Ripple Network: Propagating User Preferences on the Knowledge Graph for Recommender Systems

Hongwei Wang[1,2], Fuzheng Zhang[2], Jialin Wang[3], Miao Zhao[3], Wenjie Li[3], Xing Xie[2], Minyi Guo[1*]
[1]Shanghai Jiao Tong University, wanghongwei55@gmail.com, guo-my@cs.sjtu.edu.cn
[2]Microsoft Research Asia, {fuzzhang, xingx}@microsoft.com
[3]The Hong Kong Polytechnic University, {csjlwang, csmiaozhao, cswjli}@comp.polyu.edu.hk

ABSTRACT

To address the sparsity and cold start problem of collaborative filtering, researchers usually make use of side information, such as social networks or item attributes, to improve recommendation performance. This paper considers the knowledge graph as the source of side information. To address the limitations of existing embedding-based and path-based methods for knowledge-graph-aware recommendation, we propose *Ripple Network*, an end-to-end framework that naturally incorporates the knowledge graph into recommender systems. Similar to actual ripples propagating on the surface of water, Ripple Network stimulates the propagation of user preferences over the set of knowledge entities by automatically and iteratively extending a user's potential interests along links in the knowledge graph. The multiple "ripples" activated by a user's historically clicked items are thus superposed to form the preference distribution of the user with respect to a candidate item, which could be used for predicting the final clicking probability. Through extensive experiments on real-world datasets, we demonstrate that Ripple Network achieves substantial gains in a variety of scenarios, including movie, book and news recommendation, over several state-of-the-art baselines.

KEYWORDS

Recommender systems; knowledge graph; preference propagation

1 INTRODUCTION

The explosive growth of online content and services has provided overwhelming choices for users, such as news, movies and books. Recommender Systems (RS) intend to address the information explosion by finding a small set of items for users to meet their personalized interests. Among recommendation strategies, *Collaborative Filtering* (CF), which considers users' historical interactions and makes recommendations based on their potential common preferences, has achieved great success [11]. However, CF-based methods usually suffer from the sparsity of user-item interactions and the cold start problem. To address these limitations, researchers have proposed incorporating *side information* into CF, such as social networks [9], user attributes [27], images [35] and contexts [23].

Among various types of side information, *knowledge graph* (KG) usually contains much more fruitful facts and connections about items. A KG is a type of directed heterogeneous graph in which nodes correspond to *entities* and edges correspond to *relations*. Recently, researchers have proposed several academic and commercial KGs, such as NELL[1], DBpedia[2], Google Knowledge Graph[3] and Microsoft Satori[4]. These knowledge graphs are successfully applied in many applications such as KG completion [13], question answering [7], word embedding [33], and text classification [29].

Inspired by the success of applying KG in a wide variety of tasks, researchers also tried to utilize KG to improve the performance of recommender systems. Existing KG-aware recommendation can be classified into two categories:

(1) The first category is *embedding-based methods* [27, 28, 35], which pre-process a KG with *Knowledge Graph Embedding* (KGE) [30] algorithms and incorporates the learned entity embeddings into a recommendation framework. For example, Deep Knowledge-aware Network (DKN) [28] designs a CNN framework to combine entity embeddings with word embeddings for news recommendation. Collaborative Knowledge base Embedding (CKE) [35] combines CF with knowledge embedding, text embedding, and image embedding in a unified framework. Signed Heterogeneous Information Network Embedding (SHINE) [27] designs deep autoencoders to embed sentiment networks, social networks and profile (knowledge) networks for celebrity recommendations. Embedding-based methods show high flexibility in utilizing KG to assist recommender systems, but the adopted KGE algorithms in these methods are usually more suitable for in-graph applications such as link prediction than for recommendation [30], thus the learned entity embeddings are less intuitive and effective to characterize inter-item relations.

(2) The second category is *path-based methods* [34, 36], which explore the various patterns of connections among items in KG to provide additional guidance for recommendations. For example, Personalized Entity Recommendation (PER) [34] and Meta-Graph Based Recommendation [36] treat KG as a heterogeneous information network (HIN), and extract meta-path/meta-graph based latent features to represent the connectivity between users and items along different types of relation paths/graphs. Path-based methods make use of KG in a more natural and intuitive way, but they rely heavily on manually designed meta-paths, which is hard to optimize in practice. Another concern is that it is impossible

*M. Guo is the corresponding author. This work was partially sponsored by the National Basic Research 973 Program of China under Grant 2015CB352403.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1]http://rtw.ml.cmu.edu/rtw/
[2]http://wiki.dbpedia.org/
[3]https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html
[4]https://searchengineland.com/library/bing/bing-satori

to design hand-crafted meta-paths in certain scenarios (e.g., news recommendation) where entities and relations are not within one domain.

To address the limitations of existing methods, we propose *Ripple Network*, an end-to-end framework for knowledge-graph-aware recommendation. Ripple Network is designed for click-through rate (CTR) prediction, which takes a user-item pair as input and outputs the probability of the user clicking the item. The key idea behind Ripple Network is *preference propagation*: For each user, Ripple Network treats his historical interests as a seed set in the KG, then extends the user's interests iteratively along KG links to discover his hierarchical potential interests with respect to a candidate item. We analogize preference propagation with actual ripples created by raindrops propagating on the surface of water, in which multiple "ripples" superpose to form a resultant preference distribution of the user over the knowledge graph. The major difference between Ripple Network and existing literature is that Ripple Network combines the advantages of the above mentioned two types of methods: (1) Ripple Network incorporates the KGE methods into recommendation naturally by preference propagation; (2) Ripple Network can automatically discover possible paths from an item in a user's history to a candidate item, without any sort of hand-crafted design.

Empirically, we apply Ripple Network to three real-world scenarios of movie, book, and news recommendations. The experiment results show that Ripple Network achieves AUC gains of 1.1% to 39.4%, 3.2% to 35.3%, and 2.2% to 26.7% in movie, book, and news recommendations, respectively, compared with state-of-the-art baselines for recommendation. We also find that Ripple Network provides a new perspective of explainability for the recommended results in terms of the knowledge graph.

In summary, our contributions in this paper are as follows:

- To the best of our knowledge, this is the first work to combine embedding-based and path-based methods in KG-aware recommendation.
- We propose Ripple Network, an end-to-end framework utilizing KG to assist recommender systems. Ripple Network automatically discovers users' hierarchical potential interests by iteratively propagating users' preferences in the KG.
- We conduct experiments on three real-world recommendation scenarios, and the results prove the efficacy of Ripple Network over several state-of-the-art baselines.

2 PROBLEM FORMULATION

The knowledge-graph-aware recommendation problem is formulated as follows in this paper. Let $\mathcal{U} = \{u_1, u_2, ...\}$ and $\mathcal{V} = \{v_1, v_2, ...\}$ denote the sets of users and items, respectively. The user-item interaction matrix $\mathbf{Y} = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\}$ is defined according to users' implicit feedback, where

$$y_{uv} = \begin{cases} 1, & \text{if interaction } (u, v) \text{ is observed;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

A value of 1 for y_{uv} indicates there is an implicit interaction between user u and item v, such as click, watch, browse, etc.

In addition to the interaction matrix $\mathbf{Y}$, we also have a knowledge graph $\mathcal{G}$ available, which consists of massive entity-relation-entity triples (h, r, t). Here $h \in \mathcal{E}$, $r \in \mathcal{R}$, and $t \in \mathcal{E}$ denote the head, relation, and tail of a knowledge triple, respectively, $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities and relations in the KG. For example, the triple (*Jurassic Park*, *Is_Directed_By*, *Steven Spielberg*) states the fact that Steven Spielberg is the director of the film "Jurassic Park". In many recommendation scenarios, an item $v \in \mathcal{V}$ may associate with one or more entities in $\mathcal{G}$. For example, the movie "Jurassic Park" is linked with its namesake in KG, while news "France's baby panda makes public debut" is linked with entities "France" and "panda". Given interaction matrix $\mathbf{Y}$ as well as 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 $\hat{y}_{uv} = \mathcal{F}(u, v; \Theta)$, where $\hat{y}_{uv}$ denotes the probability that user u will click item v, and Θ denotes the model parameters of function $\mathcal{F}$.

3 RIPPLE NETWORK

In this section, we discuss the Ripple Network for recommender systems in detail. We also give some discussions on the proposed model and introduce related work.

3.1 Framework

The framework of Ripple Network is illustrated in Figure 1. Ripple Network takes a user u and an item v as input, and outputs the predicted probability that user u will click item v. For the input user u, his historical set of interests $\mathcal{V}_u$ is treated as seeds in the KG, then extended along links to form multiple ripple sets $\mathcal{S}_u^k$ $(k = 1, 2, ..., H)$. A ripple set $\mathcal{S}_u^k$ is the set of knowledge triples that are k-hop(s) away from the seed set $\mathcal{V}_u$. These ripple sets are used to interact with the item embedding (the yellow block) iteratively for obtaining the responses of user u with respect to item v (the green blocks), which are then combined to form the final user embedding (the grey block). Lastly, we use the embeddings of user u and item v together to compute the predicted probability $\hat{y}_{uv}$.

3.2 Ripple Set

A knowledge graph usually contains fruitful facts and connections among entities. For example, as illustrated in Figure 2, the film "Forrest Gump" is linked with "Robert Zemeckis" (director), "Tom Hanks" (star), "U.S." (country) and "Drama" (genre), while "Tom Hanks" is further linked with films "The Terminal" and "Cast Away" which he starred in. These complicated connections in KG provide us a deep and latent perspective to explore user preferences. For example, if a user has ever watched "Forrest Gump", he may possibly become a fan of Tom Hanks and be interested in "The Terminal" or "Cast Away". To characterize users' hierarchically extended preferences in terms of KG, in Ripple Network, we recursively define the set of k-hop relevant entities for user u as follows:

Definition 1 (relevant entity). *Given interaction matrix* $\mathbf{Y}$ *and knowledge graph* $\mathcal{G}$, *the set of* k*-hop relevant entities for user* u *is*

$$\mathcal{E}_u^k = \{t \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, ..., H, \tag{2}$$

where $\mathcal{E}_u^0 = \mathcal{V}_u = \{v \mid y_{uv} = 1\}$ *is the set of user's clicked items in the past, which can be seen as the seed set of user* u *in KG.*

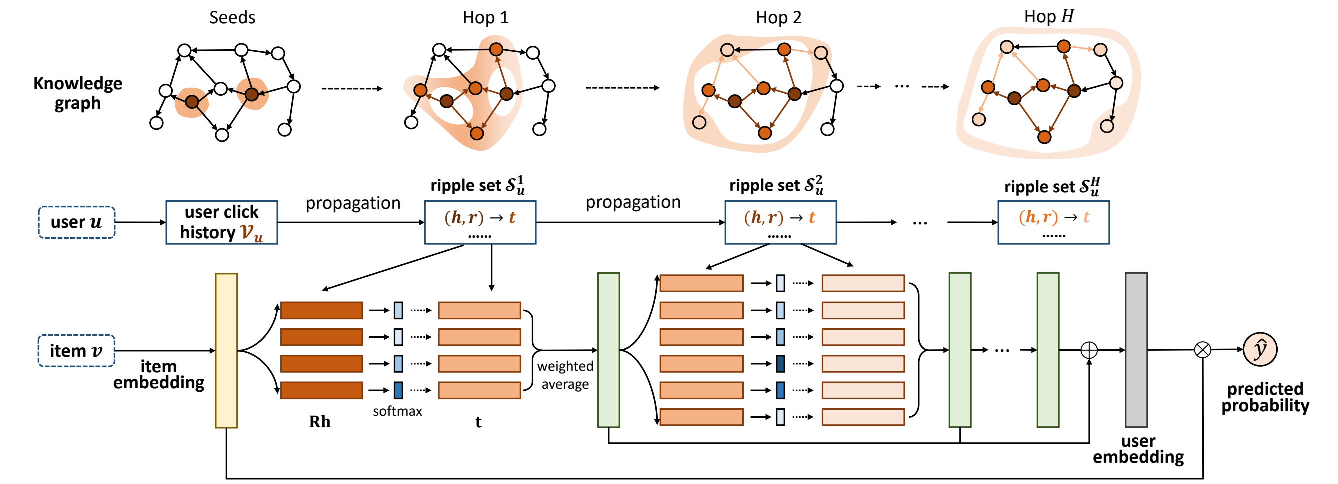


Figure 1: The overall framework of the Ripple Network. It takes one user and one item as input, and outputs the predicted probability that the user will click the item. The KGs in the upper part illustrate the corresponding ripple sets activated by the user's click history.

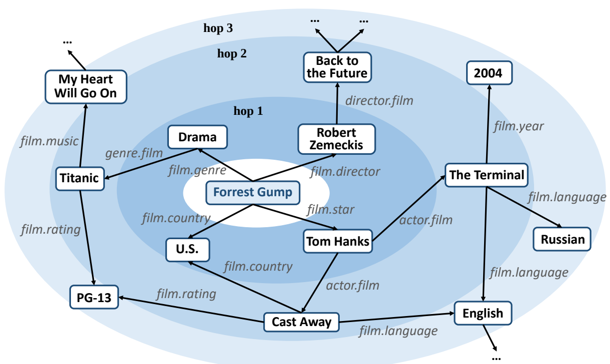


Figure 2: Illustration of ripple sets of "Forrest Gump" in KG of movies. The concentric circles denotes the ripple sets with different hops. The fading blue indicates decreasing relatedness between the center and surrounding entities.Note that the ripple sets of different hops are not necessarily disjoint in practice.

Relevant entities are natural extensions of a user's historical interests with respect to the KG. Given the definition of relevant entities, we then define the k-hop ripple set of user u as follows:

Definition 2 (ripple set). *The k-hop ripple set of user u is defined as the set of knowledge triples starting from $\mathcal{E}_u^{k-1}$:*

$$\mathcal{S}_u^k = \{(h, r, t) \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, ..., H. \quad (3)$$

The word "ripple" has two meanings: (1) Analogous to real ripples created by multiple raindrops, a user's potential interest in entities is activated by his historical preferences, then propagates along the links in KG layer by layer, from near to distant. We visualize the analogy by the concentric circles illustrated in Figure 2. (2) The strength of a user's potential preferences in ripple sets weakens with the increase of the hop number k, which is similar to the gradually attenuated amplitude of real ripples. The fading blue in

衰减 振幅

Figure 2 shows the decreasing relatedness between the center and surrounding entities.

One concern about ripple sets is their sizes may get too large with the increase of hop number k. To address the concern, note that: (1) A large number of entities in a real KG are *sink entities*, meaning they only have incoming links but no outgoing links, such as "2004" and "PG-13" in Figure 2. (2) In specific recommendation scenarios such as movie or book recommendations, relations can be limited to scenario-related categories to reduce the size of ripple sets and improve relevance among entities. For example, in Figure 2, all relations are movie-related and contain the word "film" in their names. (3) The number of maximal hop H is usually not too large in practice, since entities that are too distant from a user's history may bring more noise than positive signals. We will discuss the choice of H in the experiments part.

3.3 Preference Propagation

Traditional CF-based methods and their variants [10, 26] learn latent representations of users and items, then predict unknown ratings by directly applying a specific function to their representations such as inner product. In Ripple Network, to model the interactions between users and items in a more fine-grained way, we propose a preference propagation technique to explore users' potential interests in his ripple sets.

As shown in Figure 1, each item v is associated with an item embedding $\mathbf{v} \in \mathbb{R}^d$, where d is the dimension of embeddings. Item embedding can incorporate one-hot ID [10], attributes [27], bag-of-words [28] or context information [23] of an item, based on the application scenario. Given the item embedding $\mathbf{v}$ and the 1-hop ripple set $\mathcal{S}_u^1$ of user u, each triple (h_i, r_i, t_i) in $\mathcal{S}_u^1$ is assigned a relevance probability by comparing item v to the head h_i and the relation r_i in this triple:

$$p_i = \text{softmax}\left(\mathbf{v}^{\mathrm{T}}\mathbf{R}_i\mathbf{h}_i\right) = \frac{\exp\left(\mathbf{v}^{\mathrm{T}}\mathbf{R}_i\mathbf{h}_i\right)}{\sum_{(h,r,t)\in\mathcal{S}_u^1}\exp\left(\mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{h}\right)}, \quad (4)$$

where $\mathbf{R}_i \in \mathbb{R}^{d \times d}$ and $\mathbf{h}_i \in \mathbb{R}^d$ are the embeddings of relation r_i and head h_i, respectively. The relevance probability p_i can be regarded as the similarity of item v and the entity $\mathbf{h}_i$ measured in the space of relation $\mathbf{R}_i$. Note that it is necessary to take the embedding matrix $\mathbf{R}_i$ into consideration when calculating the relevance of item $\mathbf{v}$ and entity $\mathbf{h}_i$, since an item-entity pair may have different similarities when measured by different relations. For example, "Forrest Gump" and "Cast Away" are highly similar when considering their directors or stars, but have less in common if measured by genre or writer.

After obtaining the relevance probabilities, we take the sum of tails in $\mathcal{S}_u^1$ weighted by the corresponding relevance probabilities, and the vector $\mathbf{o}_u^1$ is returned:

$$\mathbf{o}_u^1 = \sum\nolimits_{(h_i, r_i, t_i) \in \mathcal{S}_u^1} p_i \mathbf{t}_i, \tag{5}$$

where $\mathbf{t}_i \in \mathbb{R}^d$ is the embedding of tail t_i. Vector $\mathbf{o}_u^1$ can be seen as the 1-order response of user u's click history $\mathcal{V}_u$ with respect to item v. This is similar to item-based CF methods [10, 28], in which a user is represented by his related items rather than a independent feature vector to reduce the size of parameters. Through the operations in Eq. (4) and Eq. (5), a user's interests are transferred from his history set $\mathcal{V}_u$ to the set of his 1-hop relevant entities $\mathcal{E}_u^1$ along the links in $\mathcal{S}_u^1$, which is called *preference propagation* in Ripple Network.

Note that by replacing $\mathbf{v}$ with $\mathbf{o}_u^1$ in Eq. (4), we can repeat the procedure of preference propagation to obtain user u's 2-order response $\mathbf{o}_u^2$, and the procedure can be performed iteratively on user u's ripple sets $\mathcal{S}_u^i$ for $i = 1, ..., H$. Therefore, a user's preference is propagated up to H hops away from his click history, and we observe multiple responses of user u with different orders: $\mathbf{o}_u^1, \mathbf{o}_u^2, ..., \mathbf{o}_u^H$. The embedding of user u with respect to item v is calculated by combining the responses of all orders:

$$\mathbf{u} = \alpha_1 \mathbf{o}_u^1 + \alpha_2 \mathbf{o}_u^2 + ... + \alpha_H \mathbf{o}_u^H, \tag{6}$$

where α_i's are positive trainable mixing parameters for measuring the importance of i-order response satisfying $\sum_{i=1}^H \alpha_i = 1$. Note that though the user response of last hop $\mathbf{o}_u^H$ contains all the information from previous hops theoretically, it is still necessary to incorporate $\mathbf{o}_u^k$ of small hops k in calculating user embedding since they may be diluted in $\mathbf{o}_u^H$. Finally, the user embedding and item embedding are combined to output the predicted clicking probability:

$$\hat{y}_{uv} = \sigma(\mathbf{u}^{\mathrm{T}} \mathbf{v}), \tag{7}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function.

3.4 Learning Algorithm

In Ripple Network, we intend to maximize the following posterior probability of model parameters Θ after observing the knowledge graph $\mathcal{G}$ and the matrix of implicit feedback Y:

$$\max p(\Theta | \mathcal{G}, \mathrm{Y}), \tag{8}$$

where Θ includes the embeddings of all entities, relations and items. This is equivalent to maximizing

$$p(\Theta | \mathcal{G}, \mathrm{Y}) = \frac{p(\Theta, \mathcal{G}, \mathrm{Y})}{p(\mathcal{G}, \mathrm{Y})} \propto p(\Theta) \cdot p(\mathcal{G} | \Theta) \cdot p(\mathrm{Y} | \Theta, \mathcal{G}) \tag{9}$$

according to Bayes' theorem. In Eq. (9), the first term $p(\Theta)$ measures the priori probability of model parameters Θ. Following [35], we set $p(\Theta)$ as Gaussian distribution with zero mean and a diagonal covariance matrix:

$$p(\Theta) = \mathcal{N}(\mathbf{0},\ \lambda_1^{-1} \mathbf{I}). \tag{10}$$

The second item in Eq. (9) is the likelihood function of the observed knowledge graph $\mathcal{G}$ given Θ. Recently, researchers have proposed a great many knowledge graph embedding methods, including translational distance models [3, 13] and semantic matching models [14, 18] (We will continue the discussion on KGE methods in Section 3.6.3). In Ripple Network, similar to [19], we use a three-way tensor factorization method to define the likelihood function for KGE:

$$\begin{aligned} p(\mathcal{G} | \Theta) &= \prod\nolimits_{(h,r,t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}} p\big((h,r,t) | \Theta\big) \\ &= \prod\nolimits_{(h,r,t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}} \mathcal{N}\big(I_{h,r,t} - \mathbf{h}^{\mathrm{T}} \mathbf{R} \mathbf{t},\ \lambda_2^{-1}\big), \end{aligned} \tag{11}$$

where the indicator $I_{h,r,t}$ equals 1 if $(h, r, t) \in \mathcal{G}$ and is 0 otherwise. Based on the definition in Eq. (11), the scoring functions of entity-entity pairs in KGE and item-entity pairs in preference propagation can thus be unified under the same calculation model. The last term in Eq. (9) is the likelihood function of the observed implicit feedback given Θ and the KG, which is defined as the product of Bernouli distributions:

$$p(\mathrm{Y} | \Theta, \mathcal{G}) = \prod\nolimits_{(u,v) \in \mathrm{Y}} \sigma(\mathbf{u}^{\mathrm{T}} \mathbf{v})^{y_{uv}} \cdot \big(1 - \sigma(\mathbf{u}^{\mathrm{T}} \mathbf{v})\big)^{1 - y_{uv}} \tag{12}$$

based on Eq. (2)–(7).

Taking the negative logarithm of Eq. (9), we have the following loss function for Ripple Network:

$$\begin{aligned} \min \mathcal{L} &= -\log \big(p(\mathrm{Y}|\Theta, \mathcal{G}) \cdot p(\mathcal{G}|\Theta) \cdot p(\Theta)\big) \\ &= \sum_{(u,v) \in \mathrm{Y}} -\Big(y_{uv} \log \sigma(\mathbf{u}^{\mathrm{T}} \mathbf{v}) + (1 - y_{uv}) \log \big(1 - \sigma(\mathbf{u}^{\mathrm{T}} \mathbf{v})\big)\Big) \\ &\quad + \frac{\lambda_2}{2} \sum_{r \in \mathcal{R}} \|\mathbf{I}_r - \mathbf{E}^{\mathrm{T}} \mathbf{R} \mathbf{E}\|_2^2 + \frac{\lambda_1}{2} \Big(\|\mathbf{V}\|_2^2 + \|\mathbf{E}\|_2^2 + \sum_{r \in \mathcal{R}} \|\mathbf{R}\|_2^2\Big) \end{aligned} \tag{13}$$

where $\mathbf{V}$ and $\mathbf{E}$ are the embedding matrices for all items and entities, respectively, $\mathbf{I}_r$ is the slice of the indicator tensor $\mathbf{I}$ in KG for relation r, and $\mathbf{R}$ is the embedding matrix of relation r. In Eq. (13), The first term measures the cross-entropy loss between ground truth of interactions Y and predicted value by Ripple Network, the second term measures the squared error between the ground truth of the KG $\mathbf{I}_r$ and the reconstructed indicator matrix $\mathbf{E}^{\mathrm{T}} \mathbf{R} \mathbf{E}$, and the third term is the regularizer for preventing over-fitting.

Algorithm 1 Learning algorithm for Ripple Network

Input: Interaction matrix Y, knowledge graph $\mathcal{G}$
Output: Prediction function $\mathcal{F}(u, v | \Theta)$

Initialize all parameters
Calculate ripple sets $\{\mathcal{S}_u^k\}_{k=1}^H$ for each user u;
for number of training iteration **do**
    Sample minibatch of positive and negative interactions from Y;
    Sample minibatch of true and false triples from $\mathcal{G}$;
    Calculate $\partial \mathcal{L} / \partial \mathbf{V}$, $\partial \mathcal{L} / \partial \mathbf{E}$, $\{\partial \mathcal{L} / \partial \mathbf{R}\}_{r \in \mathcal{R}}$, and $\{\partial \mathcal{L} / \partial \alpha_i\}_{i=1}^H$ on the minibatch by back-propagation according to Eq. (4)-(13);
    Update $\mathbf{V}$, $\mathbf{E}$, $\{\mathbf{R}\}_{\mathrm{r} \in \mathcal{R}}$, and $\{\alpha_i\}_{i=1}^H$ by SGD with learning rate η;
end for
return $\mathcal{F}(u, v | \Theta)$

It is intractable to solve the above objection directly, therefore, we employ a stochastic gradient descent (SGD) algorithm to iteratively optimize the loss function. The learning algorithm of Ripple Network is presented in Algorithm 1. In each training iteration, to make the computation more efficient, we randomly sample a minibatch of positive/negative interactions from $\mathbf{Y}$ and true/false triples from $\mathcal{G}$ following the negative sampling strategy in [15]. Then we calculate the gradients of the loss $\mathcal{L}$ with respect to model parameters Θ, and update all parameters by back-propagation based on the sampled minibatch.

3.5 Discussion

3.5.1 Explainability. Explainable recommender systems [24] aim to reveal why a user might like a particular item, which helps improve their acceptance or satisfaction of recommendations and increase trust in RS. The explanations are usually based on community tags [25], social networks [21], and aspect [2]. Since Ripple Network explores users' interests based on the KG, it provides a new point of view of explainability by tracking the paths from a user's history to an item with high relevance probability (Eq. (4)) in the KG. For example, a user's interest in film "Back to the Future" might be explained by the path "$user \xrightarrow{watched} Forrest\ Gump \xrightarrow{directed\ by} Robert\ Zemeckis \xrightarrow{directs} Back\ to\ the\ Future$", if the item "Back to the Future" is of high relevance probability with "Forrest Gump" and "Robert Zemeckis" in the user's 1-hop and 2-hop ripple set, respectively. Note that different from path-based methods [34, 36] where the patterns of path are manually designed, Ripple Network automatically discovers the possible explanation paths according to relevance probability. We will further present a visualized example in the experiments part.

3.5.2 Ripple Superposition. A common phenomenon in Ripple Network is that a user's ripple sets may be large in size, which dilutes his potential interests inevitably in preference propagation. However, we observe that relevant entities of different items in a user's click history often highly overlap. In other words, an entity could be reached by multiple paths in the KG starting from a user's click history. For example, "Saving Private Ryan" is connected to a user who has watched "The Terminal", "Jurassic Park" and "Braveheart" through actor "Tom Hanks", director "Steven Spielberg" and genre "War", respectively. These parallel paths greatly increase a user's interests in overlapped entities. We refer to the case as *ripple superposition*, as it is analogous to the interference phenomenon in physics in which two waves superpose to form a resultant wave of greater amplitude in particular areas. The phenomenon of ripple superposition is illustrated in the second KG in Figure 1, where the darker red around the two lower middle entities indicates higher strength of the user's possible interests. We will also discuss ripple superposition in the experiments part.

3.6 Links to Existing Work

Here we continue our discussion on related work and make comparisons with existing techniques in a greater scope.

3.6.1 Attention Mechanism. The attention mechanism was originally proposed in image classification [17] and machine translation [1], which aims to learn where to find the most relevant part of the input automatically as it is performing the task. The idea was soon transplanted to recommender systems [4, 28, 31]. For example, DADM [4] considers factors of specialty and date when assigning attention values to articles for recommendation. Ripple Network can be viewed as a special case of attention where tails are averaged weighted by similarities between their associated heads, tails, and certain item. The difference between our work and literature is that Ripple Network designs a multi-level attention module based on knowledge triples for preference propagation.

3.6.2 Memory Networks. Memory networks [16, 22] is a recurrent attention model that utilizes an external memory module for question answering and language modeling. The iterative reading operations on the external memory enable memory networks to extract long-distance dependency in texts. Researchers have also proposed using memory networks in other tasks such as sentiment classification [12] and recommendation [5, 8]. Note that these works usually focus on entry-level or sentence-level memories, while our work addresses entity-level connections in the KG, which is more fine-grained and intuitive when performing multi-hop iterations. In addition, our work also incorporates a KGE term as a regularizer for more stable and effective learning.

3.6.3 Knowledge Graph Embedding. Ripple Network also connects to a large body of work in KGE methods. KGE intends to embed entities and relations in a KG into continuous vector spaces while preserving its inherent structure. Readers can refer to [30] for a more comprehensive survey. KGE methods are mainly classified into two categories: (1) translational distance models, such as TransE [3], TransH [32], and TransR [13], exploit distance-based scoring functions when learning representations of entities and relations; and (2) semantic matching models, such as RESCAL [19], ANALOGY [18], and HolE [14], measure plausibility of knowledge triples by matching latent semantics of entities and relations. Researchers also propose incorporating auxiliary information, such as entity types, logic rules, and textual descriptions to assist KGE. However, these methods are more suitable for in-graph applications such as link prediction or triple classification, according to their learning objectives. From this point of view, Ripple Network can be seen as a KGE method that serves recommendation directly.

4 EXPERIMENTS

In this section, we evaluate Ripple Network on three real-world scenarios: movie, book and news recommendations[5].

4.1 Datasets

We utilize the following three datasets in our experiments for movie, book and news recommendation:

- MovieLens-1M[6] is a widely used benchmark dataset in movie recommendations consisting of approximately 1 million explicit ratings (ranging from 1 to 5) on the MovieLens website.
- Book-Crossing[7] contains 1,149,780 ratings (ranging from 0 to 10) of books in the Book-Crossing community.

[5]Experiment code is provided at https://github.com/hwwang55/ripple_network.
[6]https://grouplens.org/datasets/movielens/1m/
[7]http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Table 1: Basic statistics of the three datasets.

	MovieLens-1M	Book-Crossing	Bing-News
# users	6,036	26,245	141,487
# items	2,355	24,124	535,145
# interactions	752,935	174,132	1,025,192
# 1-hop triples	20,837	228,722	503,112
# 2-hop triples	178,501	231,815	1,748,562
# 3-hop triples	318,826	492,408	3,997,736
# 4-hop triples	824,955	615,743	6,322,548

Table 2: Hyper-parameter settings for the three datasets.

MovieLens-1M	$d = 32$, $H = 2$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 0.1$, $\eta = 0.005$
Book-Crossing	$d = 64$, $H = 2$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 0.1$, $\eta = 0.005$
Bing-News	$d = 32$, $H = 3$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 0.5$, $\eta = 0.005$

- Bing-News dataset contains 1,025,192 pieces of implicit feedback collected from the server logs of Bing News[8] during the period 10/16/2016–08/11/2017. Each piece of news has a title and a snippet.

Since MovieLens-1M and Book-Crossing are explicit feedback data, we transform them into implicit feedback where each entry is marked with 1 indicating that the user has rated the item with high marks (4/5 in MovieLens-1M and 8/9/10 in Book-Crossing), and sample an unwatched set marked as 0 for each user, which is of equal size with the rated ones. For MovieLens-1M and Book-Crossing, we use the ID embeddings of users and items as raw input, while for Bing-News, we concatenate the ID embedding of a piece of news and the averaged word embedding of its title as raw input for the item, since news titles are typically much longer than movie/book names and can provide more useful information.

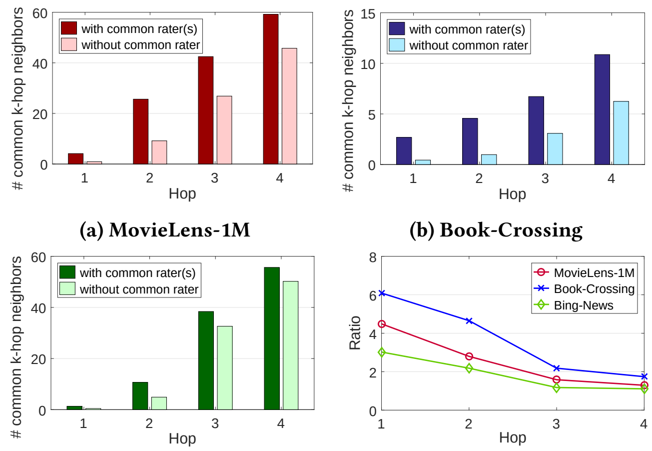
We use Microsoft Satori to construct the knowledge graph for each dataset. For MovieLens-1M and Book-Crossing, we first select a subset of triples from the whole KG whose relation name contains "movie" or "book" and the confidence level is greater than 0.9. Given the sub-KG, we match the names of all movies/books with its knowledge triples and select all well-matched entities, and extend the set of entities iteratively up to four hops. 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 three datasets are presented in Table 1.

4.2 Baselines

We compare our method with the following baselines:

- **CKE** [35] combines CF with structural knowledge, textual knowledge, and visual knowledge in a unified framework for recommendation. We implement CKE as CF plus structural knowledge module in this paper.
- **SHINE** [27] designs deep autoencoders to embed a sentiment network, social network, and profile (knowledge) network for celebrity recommendation. Here we use autoencoders for user-item interaction and item profile to predict click probability.
- **DKN** [28] 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.
- **PER** [34] treats the KG as HIN and extracts meta-path based features to represent the connectivity between users and items. In this paper, we use all item-attribute-item features for PER (e.g., “movie-director-movie").
- **LibFM** [20] is a widely used feature-based factorization model in CTR scenarios. We concatenate user ID, item ID, and the corresponding averaged entity embeddings learned from TransR [13] as input for LibFM.
- **DeepWide** [6] is a general deep model for recommendation combining a (wide) linear channel with a (deep) non-linear channel. Similar to LibFM, we use the embeddings of users, items, and entities to feed DeepWide.

[8]https://www.bing.com/news



(a) MovieLens-1M
(b) Book-Crossing
(c) Bing-News
(d) Ratio of two average numbers

Figure 3: The average number of k-hop neighbors that two items share in the KG w.r.t. whether they have common raters in (a) MovieLens-1M, (b) Book-Crossing, and (c) Bing-News datasets. (d) The ratio of the two average numbers with different hops.

4.3 Experiment Setup

In Ripple Network, we set the hop number $H = 2$ for MovieLens-1M/Book-Crossing and $H = 3$ for Bing-News. A larger number of hops hardly improves performance but does incur heavier computational overhead according to experiment results. The complete hyper-parameter settings are given in Table 2, where d denotes the dimension of embedding for items and the knowledge graph, and η denotes the learning rate. The hyper-parameters are determined by optimizing *AUC* on a validation set. For fair consideration, the latent dimensions of all compared baselines are set the same as in Table 2, while other hyper-parameters of baselines are set as default or as reported in original papers.

For each dataset, we randomly select 80% of ratings for the training set and use the remaining ones as the test set. Each experiment is repeated 5 times, and the average performance is reported. We use *Accuracy* and *AUC* to evaluate the performance of CTR prediction and use *Precision@K*, *Recall@K*, *F1@K* to evaluate the performance of top-K recommendation for all methods.

Table 3: The results of *AUC* and *Accuracy* in CTR prediction.

Model	MovieLens-1M		Book-Crossing		Bing-News	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
Ripple*	**0.913**	**0.835**	**0.840**	**0.775**	**0.778**	**0.732**
CKE	0.796	0.739	0.634	0.606	0.660	0.617
SHINE	0.778	0.732	0.668	0.636	0.614	0.587
DKN	0.655	0.589	0.621	0.598	0.761	0.704
PER	0.901	0.826	0.814	0.735	-	-
LibFM	0.892	0.812	0.763	0.705	0.744	0.688
DeepWide	0.903	0.822	0.806	0.731	0.754	0.695

* Statistically significant improvements by t-test.

4.4 Empirical Study

We conduct an empirical study to investigate the correlation between the average number of common neighbors of an item pair in the KG and whether they have common rater(s) in RS. For each dataset, we first randomly sample one million item pairs, then count the average number of k-hop neighbors that the two items share in the KG under the following two circumstances: (1) the two items have at least one common rater in RS; (2) the two items have no common rater in RS. The results are presented in Figures 3a, 3b, 3c, respectively, which clearly show that if two items have common rater(s) in RS, they likely share more common k-hop neighbors in the KG for fixed k. The above findings empirically demonstrate that *the similarity of proximity structures of two items in the KG could assist in measuring their relatedness in RS*. In addition, we plot the ratio of the two average numbers with different hops (i.e., dividing the higher bar by its immediate lower bar for each hop number) in Figure 3d, from which we observe that the proximity structures of two items under the two circumstances become more similar with the increase of the hop number. This is because any two items are probable to share a large amount of k-hop neighbors in the KG for a large k, even if there is no direct similarity between them in reality. The result motivates us to find a moderate hop number in Ripple Network to explore users' potential interests as far as possible while avoiding introducing too much noise.

4.5 Results

The results of all methods in CTR prediction and top-K recommendation are presented in Table 3 and Figures 4, 5, 6, respectively. Several observations stand out:

- CKE performs comparably poorly than other baselines, which is probably because we only have structural knowledge available, without visual and textual input.
- SHINE performs better in movie and book recommendation than news. This is because the 1-hop triples for news are too complicated when taken as profile input.
- DKN performs best in news recommendation compared with other baselines, but performs worst in movie and book recommendation. This is because movie and book names are too short and ambiguous to provide useful information in the form of word embedding.
- PER performs well on movie and book recommendation, however, it cannot be applied in news recommendation since the types of entities and relations involved in news are too complicated to pre-define meta-paths.

Table 4: The results of *AUC* w.r.t. different ratio of KG triples.

Ratio of KG triples	0.5	0.6	0.7	0.8	0.9	1.0
MovieLens-1M	0.901	0.905	0.908	0.912	0.913	0.913
Book-Crossing	0.832	0.833	0.837	0.839	0.841	0.840
Bing-News	0.757	0.765	0.772	0.775	0.775	0.778

Table 5: The results of *AUC* w.r.t. different hop numbers.

Hop number H	1	2	3	4
MovieLens-1M	0.911	0.914	0.912	0.907
Book-Crossing	0.833	0.840	0.839	0.838
Bing-News	0.762	0.776	0.779	0.774

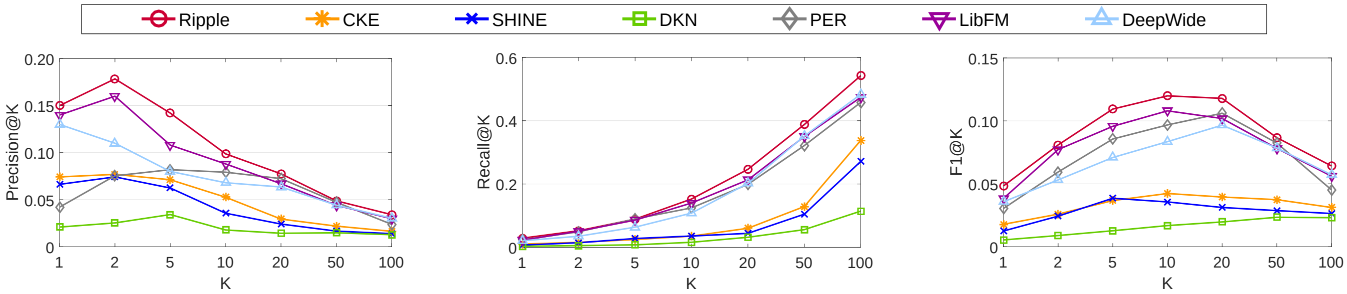
- As two generic recommendation tools, LibFM and DeepWide achieve satisfactory performance, demonstrating that they can make well use of knowledge from KG into their algorithms.
- Ripple Network performs best among all methods in the three datasets. Specifically, Ripple Network outperforms baselines by 1.1% to 39.4%, 3.2% to 35.3%, and 2.2% to 26.7% on *AUC* in movie, book, and news recommendation, respectively. Ripple Network also achieves outstanding performance in top-K recommendation as shown in Figures 4, 5, and 6.

Efficacy of usage of KG. We vary the size of KG triples to further investigate the robustness of Ripple Network. The results of *AUC* on the three datasets are presented in Table 4, from which we observe that the performance of Ripple Network drops slightly with the decrease of the number of used KG triples. Specifically, the *AUC* score declines by 1.3%, 1.0% and 2.7% with KG ratio decreasing from 1.0 to 0.5 in three scenarios, respectively. The results prove that Ripple Network can still maintain a decent performance even if the KG is sparse.

Hop number. We also vary the size of KG by the maximal hop number H to see how performance changes in Ripple Network. The results are shown in Table 5, which clearly shows that with the increase of hop number H, the *AUC* score increases initially but falls at last when H is large (typically when $H = 3$). We attribute the phenomenon to the trade-off between the positive signals from long-distance dependency and negative signals from noises: too small of an H can hardly explore inter-entity relatedness and dependency of long distance, while too large of an H brings much more noises than useful signals, as stated in Section 4.4.

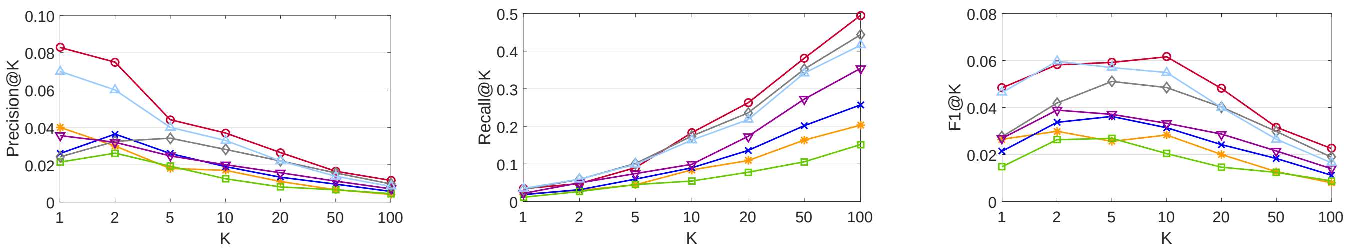
4.6 Case Study

To intuitively demonstrate the preference propagation in Ripple Network, we randomly sample a user with 4 clicked pieces of news, and select one candidate news from his test set with label 1. For each of the user's k-hop relevant entities, we calculate the (unnormalized) relevance probability between the entity and the candidate news or its k-order responses. The results are presented in Figure 7, in which the darker shade of blue indicates larger values, and we omit names of relations for clearer presentation. From Figure 7 we observe that Ripple Network associates the candidate news with the user's relevant entities with different strengths. The candidate news can be reached via several paths in the KG with high weights from the user's click history, such as "Navy SEAL"–"Special Forces"–"Gun"–"Police". These highlighted paths automatically discovered by preference propagation can thus be used to explain the



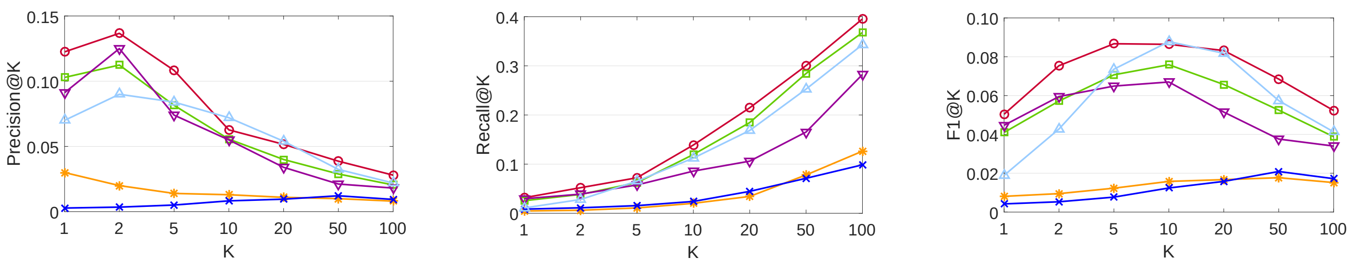
(a) *Precision@K* **(b)** *Recall@K* **(c)** *F1@K*

Figure 4: ***Precision@K*, *Recall@K*, and *F1@K* in top-*K* recommendation for MovieLens-1M.**



(a) *Precision@K* **(b)** *Recall@K* **(c)** *F1@K*

Figure 5: ***Precision@K*, *Recall@K*, and *F1@K* in top-*K* recommendation for Book-Crossing.**



(a) *Precision@K* **(b)** *Recall@K* **(c)** *F1@K*

Figure 6: ***Precision@K*, *Recall@K*, and *F1@K* in top-*K* recommendation for Bing-News.**

recommendation result, as discussed in Section 3.5.1. Additionally, it is also worth noticing that several entities in the KG receive more intensive attention from the user's history, such as "U.S.", "World War II" and "Donald Trump". These central entities result from the ripple superposition discussed in Section 3.5.2, and can serve as the user's potential interests for future recommendation.

4.7 Parameter Sensitivity

In this section, we investigate the influence of parameters d and λ_2 in Ripple Network. We vary d from 8 to 256 and λ_2 from 0.01 to 5, respectively, while keeping other parameters fixed. The results of $F1@1$ on the three datasets are presented in Figure 8. We observe from Figure 8a that, with the increase of d, the performance is boosted at first since embeddings with a larger dimension can encode more useful information, but drops after $d = 32$ or 64 due to possible overfitting. From Figure 8b, we can see that Ripple Network achieves the best performance when $\lambda_2 = 0.1$ or 0.5. This is because the KGE term with a small weight cannot provide enough regularization constraints, while a large weight will mislead the objective function.

5 CONCLUSIONS

In this paper, we propose Ripple Network, an end-to-end framework that naturally incorporates the knowledge graph into recommender systems. Ripple Network overcomes the limitations of existing embedding-based and path-based KG-aware recommendation methods by introducing preference propagation, which automatically propagates users' potential preferences and explores their hierarchical interests in the KG. Ripple Network unifies the preference propagation with regularization of KGE in a Bayesian framework for click-through rate prediction. We conduct extensive experiments in three recommendation scenarios. The results demonstrate the significant superiority of Ripple Network over strong baselines. For future work, we plan to further investigate the methods of characterizing entity-relation interactions as well as how to effectively reduce KG size while maintaining decent performance in Ripple Network.

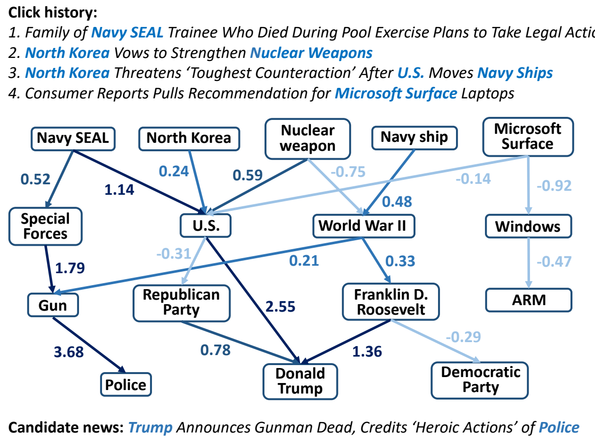
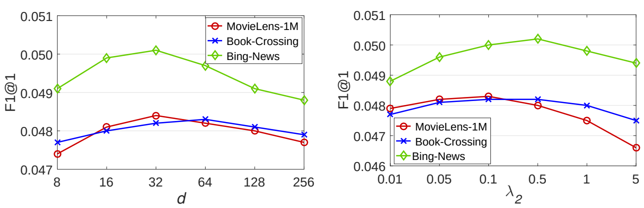


Figure 7: Visualization of relevance probabilities for a randomly sampled user w.r.t. a piece of candidate news with label 1. Links with value lower than -1.0 are omitted.



(a) Dimension of embedding **(b) Training weight of KGE term**

Figure 8: Parameter sensitivity of Ripple Network.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 717–725.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.

[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*. ACM, 335–344.

[5] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential Recommendation with User Memory Networks. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[7] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *ACL*. 260–269.

[8] Haoran Huang, Qi Zhang, and Xuanjing Huang. 2017. Mention Recommendation for Twitter with End-to-end Memory Network. In *IJCAI*.

[9] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM conference on Recommender systems*. ACM, 135–142.

[10] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[12] Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-End Adversarial Memory Network for Cross-domain Sentiment Classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence.*

[13] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. 2181–2187.

[14] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-Relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. 2168–2178.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[16] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* (2016).

[17] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*. 2204–2212.

[18] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic Embeddings of Knowledge Graphs. In *AAAI*. 1955–1961.

[19] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning*. 809–816.

[20] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[21] Amit Sharma and Dan Cosley. 2013. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 1133–1144.

[22] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. 2440–2448.

[23] Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. 2017. Collaborative Intent Prediction with Real-Time Contextual Data. *ACM Transactions on Information Systems* 35, 4 (2017), 30.

[24] Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *IEEE 23rd International Conference on Data Engineering Workshop*. IEEE, 801–810.

[25] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th international conference on Intelligent user interfaces*. ACM, 47–56.

[26] Hongwei Wang, Jia Wang, Miao Zhao, Jiannong Cao, and Minyi Guo. 2017. Joint Topic-Semantic-aware Social Recommendation for Online Voting. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*. ACM, 347–356.

[27] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*.

[28] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: deep knowledge-aware network for news recommendation. In *Proceedings of the 27th International Conference on World Wide Web Companion*. ACM.

[29] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *IJCAI*.

[30] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[31] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2051–2059.

[32] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. 1112–1119.

[33] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 1219–1228.

[34] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. 283–292.

[35] 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.

[36] 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.