statistics of the datasets are summarized in Table 4.

**Table 4: The statistics of recommendation datasets.**

Dataset	#user	#item	#links
Cellphone	27, 879	10, 429	194, 439
Electronic	22, 675	58, 741	195, 751

### 5.2 Experimental Settings

**Evaluation Protocol.** To evaluate the item recommendation performance, we use leave-one-out evaluation in the recommendation [1, 14]. The latest interactions between items and each user are used as positive items in test set, and the remaining data are used for training. Due to the loss function in our study is pair-wised, each positive item in training set will be trained with a negative

item sampled from items that the user has not interacted. As for test set, since it is too time-consuming to rank all items for each user in evaluation, 99 negative items that are not interacted with the user are random sampled and added to test set [5, 36]. Therefore, in the test set, each user is evaluated with 99 negative items and one positive item. The target here is to generate a high-quality ranked list of items for each user.

**Evaluation Metrics.** We use Recall, Normalized Discounted Cumulative Gain (NDCG), and Mean reciprocal rank (MRR). Higher score means better performance in each metric. Recall focuses on whether the positive item is in the list, while NDCG and MRR take the position of the positive item into evaluation. Considering that the length of most recommendation list in real scenarios is 5 or 10, so the ranked list is truncated at 10 for all metrics. We calculate Recall@5, Recall@10, NDCG@10, and MRR@10 for evaluation.

### 5.3 Compared Methods

Three types of baselines (traditional matrix factorization, neural network based, and recommendation with knowledge graph) are used here:

- **BPRMF** [27]. As introduced in Section 2.2.1, this method follows the idea of matrix factorization with pairwise ranking loss.
- **NCF** [14]: This is a state-of-the-art latent factor model. It pre-trains MLP and GMF part separately, and then ensembles the two models to get the final preference score. Following previous studies [5, 13], **MLP** and **GMF** are taken as baseline models too.
- **HERec** [31]: A state-of-the-art algorithm which using the knowledge graph for the recommendation. This method adopts meta-paths to generate the embeddings of users and items in the heterogeneous network with Deepwalk [26], and then use them in the recommendation. Two variants of this algorithm with different fusion functions, **HERec<sub>sl</sub>** (with the simple linear fusion function) and **HERec<sub>pl</sub>** (with personalized linear fusion function) are used as baseline models.
- **RippleNet** [34]: Another state-of-the-art algorithm that incorporates the knowledge graph into recommender systems. It stimulates the propagation of user preferences on the set of knowledge entities to learn a user’s potential interests.

**Implementation Details.** We adopt the implementation of BPRMF algorithm in MyMediaLite<sup>8</sup> (a famous open source package) on our experiments. The implementation of other algorithms are from the public codes that the authors provided in their papers (NCF<sup>9</sup>, HERec<sup>10</sup>, and RippleNet<sup>11</sup>). The four new models, RuleRec<sub>two</sub> with BPRMF, RuleRec<sub>two</sub> with NCF, RuleRec<sub>multi</sub> with BPRMF, and RuleRec<sub>multi</sub> with NCF are modified from BPRMF and NCF according to our framework respectively. We tune all the parameters to achieve the best performance of each algorithm.

The score function is defined as  $S'_{u,i} = f_w(S_{u,i}, F_{(i, I_u|R)}) = S_{u,i} + \alpha \cdot w^T F_{(i, I_u|R)}$  in this section. Different implementations of  $f_w$  and their results will be analyzed in Section 5.5. All of the four types of item associations are used in the recommendation module for both two-step and multi-task learning algorithms. Top 50 rules of each type of item associations (selected with chi-square method) are chose to the two-step based methods. To make the comparison fair, these rules are used in the multi-task learning algorithms with the

sigmoid objective function in the final experiments. The objective function is sigmoid (Eq (12)), as it performs the best in the three objective functions (Eq (10), (11), and (12)); due to the limited of length, we do not show the results here. The comparison of different amounts of rules will be introduced in Section 5.5.5.

### 5.4 Experiments and Performance Study

The experimental results of these algorithms in different domains are summarized in Table 5. We repeated each setting for 5 times and conducted the paired two-sample t-test on the 5 times experiment results for significant test. As shown in the table, the performance of algorithms in Electronic dataset is obviously worse than in Cellphone dataset. The reason is that the item count of Electronic dataset is about 6 times over the item count of Cellphone dataset (from Table 4), which makes the recommendation in Electronic dataset more difficult.

**1. The Enhanced Algorithms vs. the Originals.** NCF algorithm performs better than BPRMF algorithm in both datasets, as more complex user and item feature interactions are taken into consideration in NCF. Looking into the results of BPRMF algorithms and NCF algorithms, we find that RuleRec<sub>multi</sub> with BPRMF gets 6.5% to 11.0% improvements over BPRMF in different evaluation metrics on two domains. The improvements of RuleRec<sub>multi</sub> with NCF in Recall@5, Recall@10, NDCG@10, and MRR@10 are between 3.0% to 6.4% comparing with NCF in Cellphone domain, while the improvements of which on Electronic is lower than in Cellphone domain. Though RuleRec<sub>multi</sub> with BPRMF is improved more than RuleRec<sub>multi</sub> with NCF, RuleRec<sub>multi</sub> with NCF still achieves the best performance in Cellphone domain and RuleRec<sub>multi</sub> with BPRMF performs the best in Electronic domain.

**2. Overall Performances.** Besides, we find that any one of the enhanced algorithms outperform all baselines in both Cellphone and Electronic domains in each metric. And most of the improvements are statistically significant, showing that the derived rules from the knowledge graph are really helpful to generate a better ranked item list for the recommendation. The multi-task learning algorithms (RuleRec<sub>multi</sub> with BPRMF and RuleRec<sub>multi</sub> with NCF) show better performances than the two-step learning algorithms (RuleRec<sub>two</sub> with BPRMF and RuleRec<sub>two</sub> with NCF), indicating that the combination of recommendation loss and rule selection loss in weight training is able to boost the recommendation results. Though the learning-based selection methods perform worse than chi-square in Section 4.2, it does helpful in the multi-task learning.

**3. The Performances of HERec and RippleNet.** We also note that HERec based algorithms and RippleNet, some state-of-the-art algorithms that uses the knowledge graph for the recommendation, performs worse in these datasets. We think the possible reason is that unlike movie, book, or Yelp datasets which contains many well organized category features (such as director, movie type, actor/actress name in movie dataset) to construct a compact graph, here we link Cellphone and Electronic datasets with a real knowledge graph Freebase. Though Freebase contains more information, but it is not as clean as the on-topic sub graph and makes it harder to mine valuable information, so these algorithms perform worse. More analyses are shown in Section 5.5.1.

To summarize, the derived rules from knowledge graph are valuable for item pair feature vector learning, and the learned vector is able to enhance multiple basic recommendation models (BPRMF and NCF here). Comparing with the two-step combination method,

<sup>8</sup><http://www.mymedialite.net/index.html>

<sup>9</sup>[https://github.com/hexiangnan/neural\\_collaborative\\_filtering](https://github.com/hexiangnan/neural_collaborative_filtering)

<sup>10</sup><https://github.com/librahu/HERec>

<sup>11</sup><https://github.com/hwwang55/RippleNet>

**Table 5: Performance Comparison between RuleRec and Other Methods in Different Domains.** RuleRec<sub>two</sub> and RuleRec<sub>multi</sub> are our proposed models. RuleRec<sub>two</sub> is a two-step rule-based model and RuleRec<sub>multi</sub> is a multi-task model. These models use BPRMF or NCF as a recommendation model. \* indicates statistical significance at  $p < 0.01$  compared to the best baseline model.

Methods / Dataset	Cellphone				Electronic			
	Recall@5	Recall@10	NDCG@10	MRR@10	Recall@5	Recall@10	NDCG@10	MRR@10
BPRMF [27]	0.3238	0.4491	0.2639	0.2058	0.1886	0.2763	0.1571	0.1207
GMF [13]	0.3379	0.4666	0.2789	0.2223	0.1988	0.2835	0.1657	0.1298
MLP [5]	0.3374	0.4779	0.2790	0.2182	0.2000	0.2883	0.1681	0.1315
NCF [14]	0.3388	0.4751	0.2761	0.2151	0.2005	0.2916	0.1679	0.1300
Hec <sub>sl</sub> [31]	0.2436	0.3481	0.2040	0.1600	0.1870	0.2851	0.1534	0.1135
Hec <sub>pl</sub> [31]	0.2511	0.3564	0.2090	0.1641	0.1948	0.2851	0.1628	0.1256
RippleNet [34]	0.2834	0.4042	0.2219	0.1780	0.1965	0.2865	0.1638	0.1265
RuleRec <sub>two</sub> (BPRMF)	0.3495*	0.4768	0.2813*	0.2201*	0.2050*	0.2932	0.1707*	0.1334*
RuleRec <sub>multi</sub> (BPRMF)	0.3568*	0.4829*	0.2864*	0.2246*	0.2071*	0.2946*	<b>0.1718*</b>	<b>0.1341*</b>
RuleRec <sub>two</sub> (NCF)	0.3538*	0.4876*	<b>0.2902*</b>	<b>0.2296*</b>	0.2049*	<b>0.2947*</b>	0.1681	0.1296
RuleRec <sub>multi</sub> (NCF)	<b>0.3569*</b>	<b>0.4894*</b>	<b>0.2902*</b>	0.2290*	<b>0.2074*</b>	0.2917	0.1702*	0.1330

**Table 6: Performance Comparison on MovieLens dataset.**

Model	AUC
Hec <sub>sl</sub> [31]	0.894
Hec <sub>pl</sub> [31]	0.895
RippleNet [34]	<b>0.921</b>
RuleRec <sub>two</sub> (BPRMF)	0.907

multi-task learning for both recommendation and rule selection contributes more on rule weight learning. Due to the flexible of the proposed framework, the derived rules are able to combine with other recommendation models to boost performances significantly.

## 5.5 Case Study and Performance Analysis

### 1. Performance Comparison in compact heterogeneous graph.

Experiments in Section 5.4 are conducted on a large heterogeneous graph extracted from real knowledge graph. In this subsection, some extra experiments are conducted on a compact heterogeneous graph, which is constructed based on item attributes, in MovieLens-1M dataset<sup>12</sup>. We adopt the proposed algorithm and HERec algorithm in this dataset following the setting in RippleNet.

The experimental results are shown in Table 6. Our model performs better than HERec while worse than RippleNet, there are two possible reason: 1) relation type is very limited in this dataset (only 7), so the power of rule selection for the recommendation in RuleRec is limited in this scenario. 2) MovieLens-1M is different from real knowledge graph datasets in Section 5.4 (which is constructed by linking items into Freebase), the connection coverage of it is very perfect and RippleNet benefits a lot from this. The results indicate that the proposed algorithms is able to achieve noteworthy performance in compact heterogeneous graph.

**2. Explainability of the learned rules.** In Section 5.4, the results indicate the derived rules are useful in providing more accurate recommendation results. In this section, we will show the explainability of the derived rules for the recommendation. Two positive weighted rules on RuleRec<sub>multi</sub> are shown as the following :

- $R_1 = \text{"computer.computer.manufacturer"}$
- $R_2 = \text{"computer.computer.compatible_oses" -> "computer.os_compatibility.operating_system" -> "computer.operating_system.includes_os_versions"}$

Where the words with quotation marks are the relation types defined in Freebase (such as "computer.computer.manufacturer"). These rules are with positive weights in the recommendation module, indicating that if a new item  $b$  exists a path between it and item  $a$  user bought before, item  $b$  is more likely to get higher score.

First we try to verify if item pairs with these rules affect user’s purchase. As to  $R_1$ , it links a computer product and its manufacturer. If two items  $a$  and  $b$  have a path in  $R_1$ , it means that item  $b$  is likely to be manufactured by the same as item  $a$ . For  $R_2$ , two example entity paths in this rule are: 1) "Mac Mini" - "os x yosemite" - "OS X" - "IOS" and 2) "Surface Pro" - "Windows 10" - "Windows" - "Windows Phone". It shows that users are tend to use similar operating systems in both cellphone and computer. As you can see, these rules are consistent with our common sense.

Then, to check whether users agree that the selected rule will be helpful to improve the explainability of the recommendation if the rules are used in real scenarios, the derived rules in Cellphone dataset are labeled by three experts (only agree or disagree, 100 rules from ALB and BT associations). The results show that over 94% learned rules are accepted by users (87% rules are accepted by all users).

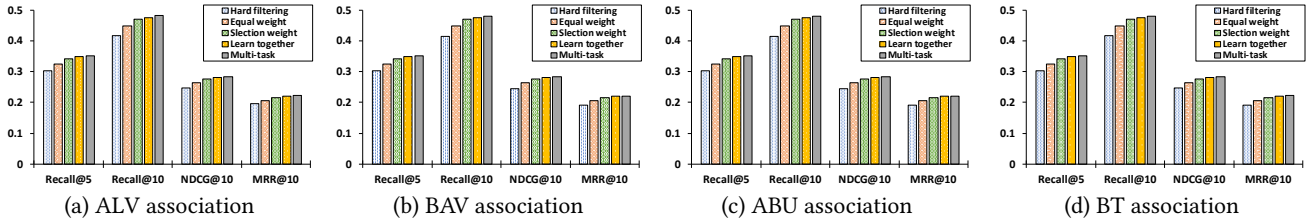
Due to the effective rule  $i$  in calculating user preference on a specific item will get higher score ( $w_i^T F_{(i, I_u | r_i)}$ ) for the preference prediction. So for each item in the ranked list, unless it has no path between it and items in user’s purchase history, we can generate the most important rule for it by ranking the score of each rule in preference prediction.

**3. Study on different model integration strategies.** The score function (Eq (14)) and the rule weight vector  $w$  in  $f_w$  affect the performance of the recommendation module. We experiment with several ways to identify the best combination methods.

- Hard filtering: Remove candidate items that have no rule with any item in  $I_u$ . Formally,  $S'_{u,i} = S_{u,i} \cdot I(F_{(i, I_u | R)})$ , where  $I(F_{(i, I_u | R)}) = 1$  if  $\sum F_{(i, I_u | R)} > 1$  otherwise 0.
- Equal weight: Each rule gets an equal weight in prediction.  $S'_{u,i} = S_{u,i} + \alpha \cdot w^T F_{(i, I_u | R)}$ , and  $w = [0.02, \dots, 0.02]$ .
- Selection weight:  $S'_{u,i} = S_{u,i} + \alpha \cdot w^T F_{(i, I_u | R)}$ , and  $w = w_{ruleselection}^{ruleselection}$  is the rule weight vector trained by Eq (12) in rule selection step.
- Learn together: The rule weight vector is trained with the original recommendation model.  $f_w(a, b) = a + w^T b$ .

<sup>12</sup><https://github.com/hwwang55/RippleNet/tree/master/data/movie>





**Figure 6: Performance comparison under different score function settings in the Cellphone dataset.** We select the top 50 rules in each association by chi-square method. Among all score function settings, the multi-task learning method performs the best.

**Table 7: The results of using a single association vs. all associations.** BPRMF is used as a recommendation model. Our proposed model shows the best performances when using all kinds of associations.

Type	Recall@5	Recall@10	NDCG@10	MRR@10
None	0.3238	0.4491	0.2639	0.2058
ALV	0.3527	0.4815	0.2844	0.2225
BAV	0.3513	0.4812	0.2840	0.2222
ABU	0.3511	0.4811	0.2838	0.2218
BT	0.3514	0.4810	0.2841	0.2222
<b>ALL</b>	<b>0.3568</b>	<b>0.4829</b>	<b>0.2864</b>	<b>0.2246</b>

- Multi-task: The rule weight vector  $w$  is shared by recommendation learning and rule selection part, and the score prediction function is  $f_w(a, b) = a + w^T b$ .

The results of applying multiple score functions on RuleRec(BPRMF) model in Cellphone dataset are shown in Fig. 6. The performances of the score functions are Hard filtering < Equal weight < Selection weight < Learn together < Multi-task with all metrics in all associations. First, we can see that the multi-learning method achieves the best performance on all metrics, and the improvements are significant, which indicates multi-task learning is very helpful in rule weight learning for better recommendation results. Second, since the hard filtering method is likely to ignore both negative items and positive items (from Table 3, we can see that sometimes there is no rule between the positive item and item purchase history  $I_u$ ). Third, though selection weight contributes on the recommendation (better than equal weight), it is still worse than Learn together model.

#### 4. Study on single association vs. all associations

In this subsection, we compare the performance of RuleRec<sub>multi</sub> with BPRMF with only one type of association and all associations, the results are summarized in Table 7.

First, we can see that with the rules derived by any one of the four associations, RuleRec<sub>multi</sub> with BPRMF outperforms BPRMF algorithm significantly. The performances of using different associations are similar, but all of them are valuable for mining the item relationships to boost the recommendation results. Second, RuleRec<sub>multi</sub> with BPRMF derived by all kinds of associations outperforms RuleRec<sub>multi</sub> with a single association, indicating that the combination contributes for the recommendation models.

**5. Recommendation with different rule counts.** In Section 3.2, rule selection is introduced as an important part in rule learning. Does rule selection is really necessary? We conduct further experiments on each association with different of derived of rules (selected with chi-square method, 50, 100, 200, 300, 400, and 500 respectively) with RuleRec<sub>two</sub> and RuleRec<sub>multi</sub> with BPRMF.

From Fig. 7, the performance of RuleRec<sub>two</sub> with BPRMF decreases in recall@5, MRR@10 as the rule number increases. At the same time, NDCG@10 keeps stable and Recall@10 increases. The overall performances is not getting better when more rules are applied in the recommendation learning. The possible reason is that with the grows of rule number, lots of "bad" rules are included and the two-step model RuleRec<sub>two</sub> with BPRMF shows worse ability in dealing with them properly. However, due to the rule selection is taken into consideration in the multi-task learning based algorithms, we find that the performance of RuleRec<sub>multi</sub> with BPRMF algorithm shows better performances as the rule number increases (Fig. 8). Furthermore, the performances of RuleRec<sub>multi</sub> with BPRMF is significantly better than those of RuleRec<sub>two</sub> with BPRMF (paired two-sample t-test on the experimental results with different count of rules.  $p < 0.01$ ). The results show that the multi-task learning based algorithms are able to tackle with large number of rules even though there are useless rules.

## 6 RELATED WORK

**Combine Side-information for the Recommendation.** Matrix factorization based algorithms [27, 28] are widely used to tackle recommendation problems. Recently, recommendation algorithms achieve remarkable improvements during these years with help of deep learning models [12, 14, 30, 47, 49] and the successful introducing of side-information [15, 19, 22, 23, 34]. In this study, we focus on the introducing of side-information in the knowledge graph for the recommendation, and there already two types of studies using the knowledge graph in the recommendation: path-based and embedding learning based.

Path-based methods adopt random walk on predefined meta-paths between user and items in the knowledge graph to calculate user's preference on an item. Yu et al. first propose to use meta-paths to utilize user-item preferences and then expand matrix factorization for the recommendation [43]. Shi et al. use weighted paths for explicit recommendation [32]. Zhao et al. design a factorization machine with the latent features from different meta-paths [48]. Catherine et al. design a first-order probabilistic logical reasoning system, named ProPPR, to integrate different meta-paths in a knowledge graph [3, 4]. All these methods achieve improvements in the recommendation, while the weakness of them is that they ignore the type of item associations.

Embedding learning based methods conduct user/item representation learning based on the Knowledge graph structure firstly. The learned embedding [8] is applied in Zhang's study to get item embedding for the recommendation [24]. Zhang et al. use TransR [18] to learn the structural vectors of items, and these vectors are part of the final item latent vector for preference prediction [44]. Besides,

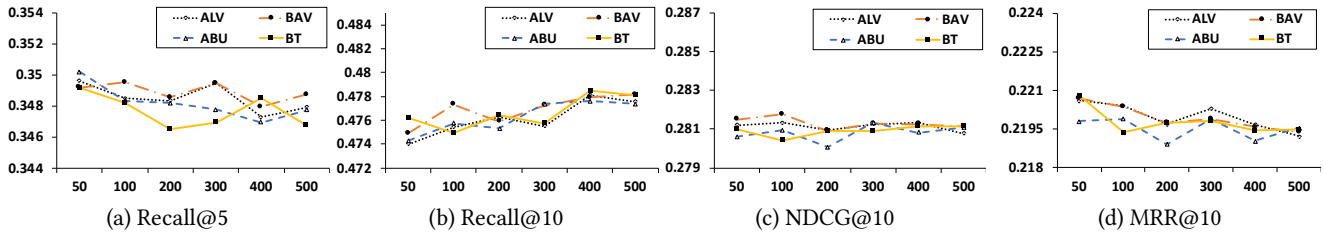


Figure 7: The performance of using different number of rules in RuleRec<sub>two</sub> with BPRMF.

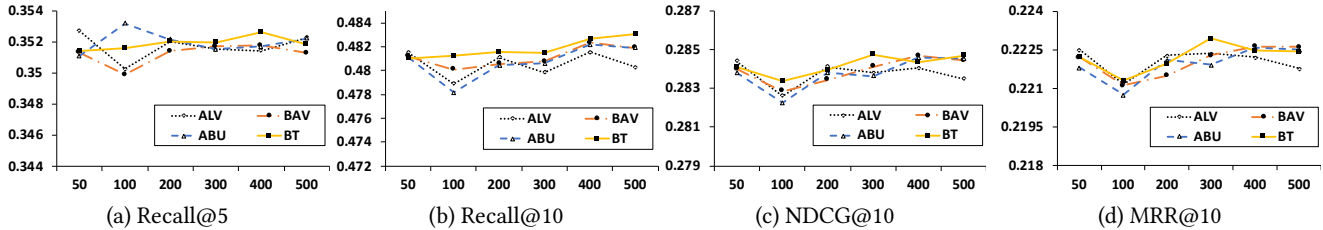


Figure 8: The performance of using different number of rules in RuleRec<sub>multi</sub> with BPRMF.

some previous studies propose new algorithms in which Meta-path guided random walks are used in a heterogeneous network for user and item embedding learning and achieve outperform results [31, 34, 37, 48] in different ways. However, the embedding learning based methods give up the explainable strength of the knowledge graph, which is very valuable for the recommendation.

**Rule Learning in the Knowledge Graph.** Item-item relationships are considering as useful features for providing better recommendation results. Julian et al. firstly propose to a topic model based method to predict relationships (substitute or complementary) between products from reviews [20]. In their study, the ground truth is calculated in a data-driven way in Amazon. Then, more algorithms attempt to improve the prediction results with better algorithms. Word dependency paths are taken into consideration in Hu et al.’s work [41] and Wang et al. adopt a embedding based method to enhance the performance of relationship prediction [37].

However, these methods are suffering from cold items. So we proposed to not predict item associations directly, but mine meaningful rules in the knowledge graph with the ground truth item pairs. The rules will be applied to generate feature vectors for different item pairs without user reviews.

Knowledge graph is a multi-relational graph that composed of entities as nodes and relations as different types of edges [38]. In the past years, knowledge graphs have been used as important resources for many tasks [7, 39]. One of the main usage of knowledge graph is reasoning and entity relationship prediction. Lots of research are focus on reasoning, such as [40, 42]. While these studies focus on link prediction but not rule inducing, which is not proper for our study.

On the other line of research, several work attempt to learn the useful rules but not the prediction results from the knowledge graph with the ground truth entity pairs. Random walk based algorithms are proposed in Lao et al.’s studies [16, 17] and others’ [9, 35]. These methods are able to show why the entity pair has a certain relationship according to the derived rules, which makes the results more explainable. In our study, we adopt a similar algorithm as them in rule learning module.

## 7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel and effective joint optimization framework for inducing rules from a knowledge graph with items and recommendation based on the induced rules.

Our framework consists of two modules: rule learning module and recommendation module. The rule learning module is able to derive useful rules in a knowledge graph with different type of item associations, and the recommendation module introduces the rules to the recommendation models for better performance. Furthermore, there are two ways to implement this framework: two-step and jointly learning.

Freebase, a large-scale knowledge graph, is used for rule learning in this study. The framework is flexible to boost different recommendation algorithms. We modify two recommendation algorithms, a classical matrix factorization algorithm (BPRMF) and a state-of-the-art neural network based recommendation algorithm (NCF), to combine with our framework. The proposed four rule enhanced recommendation algorithms achieve remarkable results in multiple domains and outperform all baseline models, indicating the effectiveness of our framework. Besides, the derived rules also show the ability in explaining why we recommend this item for the user, boosting the explainability of the recommendation models at the same time. Further analysis shows that our multi-task learning based combination methods (RuleRec<sub>multi</sub> with BPRMF and RuleRec<sub>two</sub> with NCF) outperform the two-step method with different number of rules. And the combination of rules derived by different associations contributes to better recommendation results.

In future, we plan to investigate how to design a embedding learning based combination algorithm which keeps the recommendation results explainable with the knowledge graph.

## ACKNOWLEDGEMENTS

This work is supported by Natural Science Foundation of China (Grant No. 61672311, 61532011) and The National Key Research and Development Program of China (2018YFC0831900). Dr. Xiang Ren has been supported in part by NSF SMA 18-29268, Amazon Faculty Award, and JP Morgan AI Research Award.

## REFERENCES

- [1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1341–1350.
- [2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 1247–1250.
- [3] Rose Catherine and William Cohen. 2016. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 325–332.
- [4] Rose Catherine, Kathryn Mazaitis, Maxine Eskenazi, and William Cohen. 2017. Explainable entity-based recommendations with knowledge graphs. *arXiv preprint arXiv:1707.05254* (2017).
- [5] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation.. In *IJCAI* 3329–3335.
- [6] 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 (I-Semantics)*.
- [7] Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs. *arXiv preprint arXiv:1801.03825* (2018).
- [8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
- [9] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 192–202.
- [10] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 507–517.
- [11] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1661–1670.
- [12] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.
- [13] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. *arXiv preprint arXiv:1808.03912* (2018).
- [14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiangrendle2009bprng Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.
- [15] Liang Hu, Songlei Jian, Longbing Cao, and Qingkui Chen. 2018. Interpretable Recommendation via Attraction Modeling: Learning Multilevel Attractiveness over Multimodal Movie Contents.. In *IJCAI* 3400–3406.
- [16] Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning* 81, 1 (2010), 53–67.
- [17] Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 529–539.
- [18] 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. 2181–2187.
- [19] Weizhi Ma, Min Zhang, Chenyang Wang, Cheng Luo, Yiqun Liu, and Shaoping Ma. 2018. Your Tweets Reveal What You Like: Introducing Cross-media Content Information into Multi-domain Recommendation. In *IJCAI* 3484–3490.
- [20] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.
- [21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.
- [22] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 31–39.
- [23] Sharad Nandanwar, Aayush Moroney, and M Narasimha Murty. 2018. Fusing Diversity in Recommendations in Heterogeneous Information Networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 414–422.
- [24] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 32–36.
- [25] Haekyu Park, Hyunsik Jeon, Junghwan Kim, Beunguk Ahn, and U Kang. 2017. Uniwalk: Explainable and accurate recommendation for rating and network data. *arXiv preprint arXiv:1710.07134* (2017).
- [26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*.
- [27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
- [28] Jasson DM Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 713–719.
- [29] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press.
- [30] Yilin Shen, Yue Deng, Avik Ray, and Hongxia Jin. 2018. Interactive recommendation via deep neural memory augmented contextual bandits. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 122–130.
- [31] Chuan Shi, Binbin Hu, Xin Zhao, and Philip Yu. 2018. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).
- [32] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. 2015. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 453–462.
- [33] Beidou Wang, Martin Ester, Jiajun Bu, and Deng Cai. 2014. Who also likes it? generating the most persuasive social explanations in recommender systems. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- [34] 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, 417–426.
- [35] Quan Wang, Bin Wang, Li Guo, et al. 2015. Knowledge Base Completion Using Embeddings and Rules.. In *IJCAI* 1859–1866.
- [36] Zengmao Wang, Yuhong Guo, and Bo Du. 2018. Matrix completion with Preference Ranking for Top-N Recommendation.. In *IJCAI* 3585–3591.
- [37] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 619–627.
- [38] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*, Vol. 14. 1112–1119.
- [39] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1271–1279.
- [40] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690* (2017).
- [41] Hu Xu, Sihong Xie, Lei Shu, and S Yu Philip. 2016. Cer: Complementary entity recognition via knowledge expansion on large unlabeled product reviews. In *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 793–802.
- [42] Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*. 2319–2328.
- [43] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 347–350.
- [44] 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, 353–362.
- [45] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *arXiv preprint arXiv:1804.11192* (2018).
- [46] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 83–92.
- [47] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 717–726.
- [48] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 635–644.
- [49] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. *arXiv preprint arXiv:1805.02343* (2018).