Ekar: An Explainable Method for Knowledge Aware Recommendation

Weiping Song*†
Department of Computer Science, School of EECS, Peking University
weiping.song@pku.edu.cn

Zhijian Duan*
Department of Computer Science, School of EECS, Peking University
zjduan@pku.edu.cn

Ziqing Yang
Department of Computer Science, School of EECS, Peking University
yangziqing@pku.edu.cn

Hao Zhu
Department of Computer Science, School of EECS, Peking University
hzhu1998@pku.edu.cn

Ming Zhang
Department of Computer Science, School of EECS, Peking University
mzhang_cs@pku.edu.cn

Jian Tang
Mila-Quebec AI Institute, HEC Montreal & CIFAR AI Chair
jian.tang@hec.ca

arXiv:1906.09506v2 [cs.IR] 6 Jan 2022

ABSTRACT

This paper studies recommender systems with knowledge graphs, which can effectively address the problems of data sparsity and cold start. Recently, a variety of methods have been developed for this problem, which generally try to learn effective representations of users and items and then match items to users according to their representations. Though these methods have been shown quite effective, they lack good explanations, which are critical to recommender systems. In this paper, we take a different route and propose generating recommendations by finding meaningful paths from users to items. Specifically, we formulate the problem as a sequential decision process, where the target user is defined as the initial state, and the edges on the graphs are defined as actions. We shape the rewards according to existing state-of-the-art methods and then train a policy function with policy gradient methods. Experimental results on three real-world datasets show that our proposed method not only provides effective recommendations but also offers good explanations.

CCS CONCEPTS

• **Information systems** → *Recommender systems*; • **Computing methodologies** → *Sequential decision making*; Neural networks.

KEYWORDS

KG-based Recommendation, Explainability, Reasoning, Deep Reinforcement Learning

1 INTRODUCTION

Recommender systems are essential to a variety of online applications such as e-Commerce Websites and social media platforms by providing the right items or information to the users. One critical problem of recommender systems is data sparsity, i.e., some items are purchased, rated, or clicked by only a few users or no users at all. Recently, there is an increasing interest in knowledge graph-based recommender systems, since knowledge graphs can provide complementary information to alleviate the problem of data sparsity and have been proved quite useful [3, 15, 23, 24, 35, 37].

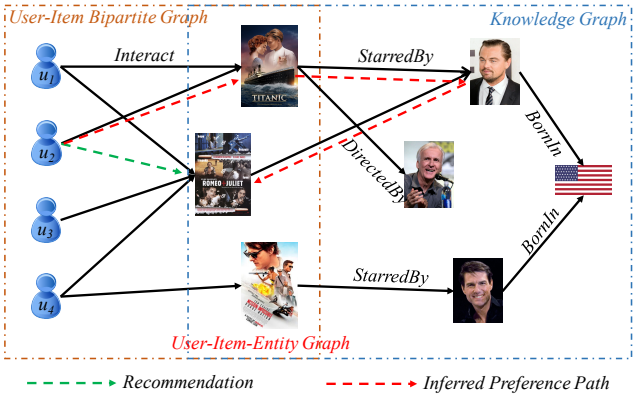

Figure 1: Explainable recommendation via reasoning over the integrated user-item-entity graph. Our Ekar generates the recommendation (i.e., "Romeo and Juliet") by inferring a preference path "$u_2 \xrightarrow{Interact}$ Titanic $\xrightarrow{StarredBy}$ LeonardoDiCaprio $\xrightarrow{StarredIn}$ Romeo and Juliet".

Generally, existing knowledge graph-based recommendation methods try to learn effective representations of users and items according to the user-item interaction graphs and item-entity knowledge graphs, and then match the items to the users according to learned representations. For example, Zhang et al. [35] learn item representations by combining their representations in user-item graphs and knowledge-graphs. Zhang et al. [36] learn user and item representations on the integrated user-item-entity graphs based on knowledge graph embedding (KGE) method like TransE [2]. Wang et al. [25] and Cao et al. [3] jointly optimize the recommendation and knowledge graph embedding tasks in a multi-task learning setting via sharing item representations. These methods have been proved quite effective by integrating information from both user behaviors and knowledge graphs.

Although these methods are very effective, they lack good explanations. Intuitively, if the recommender system can give an

*Both authors contributed equally to this research.
†This work was done when the first author was visiting Mila.

,
.

explanation of a recommendation, the users would have more interest and trust in the recommended item [9, 14, 37, 38]. Indeed, there is some existing work that aims to provide such explanations for the recommendation results. For example, the RippleNet [23] aims to explain the recommendations by analyzing the attention scores. However, their method relies on the post analysis of the soft attention scores, which may not always be trustworthy.

In this paper, we take a different route and propose to generate a path from the target user to relevant items in the integrated user-item-entity graph. Take the movie recommendation in Figure 1 as an example. For the target user u_2, a path (the red dashed lines) is generated to the item "Romeo and Juliet" since 1) user u_2 watched movie "Titanic"; 2) "Titanic" is starred by "Leonardo DiCaprio"; and 3) "Leonardo DiCaprio" also stars in "Romeo and Juliet". We can see that such a path offers good explanations of the recommendations in addition to provide meaningful recommendations.

However, finding meaningful paths on the large user-item-entity graph is challenging. One may enumerate all paths between user-item pairs and then use a classification/ranking model to select the most meaningful paths [28]. Nevertheless, enumerating paths between users and items is intractable due to the exponentially large path space in the user-item-entity graph. Although sampling a certain number of paths via breadth-first-search can be a practical substitution to enumerating, it has no assurance on the meaningfulness of sampled paths. In this paper, we instead formulate the generation of meaningful user-to-item paths as a sequential decision process. Specifically, the recommender agent starts from target users and extends its paths to relevant items by sequentially selecting walks on the user-item-entity graph. During training, we assign each path a positive reward if the starting user and terminal entity constitute an observation in recommendation data. Considering that the reward could be extremely sparse at the beginning due to the huge exploration space, we further augment the reward signals by reward shaping [17], where a soft reward function is first learned using state-of-the-art knowledge graph embedding methods [5, 21, 33]. We use the REINFORCE [29] algorithm to maximize the expected rewards of our recommender agent. Finally, we verify the effectiveness and the explainability of the proposed method on three real-world datasets. Quantitative results demonstrate that our proposed Ekar[1] model: 1) significantly outperforms existing state-of-the-art KG-based recommendation methods, 2) offers clear and convincing explanations in the form of meaningful paths from users to recommended items.

To summarize, the contributions of this work are as follows:

- We propose generating (rather than discriminating) meaningful paths from users to items, where the end of paths (i.e., items) are recommendations and the paths themselves are explanations w.r.t. the recommendations.
- We propose a novel deep reinforcement learning-based approach to infer such meaningful paths. To encourage exploration and stabilize training, we design a well-shaped reward function based on existing knowledge graph representation learning methods.
- We conduct extensive experiments to validate the proposed approach. Experimental results show that our model significantly outperforms previous state-of-the-art method (KTUP [3]) with at most 31.31% hit ratio gain for top-N recommendation. What's more, studies on recommendation paths show good explainability of our approach.

[1]Ekar represents **E**xplainable **k**nowledge **a**ware **r**ecommendation.

Organization. Section 2 reviews related work. In Section 3, we give formal definitions used in this paper. Section 4 introduces our proposed deep reinforcement learning-based approach for explainable knowledge graph-based recommendation. We introduce two mechanisms for action selection in Section 5. Extensive experimental results and analysis are presented in Section 6, followed by the conclusion of our work in Section 7.

2 RELATED WORK

Our work is conceptually related to the explainable recommendation, knowledge graph-based recommendation, and recent advancements in applying reinforcement learning into relational reasoning.

2.1 Explainable Recommendation

As a widespread concern in the AI community, explainability has been widely discussed in recommender systems. According to Zhang and Chen [37], most of the existing explainable recommendation methods typically provide explanations via identifying users' preference on item features [1, 26, 38], understanding latent factors with topic modeling [16, 30, 40], or ranking over the user-item-aspect graph [8]. However, these methods require external information (e.g., reviews) about items, which may be difficult to collect. Some recent advancements utilize the attention mechanism [12, 20] to provide explanations, but they need extra efforts to explore attention scores. Since knowledge graphs (KG) provide common knowledge about our world, many recent works use knowledge graphs [3, 23, 24, 35] to provide explainable recommendations, which will be further discussed in the following paragraph.

2.2 KG-based Recommendation

Our work is closely related to KG-based recommendation, which utilizes general knowledge graphs (e.g., DBpedia, YAGO, and Satori) to improve recommender systems. Existing KG-based recommendation methods can be roughly divided into two classes: embedding-based methods and path-based methods. In embedding-based methods, users and items are represented by low-dimensional vectors, where entities' embeddings from the knowledge graph are used to enhance corresponding items' representations [3, 24, 25, 27, 35]. Although these methods perform well, it's hard to explain the recommendation results because representations are in a latent space. In path-based methods, meta-paths and meta-graphs are commonly used to extract various semantic dependencies between users and items [34, 39]. However, it is almost computationally infeasible to enumerate all the useful meta-paths or meta-graphs. Moreover, the meta-paths and meta-graphs need to be manually defined and cannot generalize to new datasets. Instead of pre-defining specific paths, the RippleNet [23] directly propagates users' preferences along edges in KG via the attention mechanism and then interprets the recommendations according to the attention scores, which however might not be trustworthy. The most recent work KPRN [28] uses LSTM to model the paths between users and items. However, sampling paths via breadth-first-search (BFS) is inefficient and may

miss meaningful paths. Different from KPRN, our method defines the path finding problem as a sequential decision problem. We train an agent to automatically generate a meaningful path between a user and his/her relevant item via policy gradient methods. A concurrent work to ours is PGPR [31], which also formulate the recommendation task as a sequential decision process over knowledge graphs. The major differences are two folds: 1) we encode the complete path history as current state while PGPR only considers a small portion of it; 2) we first pre-train a state-of-the-art KGE model for reward shaping and therefore achieve much better performance.

2.3 Relational Reasoning with Reinforcement Learning

Our work is also related to recent work on knowledge graph reasoning with reinforcement learning [4, 13, 32], which aims to train an agent to walk on knowledge graphs to predict the missing facts. However, their goal is different from ours. We focus on the problem of recommendation with knowledge graphs and aim at finding meaningful paths for explaining the recommendation results while they focus on facts prediction.

3 DEFINITIONS

To relieve the data sparsity issue, incorporating auxiliary knowledge about items has been attracting increasing attention[3, 23, 24, 35, 37]. Typically external knowledge is represented by a graph, where the nodes are entities and the edges are relations between two entities. Since some of entities in knowledge graphs and some of items in user-item graphs can be aligned, we can merge the knowledge graphs and the user-item graphs into an integrated user-item-entity graph, which is formally defined as follows:

DEFINITION 1. **(User-item-entity Graph)** Let $\mathcal{G} = (\mathcal{U}, \mathcal{I})$ denote the user-item bipartite graph, where $\mathcal{U}$ is the set of users, and $\mathcal{I}$ is the set of items. Besides, we also have access to an open knowledge graph $\mathcal{G}_k = (\mathcal{E}_k, \mathcal{R}_k)$, where $\mathcal{E}_k$ is the entity set and $\mathcal{R}_k$ is the relation set. Each triplet $< e_h, r, e_t > \in \mathcal{G}_k$ indicates there exists a relation $r \in \mathcal{R}_k$ from head entity $e_h \in \mathcal{E}_k$ to tail entity $e_t \in \mathcal{E}_k$. For example, $<$ *Titanic, DirectedBy, JamesCameron* $>$ reflects the fact that "Titanic" is directed by "James Cameron". As some items/entities are shared in $\mathcal{I}$ and $\mathcal{E}_k$, we merge the user-item bipartite graph $\mathcal{G}$ and the knowledge graph $\mathcal{G}_k$ into an integrated *user-item-entity graph* $\mathcal{G}' = (\mathcal{V}', \mathcal{R}')$, where $\mathcal{V}' = \mathcal{U} \cup \mathcal{I} \cup \mathcal{E}_k$. For the user-item interaction graph $\mathcal{G}$, we assume all the edges belong to a special relation "Interact", and therefore $\mathcal{R}' = \{\text{"Interact"}\} \cup \mathcal{R}_k$. An example of such user-item-entity graph could be Figure 1.

Typically users do not consume items at random. In other words, there should be *implicit reasons* why a specific item is selected by a user. As shown by the red line in Figure 1, a user may choose to watch movies starred by the same actor who showed superb acting skills in the movie they watched before. To improve users' acceptance of recommendations, we therefore propose to infer such meaningful user-to-item paths to *explicitly* provide explanations. We define the resulting path-based explainable recommendation problem as follows:

DEFINITION 2. **(Path-based explainable recommendation)** Given a user u, *path-based explainable recommendation* task aims to generate a set of paths from u to relevant items on the user-item-entity graph $\mathcal{G}'$. Such paths not only allow to find the relevant items but also offer good explanations.

4 METHODOLOGY

In this section, we describe our proposed Ekar model for path-based explainable recommendation task in detail.

Overall, we formulate the generation of explainable paths as a Markov Decision Process (MDP) on the user-item-entity graph, and we use deep reinforcement learning to solve it. As shown in Figure 2, the user-item-entity is treated as the environment, from which we get the observation s_t, i.e., a sequence of visited nodes and edges. Based on the encoded state $\mathbf{s}_t$, a policy network consisting of two fully-connected layers outputs the probability distribution over possible action space. Finally, we define the reward being +1 if our agent successfully finds those items consumed by the target users u in history. Furthermore, we augment the rewards based on pre-trained state-of-the-art KGE models for the purpose of stabilizing training and encouraging the agent to explore diverse recommendation paths.

4.1 Formulating Recommendation as a Markov Decision Process

There are some existing methods [28] trying to find meaningful paths between users and items. These methods first sample a collection of paths with breadth-first or depth-first search strategy and then measure the meaningfulness of the paths with a classification or ranking model. However, the number of possible paths between a user and an item could be exponentially large, and sampling a few of them could miss the meaningful ones. Moreover, the paths sampled through BFS or DFS strategy may not always be meaningful. In this paper, we take a different route and formulate the problem as a sequential decision making problem on the user-item-entity graph $\mathcal{G}'$. We aim to train an agent to walk on $\mathcal{G}'$ to find relevant items. Starting from a target user u, the agent sequentially selects the next neighbor on the integrated user-item-entity graph $\mathcal{G}'$ until it reaches the predefined maximum number of steps T. Formally, we define the states, actions, transition, and rewards of the Markov Decision Process as follows:

States. We represent the state as the sequence of traversed relations and entities so far, i.e., $s_t = (r_0, e_0, r_1, e_1, ..., r_t, e_t) \in \mathcal{S}_t$, where $r_t \in \mathcal{R}'$ and $e_t \in \mathcal{V}'$ are relations and entities respectively. The initial state $s_0 = (r_0, e_0)$ represents the target user, and r_0 is an artificially introduced relation to be consistent with other (r_t, e_t) pairs.

Actions. When the agent is under state s_t, it can choose an outgoing edge of entity e_t as its next action. Formally, we define the possible actions under state s_t as $\mathcal{A}_t = \{a = (r', e') | (e_t, r', e') \in \mathcal{G}'\}$.

Transition. For the state transition $\mathcal{P}(\mathcal{S}_{t+1} = s | \mathcal{S}_t = s_t, \mathcal{A}_t = a_t)$, we adopt a deterministic strategy and simply extend the current state s_t by adding the new action $a_t = (r_{t+1}, e_{t+1})$ as the next state, i.e., $s_{t+1} = (r_0, e_0, ..., r_t, e_t, r_{t+1}, e_{t+1})$.

Rewards. No intermediate reward is provided for $(s_{t<T}, a_{t<T})$. The final reward depends on whether or not the agent correctly finds interacted items of the user u. Given the terminal entity e_T, the

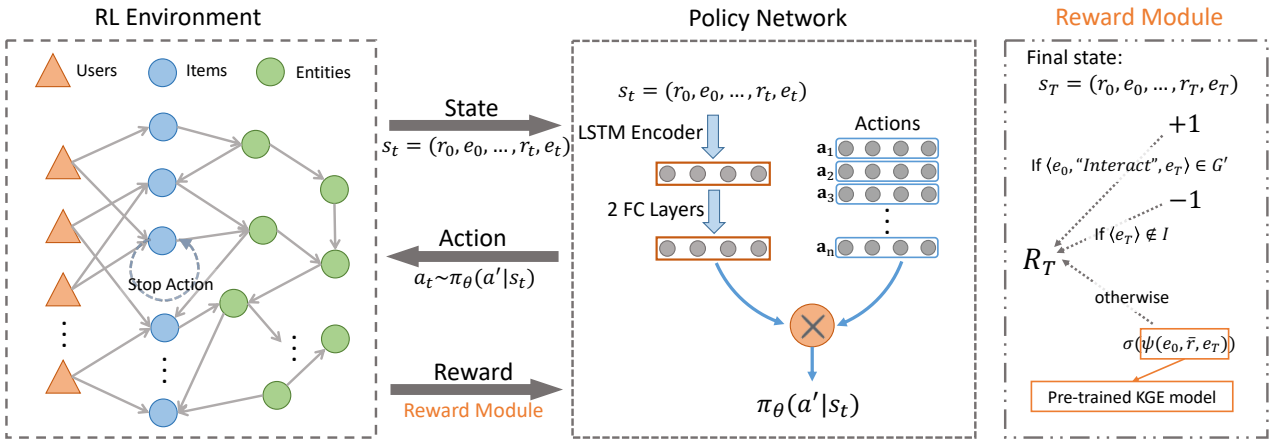

Figure 2: A schematic view of our proposed Ekar model. We formulate the generation of explainable paths as a sequential decision process. The user-item-entity graph is the RL environment and a policy network interacts with it. To stabilize training and encourage the agent to explore diverse paths, we design a well-shaped reward function.

final reward R_T is +1 if user e_0 has interacted with e_T, 0 if e_T is an item but user e_0 has not interacted with it and −1 if e_T is not an item-type entity.

4.2 Solving Recommendation MDP with Policy-based Reinforcement Learning

Next, we introduce a deep policy-based reinforcement learning method to solve the above MDP. The learned policy is then used to generate recommendations and explanations during inference.

4.2.1 Architecture of Policy Network. Since there are usually millions of entities and hundreds of relations in user-item-entity graph $\mathcal{G}'$, it is almost impossible to utilize discrete states and actions directly, the number of which is exponential to the number of symbolic atoms in s_t and a_t respectively. We, therefore, choose to represent entities and relations in $\mathcal{G}'$ with low-dimensional embeddings. Each action $a = (r, e)$ is represented as the concatenation of relation and entity embeddings, i.e., $\mathbf{a} = [\mathbf{r}'; \mathbf{e}']$. The state $s_t = (r_0, e_0, ..., r_t, e_t)$ is encoded by an LSTM [10]:

$$\begin{aligned} \mathbf{s}_0 &= LSTM(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_0]), \\ \mathbf{s}_t &= LSTM(\mathbf{s}_{t-1}, [\mathbf{r}_t; \mathbf{e}_t]), t > 0 \end{aligned} \tag{1}$$

where $\mathbf{0}$ is a zero vector and $\mathbf{s}_t$ is the low-dimensional representation of state s_t.

Based on the parameterized state $\mathbf{s}_t$ and the parameterized action $\mathbf{a}$, we calculate the probability distribution over possible action space $\mathcal{A}_t$ as follows:

$$\begin{aligned} \mathbf{y}_t &= \mathbf{W}_2 ReLU(\mathbf{W}_1 \mathbf{s}_t + \mathbf{b}_1) + \mathbf{b}_2, \\ \pi_\theta(a'|s_t) &= \frac{\exp(\mathbf{a}'^\top \mathbf{y}_t)}{\sum_{a \in \mathcal{A}_t} \exp(\mathbf{a}^\top \mathbf{y}_t)}, \end{aligned} \tag{2}$$

where $\{\mathbf{W}_1, \mathbf{W}_2\}$ and $\{\mathbf{b}_1, \mathbf{b}_2\}$ are weight matrices and weight vectors of a two-layer fully-connected neural network, $ReLU(x) = max(0, x)$ is the non-linear activation function and $\pi_\theta(a'|s_t)$ is the probability of taken action a' under state s_t.

4.2.2 Reward Augmentation. According to our initial definition of rewards, the agent gets a positive reward if and only if it successfully finds the target item. However, this might be problematic for a few reasons: First, for a large user-item-entity graph $\mathcal{G}'$, it is very difficult for the agent to reach the correct items due to the huge search space, especially at the beginning of training [32]. In other words, the rewards will be very sparse. As a result, the learning process of the agent could be very inefficient and take a long time to converge. Second, the goal of recommender systems is to infer new items that users are likely to interact with in the future, rather than repeating users' historical items. However, receiving positive rewards only from historical items discourages the agent to explore new paths and items, which should be the target of recommender systems. To accelerate the training process and meanwhile encourage the agent to explore items that have not been purchased or rated by the target user, we propose to shape the rewards [17] in the following way:

$$R_T = \begin{cases} 1, & \text{if } e_T \in \mathcal{I} \text{ and } < e_0, \bar{r}, e_T > \in \mathcal{G}', \\ \sigma(\psi(e_0, \bar{r}, e_T)), & \text{if } e_T \in \mathcal{I} \text{ and } < e_0, \bar{r}, e_T > \notin \mathcal{G}', \\ -1, & \text{otherwise}, \end{cases} \tag{3}$$

where $\sigma(x) = \frac{1}{1+e^{(-x)}}$ is the sigmoid function and $\bar{r}$ represents the relation "Interact". $\psi(e_0, \bar{r}, e_T)$ is the score function that measures the correlation between user e_0 and the searched item e_T. In our study, $\psi(e_0, \bar{r}, e_T)$ is pre-trained by maximizing the likelihood of all triplets in graph $\mathcal{G}'$ and can be the score function of any state-of-the-art knowledge graph embedding models [5, 21, 33]. For example, the score function is $\psi(e_0, \bar{r}, e_T) = -\|e_0 + \bar{r} - e_T\|$ in TransE [2] model. Different from the original rewards defined in Section 4.1, items that the target user has not interacted with now receive positive rewards, which are determined by the pre-trained knowledge graph embeddings.

4.2.3 Optimization. Finally, we use the policy gradient [22] method to optimize our policy network. During training, the agent starts

with an initial state (r_0, e_0), where e_0 is the target user, and sequentially extends its path to a maximum length of T. We then use the reward function (i.e., Eq. 3) to assign the trajectory $(s_0, a_0, s_1, a_1, ..., s_T)$ a final reward. Formally, we define the expected rewards over all traversed paths of all users as:

$$J(\theta) = \mathbb{E}_{e_0 \in \mathcal{U}}[\mathbb{E}_{a_1, a_2, ..., a_T \sim \pi_\theta(a_t|s_t)}[R_T]]. \quad (4)$$

which is maximized via gradient ascent, and the gradients of all parameters θ are derived by the REINFORCE [29] algorithm, i.e.,

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t R_T \log \pi_\theta(a_t|s_t). \quad (5)$$

5 FURTHER CONSTRAINTS ON ACTIONS

The current action space $\mathcal{A}_t$ under state s_t is defined as the set of outgoing edges of current entity e_t. This could be problematic for two reasons. First, for $t < T$, if entity e_t is already the correct item (i.e., $(e_0, e_t) \in \mathcal{G}$), the agent should stop and not continue to walk to other entities. Second, since the REINFORCE algorithm tends to encourage the agent to repeat historical experiences which receive high rewards [6], the algorithm may discourage the agent from exploring new paths and items, which could be relevant to the target user. We address the two problems in the following ways:

5.1 Stop Action

As the length of paths may vary for different user-item pairs, we should provide the agent an option to automatically terminate when it believes that it has found the right items ahead of T. Following Das et al. [4]and Lin et al. [13], we add a special link from each node to itself. In this way, we allow the agent to stay at the ground truths, which can be understood as a stop action. We show the impact of using stop action in Section 6.4 by setting different path length T.

5.2 Action Dropout

To prevent the agent from repeating historical high-reward paths and encourage it to explore more possibilities, we propose to use action dropout [13] during the training. Specifically, instead of sampling an action from original $\pi_\theta(a_t|s_t)$, we use a mask upon $\pi_\theta(a_t|s_t)$ to randomly drop some actions. In addition, action dropout can also help alleviate the problem of irrelevant paths between a user and an item since these paths may be found coincidentally at the beginning of training.

6 EXPERIMENTS

In this section, we evaluate Ekar on three real-world datasets[2]. Compared to other state-of-the-art methods, our proposed approach has the following advantages:

- **Effectiveness**. Ekar significantly outperforms existing state-of-the-art KG-based recommendation methods in terms of recommendation accuracy.
- **Explainability**. Case studies on generated paths demonstrate that Ekar can offer good explanations for recommended items.

Next, we first describe the evaluation datasets and experimental set-ups.

[2]We will make the code and processed datasets public if our paper gets accepted.

6.1 Data and Experiment Settings

6.1.1 Data. We test Ekar on three benchmark datasets for KG-based recommendation:

- *Last.FM*[3]. This dataset contains a set of music artist listening information from a popular online music system Last.Fm.
- *MovieLens-1M*[4]. MovieLens-1M provides users' ratings towards thousands of movies. For these two datasets, we convert the explicit ratings into implicit feedback where each observed rating is treated as "1", and unobserved ratings are marked as "0"s. Following [25], we use Microsoft Satori to construct knowledge graphs for Last.FM and MovieLens-1M datasets respectively.
- *DBbook2014*[5]. This dataset provides users' reading history in the book domain. Its supporting knowledge graph is extracted from DBpedia.

As we focus on KG-based recommendation, we remove items that have no matching entities in the corresponding knowledge graph. The statistics of processed datasets are presented in Table 1. Following [3, 23, 28], we randomly split the interactions of each user into training, validation, and test set with ratio 6:2:2.

6.1.2 Baseline Methods. We compare Ekar with two kinds of methods: 1) Classical similarity-based methods including: **ItemKNN**, which recommends items that are most similar to target user's historical items; **BPR-MF** [34], which is a widely-used matrix factorization method using Bayesian Personalized Ranking (BPR) loss. 2) KG-based recommendation methods including: **RippleNet** [23], which propagates users' interests over knowledge graph with attention mechanism; **CFKG** [36], which learns users' and items' representations by applying TransE [2] on the graph $\mathcal{G}'$; **MKR** [25], which learns both user-item matching task and knowledge graph embedding task under multi-task learning framework; **KTUP** [3], which is a state-of-the-art KG-based recommender that jointly learns translation-based recommendation [7] and translation-based knowledge graph embedding; **ConvE-Rec**, which learns users' and items' embeddings based on integrated graph $\mathcal{G}'$ with ConvE [5]. As we use ConvE model for reward shaping, we treat ConvE-Rec as a special recommender.

6.1.3 Evaluation Metrics. Following [3, 8, 20], we adopt *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) to evaluate the effectiveness of proposed Ekar and baseline methods. We use the same definition of HR and NDCG in [8], where HR measures whether test items are present in the recommendation list and NDCG assesses the ranking quality of test items respectively. In our study, we always report the averaged HR@K and NDCG@K scores across all users over five runs.

6.1.4 Implementation Details. First of all, we add $< e_t, r^{-1}, e_h >$ into $\mathcal{G}'$ if a triplet $< e_h, r, e_t >$ exists to enhance the connectivity of graph, where r^{-1} is the inverse of relation r. Following [3], we only preserve those triplets that are directly connected to items in each supporting knowledge graph. Since the volume of action space $\mathcal{A}_t$ can be quite large for some nodes, it's time- and memory-consuming to maintain the size of action space based on the largest

[3]https://grouplens.org/datasets/hetrec-2011/
[4]https://grouplens.org/datasets/movielens/1m/
[5]http://2014.eswc-conferences.org/important-dates/call-RecSys.html

Data	User-Item Interaction				Knowledge Graph		
	# Users	# Items	# Events	Sparsity	# Entities	# Relations	# Triplets
Last.FM	1,872	3,846	21,173	99.71%	9,366	60	15,518
MovieLens-1M	6,040	2,347	656,462	95.37%	7,008	7	20,782
DBbook2014	5,576	2,598	65,445	99.55%	10,149	13	135,580

Table 1: Statistics of evaluation datasets and corresponding knowledge graphs.

Model	Last.FM		MovieLens-1M		DBbook2014	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
ItemKNN	0.0605	0.0511	0.0738	0.2273	0.0702	0.0665
BPR-MF	0.1199	0.0916	0.0895	0.1914	0.0829	0.0565
RippleNet	0.1008	0.0641	0.1269	0.2516	0.0763	0.0571
CFKG	0.1781	0.1226	0.1393	0.2512	0.1428	0.1036
MKR	0.1447	0.0850	0.1073	0.2245	0.0863	0.0575
KTUP	0.1891	0.1566	0.1579	0.3230	0.1761	0.1299
ConvE-Rec	0.2426	0.1742	0.1993	0.3676	0.1850	0.1357
Ekar	0.2201	0.1552	0.1889	0.3543	0.1716	0.1266
Ekar*	**0.2483**	**0.1766**	**0.1994**	**0.3699**	**0.1874**	**0.1371**
Gain over KTUP	31.31%	13.41%	26.28%	14.52%	6.41%	5.54%

Table 2: Performance of different models on three datasets. The gain of Ekar* over KTUP is statistically significant at the 0.001 level according to t-test.

action space for all states. To simply implementation and computation, we only preserve a fixed number (256 in our experiments) of "important" outgoing edges for each node, and the importance of edges are decided by the PageRank scores [18] of their end nodes. For nodes that have less than 256 outgoing edges, we up-sampling their outgoing edges to 256.

We implement Ekar with Pytorch [19]. Entity and relation embeddings are pre-trained by applying ConvE [5] on graph $\mathcal{G}'$, and the embedding size is set to 32 for all methods except for ItemKNN, which has no latent representations. Meanwhile, we use the score function of ConvE to compute augmented rewards (i.e., Equation 3). From Figure 1, we can see path patterns "User->Item->Entity->item" and "User->Item->User->Item" are more probable to be meaningful, so we empirically set the maximum path length T to 3 as our default setting. We select action dropout rate from {0.1-0.9}, dropout rate for entity/relation embeddings from {0.1-0.9} using grid search. Meanwhile, grid search is also applied to select the optimal hyperparameters for other baseline methods based on the performance on validation data. For training, we use Adam [11] optimizer for all models with batch size of 512. For recommendation, we use beam search with beam size 64 to generate paths for target users. For duplicate paths leading to the same item, we keep the one with the highest probability. Finally, we adopt two different ranking strategies to generate final top-K recommendation list: (1) ranks the searched items according to the path probabilities and we denote it as Ekar, 2) ranks the searched items based on "rewards" defined by $\sigma(\psi(e_0, e_T))$ in Equation 3 and we denote it as Ekar*.

6.2 Analysis of Recommendation Performance

We report the recommendation accuracy of different methods in Table 2. We can see that KG-based recommendation methods consistently outperform classical similarity-based methods, which indicates that knowledge graphs indeed help to alleviate the problem of data sparsity in the recommendation. Among KG-based recommendation methods, the RippleNet performs worst, which may be attributed to representing users with multi-hop away entities. KTUP performs strongly because it takes the advantages of both translation-based recommendation and multi-task learning. Note that the only difference between ConvE-Rec and CFKG is the used knowledge graph embedding methods; however, ConvE-Rec achieves much better performance. The reason behind this is that ConvE is a state-of-the-art knowledge graph embedding method, which outperforms TransE used in CFKG. By using pre-trained ConvE embeddings to augment rewards, our Ekar and Ekar* significantly outperform existing state-of-the-art KG-based recommendation methods in most cases and perform comparably to the unexplainable ConvE-Rec model, which shows that our proposed methods are quite effective.

6.3 Analysis of Explainability

After demonstrating the effectiveness of Ekar, we now illustrate its explainability, which is the main contribution of this work to recommender systems. As introduced in previous sections, Ekar provides recommendations by generating meaningful paths from users to items, where paths serve as explanations for recommended items. To give you an intuitive example, we randomly select a real

Model	Last.FM		MovieLens-1M		DBbook2014	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Ekar	**0.2201**	**0.1552**	**0.1889**	**0.3543**	**0.1716**	**0.1266**
Ekar-KG	0.2061	0.1466	0.1869	0.3489	0.0802	0.0525
Ekar-RS	0.0614	0.0349	0.0654	0.1132	0.1174	0.0867
Ekar-AD	0.1350	0.0827	0.1715	0.3217	0.1449	0.1083
Ekar(T=5)	0.2108	0.1505	0.1859	0.3500	0.1524	0.1125

Table 3: Performance w.r.t. model variants, where [-] means removing that component from the Ekar.

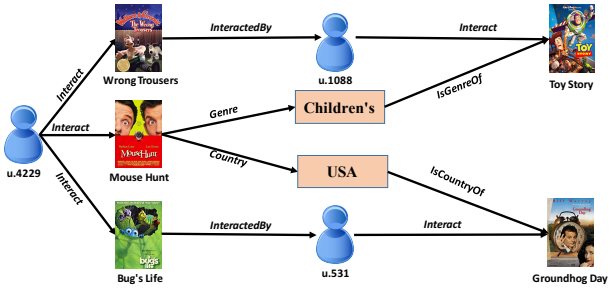

Figure 3: Recommendations and explanations for user #4229 in MovieLens-1M dataset. "Toy Story" is recommended because 1) it shares the same movie genre(i.e., Children's) with "Mouse Hount", and 2) it is watched by another user who also watched "Wrong Trousers". These explainable paths are automatically discovered by Ekar in a generative manner.

Explainable preference paths	Pct. (%)
$\text{U} \xrightarrow{Int.} \text{M} \xrightarrow{Int.^{-1}} \text{U} \xrightarrow{Int.} \text{M}$	78.6
$\text{U} \xrightarrow{Int.} \text{M} \xrightarrow{Country} \text{C} \xrightarrow{IsCountryOf} \text{M}$	15.4
$\text{U} \xrightarrow{Int.} \text{M} \xrightarrow{Genre} \text{G} \xrightarrow{IsGenreOf} \text{M}$	2.3
$\text{U} \xrightarrow{Int.} \text{B} \xrightarrow{Type} \text{T} \xrightarrow{IsTypeOf} \text{B}$	64.9
$\text{U} \xrightarrow{Int.} \text{B} \xrightarrow{LinkedTo} \text{WP} \xrightarrow{Link} \text{B}$	21.7
$\text{U} \xrightarrow{Int.} \text{B} \xrightarrow{Int.^{-1}} \text{U} \xrightarrow{Int.} \text{B}$	7.4

Table 4: Most frequent path patterns of Ekar during inference on MovieLens-1M (top) and DBbook2014 (bottom) datasets. "U", "M", "C", "G", "B", "T" and "WP" represent User, Movie, Country, Genre, Book, Type and WikiPage respectively. "$Int.$" and "$Int.^{-1}$" are "Interact" and "Interacted^{-1}" in short respectively.

user from MovieLens-1M dataset and search preference paths for them with Ekar. As shown in Figure 3, we can easily understand that "Toy Story" is recommended because it shares the same genre (i.e., Children's) with "Mouse Hunt", which the user watched before. What'more, we find that Ekar model tends to provide multiple explanations for the recommendations. For example, movie "Toy Story" is also liked by another user (#1088) who watched "Wrong Trousers". Such diverse explanations may better motivate users to accept the corresponding recommendations.

Beyond explanations for an individual user, we are also interested in global preference path patterns discovered by Ekar. More specifically, we try to figure out what are the typical path patterns w.r.t. different datasets. From Table 4, we can see that Ekar relies more on the path pattern "$\text{User} \xrightarrow{Interact} \text{Movie} \xrightarrow{Interact^{-1}} \text{User} \xrightarrow{Interact} \text{Movie}$" on MovieLens-1M dataset. Interestingly, we find that Ekar learns relatively diverse path patterns on DBbook2014 dataset such as "$\text{User} \xrightarrow{Interact} \text{Book} \xrightarrow{LinkedTo} \text{WikiPage} \xrightarrow{Link} \text{Book}$" and "$\text{User} \xrightarrow{Interact} \text{Book} \xrightarrow{Type} \text{Type} \xrightarrow{IsTypeOf} \text{book}$", which lead to new books that share the same WikiPage or Type with users' historical books respectively. The reason behind the discrepancy of path patterns on two datasets may be two folds. First, note that the average number of interactions for each user in MovieLens-1M data is about 100 while this number is 12 in DBbook2014 data. Therefore it is easier to find a user sharing similar movie preference in MovieLens-1M data than to find a user with similar reading taste in DBbook2014 data. Second, the size of supporting knowledge graph for DBbook2014 data is much larger than the number of user-item interactions, so our Ekar learns to make more use of external knowledge when the user-item interactions are sparse.

6.4 Ablation Study

In this section, we compare different variants of Ekar to show the influences of some essential components such as KG, reward shaping, action dropout and maximum path length T, which are denoted as Ekar-KG, Ekar-RS, Ekar-AD and Ekar(T=5) respectively in Table 3. We find the influence of KG is very significant on Last.FM and DBbook2014 datasets. This is because these two datasets are extremely sparse, while the MovieLens-1M data is relatively dense. Removing reward shaping leads to a severe performance drop on all datasets because Ekar without reward shaping assigns zero rewards to all items that a user has not interacted with. In this way, the agent is penalized for exploring potential items that are of interest to users and therefore cannot effectively generate recommendations. Besides reward shaping, we also use action dropout to further encourage the agent to explore diverse paths, and we can see that Ekar-AD performs worse than the full model Ekar. At last, we try a larger maximum path length T to enable the agent to explore longer paths. We find that Ekar(T=5) performs worse than Ekar on all datasets. This is because long paths may introduce more noise and thus be less meaningful. However, thanks to the stop mechanism, the performance drop is not significant.

Explainable preference paths	Pct. (%)
$U \xrightarrow{Int.} M \xrightarrow{Int.^{-1}} U \xrightarrow{Int.} M \xrightarrow{Int.^{-1}} U \xrightarrow{Int.} M$	58.7
$U \xrightarrow{Int.} M \xrightarrow{Int.^{-1}} U \xrightarrow{Int.} M \xrightarrow{Country} C \xrightarrow{IsCountryOf} M$	25.7
$U \xrightarrow{Int.} B \xrightarrow{Int.^{-1}} U \xrightarrow{Int.} B \xrightarrow{Type} T \xrightarrow{IsTypeOf} B$	52.3
$U \xrightarrow{Stop} U \xrightarrow{Int.} B \xrightarrow{Type} T \xrightarrow{IsTypeOf} B \xrightarrow{Stop} B$	13.2
$U \xrightarrow{Stop} U \xrightarrow{Int.} B \xrightarrow{Type} T \xrightarrow{IsTypeOf} B \xrightarrow{Link} B$	9.9

Table 5: Most frequent path patterns of Ekar(T=5) during inference on MovieLens-1M (top) and Last.FM (bottom) datasets. "U", "M", "C", "B" and "T" represent User, Movie, Country, Book and Type respectively. "$Int.$" and "$Int.^{-1}$" are "Interact" and "Interacted^{-1}" in short respectively.

Model	Score function $\psi(e_0, e_T)$
DistMult	$\langle \mathbf{e}_0, \mathbf{r}, \mathbf{e}_T \rangle$
ConvE	$g(\text{vec}(g([\overline{\mathbf{e}_0}; \overline{\mathbf{r}}] * \omega))\mathbf{W})\mathbf{e}_T$

Table 6: Score functions w.r.t. different KGE methods, where $\langle \cdot \rangle$ denotes generalized inner product of three vectors, $\bar{\cdot}$ denotes a 2D shaping of vectors, $*$ is the convolution operator, ω denotes filters in convolutional layers, $g(\cdot)$ is a non-linear activation function and vec$(\cdot)$ converts a tensor to a vector. $\mathbf{e}_0$, $\mathbf{e}_T$ and $\mathbf{r}$ are embeddings of user e_0, entity e_T and relation "Interact" respectively.

To better understand the role of stop action, we present the details about Ekar(T=5)'s paths patterns in Table 5. We have two observations. First, there are many paths of length five, which means that our agent is recommending items that have high-order similarity to users' historical items. Second, since the supporting knowledge graph is very large on DBbook2014 dataset and paths with length five may not be always useful, our agent learns to infer path with length of three or four by taking stop actions.

6.5 Ekar with Different KGE Methods

Although Ekar model has been utilizing ConvE model so far to pre-train entity and relation embeddings for both initialization and reward shaping, it is notable that Ekar is independent of any specific knowledge graph embedding methods. Therefore we also test our model with another widely-used knowledge graph embedding method DistMult [33]. The score functions of DistMult and ConvE are presented in Table 6. For a fair comparison, we use the same experimental settings and just substitute DistMult for ConvE in our experiments. Following ConvE-Rec, we denote recommendation with DistMult as DistMult-Rec.

The results of using different knowledge graph embedding methods on MovieLens-1M and DBbook2014 are presented in Table 7. First, we can see that ConvE-Rec outperforms DistMult-Rec on these two datasets. This is because ConvE has proven more effective than DistMult for knowledge graph completion, as a result

Model	MovieLens-1M		DBbook2014	
	HR@10	NDCG@10	HR@10	NDCG@10
ConvE-Rec	0.1993	0.3676	0.1850	0.1357
DistMult-Rec	0.1773	0.3341	0.1535	0.1090
Ekar (ConvE)	0.1889	0.3543	0.1716	0.1266
Ekar (DistMult)	0.1761	0.3352	0.1367	0.0958
Ekar* (ConvE)	0.1994	0.3699	0.1874	0.1371
Ekar* (DistMult)	0.1774	0.3343	0.1482	0.1061

Table 7: Effectiveness comparison of using different knowledge graph methods for entity/relation initialization and reward shaping. The trends on Last.FM dataset are similar hence omited.

of which our Ekar with ConvE also outperforms Ekar with DistMult. Second, Ekar (DistMult) and Ekar* (DistMult) outperform DistMult-Rec on MovieLens-1M dataset and perform comparably to DistMult-Rec on DBbook2014 dataset, which is consistent with the observations of Ekar (ConvE) and Ekar* (ConvE). We omit the results on Last.FM dataset because they show similar trends and lead to the same conclusion.

6.6 Convergence Analysis

We present the running time of Ekar in Figure 4. As can be seen, Ekar converges fast on MovieLens-1M dataset with less than ten minutes, while it takes a bit more time to converge on the other two datasets. The reason for different convergence behaviors is that it is easier to walk to correct items on dense datasets (e.g., MovieLens-1M) than on sparse datasets (e.g., DBbook2014). Overall, our Ekar is efficient because we initialize entity/relation embeddings with pre-trained knowledge graph embeddings, and we use reward shaping to augment reward signals.

7 CONCLUSION

In this paper, we introduced a novel approach to provide explanations for recommendation with knowledge graphs. Our proposed Ekar generates meaningful paths from users to relevant items by learning a walk policy on the user-item-entity graph. Experimental results show that Ekar outperforms existing KG-based recommendation methods and is quite efficient. Furthermore, we demonstrate the explainability of Ekar via insightful case studies on different datasets. Future work includes incorporating domain knowledge to design proper reward functions for recommendation task and developing a distributed version of Ekar for even larger datasets.

REFERENCES

[1] 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 (KDD '17)*. ACM, New York, NY, USA, 717–725. https://doi.org/10.1145/3097983.3098170

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[3] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better

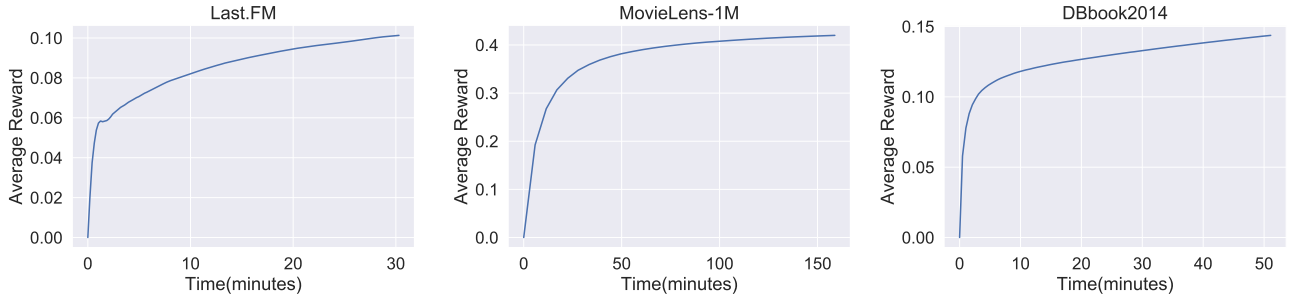

Figure 4: Results of running time with batch size of 512 and maximum path length of 3. The x-axis is the training time in minutes, and the y-axis is the average rewards over training samples.

Understanding of User Preferences. In *The World Wide Web Conference (WWW '19)*. ACM, New York, NY, USA, 151–161. https://doi.org/10.1145/3308558.3313705

[4] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

[5] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

[6] Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From Language to Programs: Bridging Reinforcement Learning and Maximum Marginal Likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1051–1062.

[7] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 161–169. https://doi.org/10.1145/3109859.3109882

[8] 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 (CIKM '15)*. ACM, New York, NY, USA, 1661–1670. https://doi.org/10.1145/2806416.2806504

[9] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00)*. ACM, New York, NY, USA, 241–250. https://doi.org/10.1145/358916.358995

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[12] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. ACM, New York, NY, USA, 1419–1428. https://doi.org/10.1145/3132847.3132926

[13] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3243–3253.

[14] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I Like It: Multi-task Learning for Recommendation and Explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, New York, NY, USA, 4–12. https://doi.org/10.1145/3240323.3240365

[15] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *The World Wide Web Conference (WWW '19)*. ACM, New York, NY, USA, 1210–1221. https://doi.org/10.1145/3308558.3313607

[16] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 165–172. https://doi.org/10.1145/2507157.2507163

[17] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The PageRank citation ranking: Bringing order to the Web. In *Proceedings of the 7th International World Wide Web Conference*. Brisbane, Australia, 161–172. citeseer.nj.nec.com/page98pagerank.html

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[20] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. ACM, New York, NY, USA, 555–563. https://doi.org/10.1145/3289600.3290989

[21] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

[22] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[23] 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 (CIKM '18)*. ACM, New York, NY, USA, 417–426. https://doi.org/10.1145/3269206.3271739

[24] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1835–1844. https://doi.org/10.1145/3178876.3186175

[25] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *The World Wide Web Conference (WWW '19)*. ACM, New York, NY, USA, 2000–2010. https://doi.org/10.1145/3308558.3313411

[26] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. ACM, New York, NY, USA, 165–174. https://doi.org/10.1145/3209978.3210010

[27] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*. Association for Computing Machinery, New York, NY, USA, 950–958. https://doi.org/10.1145/3292500.3330989

[28] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

[29] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[30] Yao Wu and Martin Ester. 2015. FLAME: A Probabilistic Model Combining Aspect Based Opinion Mining and Collaborative Filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. ACM, New York, NY, USA, 199–208. https://doi.org/10.1145/2684822.2685291

[31] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. ACM, New York, NY, USA, 285–294. https://doi.org/10.1145/3331184.3331203

[32] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 564–573.

[33] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

[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 (WSDM '14)*. ACM, New York, NY, USA, 283–292. https://doi.org/10.1145/2556195.2556259

[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 (KDD '16)*. ACM, New York, NY, USA, 353–362. https://doi.org/10.1145/2939672.2939673

[36] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. In *Proceedings of the 41th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. ACM, New York, NY, USA. https://doi.org/10.3390/a11090137

[37] Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192* (2018).

[38] 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 (SIGIR '14)*. ACM, New York, NY, USA, 83–92. https://doi.org/10.1145/2600428.2609579

[39] 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 (KDD '17)*. ACM, New York, NY, USA, 635–644. https://doi.org/10.1145/3097983.3098063

[40] Kaiqi Zhao, Gao Cong, Quan Yuan, and Kenny Q Zhu. 2015. SAR: A sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 675–686. https://doi.org/10.1109/ICDE.2015.7113324