Fairness-Aware Knowledge Graph Based Recommendation via Reinforcement Learning

Haolun Wu*
ID: 260898626
School of Computer Science
McGill University
Montreal, QC H3A 0G4
haolun.wu@mail.mcgill.ca

Amir Abushanab*
ID: 260624419
School of Computer Science
McGill University
Montreal, QC H3A 0G4
amir.abushanab@mail.mcgill.ca

Abstract

Personalized Recommender Systems have emerged as the tools for combating information overload and satisfying diverse user needs. They play an increasingly important role in our everyday lives, and yet previous research on recommender systems has mainly focused on optimizing end-user satisfaction without taking into account the various other stakeholders impacted by those recommendations. Also, they are regarded to lack of interpretability, since their recommendation is just calculating a similarity score based on the latent representation vectors between users and items. Our goal of this work is to create a system that will allow for (1) fairness-aware on multi-stakeholder and (2) high explainable recommendation with good quality. Specifically, we propose a model, *Skywalker*, in the setting of a two-sided marketplace such as Amazon E-commerce, where we simultaneously optimize for user and producer satisfaction while still taking into account fairness metrics derived from the brand popularity of each item. To that end, we formulate the recommendation problem as a Markov Decision Process (MDP) and propose a novel technique, demonstrating how utilizing *Multi-agent Policy Guided Reinforcement Learning in a Knowledge Graph Environment* can produce a set of recommendation that is optimal for both users and producers. Our framework is flexible and can be generalized to multiple stakeholders.

1 Introduction

With the ever-growing volume of online information, recommender system (RS) has been an effective strategy to overcome such information overload by providing users with suggestions on what they may like to view or purchase. Recommender system has been playing a vital and indispensable role in various information access systems to boost business and facilitate decision-making process and it is also an essential tool from E-commerce activities, information seeking on media to disease diagnosis [6].

In community, many approaches have been proposed to enhance the quality of recommendation, one of the notable state-of-the-art methods is to use knowledge graph (KG) in recommender systems. The powerfulness of KG-based recommender systems lies into the ability of incorporating rich side information and relations between entities, thus enriching the representation vectors of users and items in the context of a recommendation task. Also, obviously, it can alleviate the problem of data sparsity which is a very common issue in collaborative filtering.

However, one of the key problems of the pure KG-based models is that they lack of ideal interpretability. KG-based recommendation methods can be divided into embedding-based KG and path-based KG. In typical embedding-based KG [38], embeddings obtained from the knowledge base are directly used to enrich the latent representations of users and items, followed by a typical matrix factorization on the user-item interaction matrix. Such recommendation primarily uses a scoring function to match the most relevant items to a given user, which is not interpretable, since there is no path generating on the knowledge graph during recommendation, and the user cannot know the exact reason been recommended to an item. Even though Ai *et al.* [1] acclaims they find explainable paths between users and items, this kind of explanation is post-hoc: the reasoning of recommendation is not based on the paths found afterwards. As for path-based KG [32], meta-paths are first hand defined and the recommendation is based on these paths. Even though the meta-paths themselves can represent clear interpretability, the drawback of path-based KG is that it is impossible to enumerate all reasonable paths when the graph is very huge, which is always the real case.

In addition to the aforementioned limitation on interpretability, we still consider another direction which is not investigated enough in community, which is the fairness in recommender systems. Previous works on fairness mostly focus on the user side [36, 3], aiming to force the recommendation to be invariant to different user communities. However, the user is not the only stakeholder in a recommendation context, producers and supplier are at least of the same significance in the marketplace. Therefore, to propose a more reasonable fairness metric considering multi-stakeholder in recommender systems is a necessary and insightful research direction.

In this paper, we investigate interpretability and fairness-aware in recommender systems, aiming to solve the two aforementioned issues. To the best of our knowledge, there only exist two works [29, 33] combining KG-based recommendation with reinforcement learning to achieve the explainable recommendation. As for the fairness on two-side, we only notice Mehrotra *et al.* [21] proposed a primary definition based on a music recommendation platform. Based on their works, we formally formulate the recommendation problem as a Markov Decision Process (MDP) and propose a model named *Skywalker*, implementing a novel technique of *multi-agent policy guided reinforcement learning in a knowledge graph environment*, which improves the performance on traditional recommendation evaluation metrics (e.g., nDCG, Precision, Recall, Hit Ratio) greatly. We define our new fairness metrics based on the brands popularity of items in Amazon E-commerce datasets, also considering the ranking position in the recommendation. We observe the trade-off between the fairness on two-side and the recommendation quality evaluations, and highlight a model which can be regarded as achieving a good balance between these metrics. Finally, by displaying the global reasoning paths and investigating specific user study cases, we strongly approve the interpretability of our proposed method.

2 Related Work

2.1 Collaborative Filtering

Collaborative Filtering methods have been one of the most popular approaches on recommender systems these days. They are based solely on the past interactions recorded between users and items in order to produce new recommendations. These interactions are stored in a user-item interaction matrix M, where each row i represents a user, each column j represents a certain item, while each entry M_{ij} represents the rating value between user i and item j [28].

Early approaches on collaborative filtering can be regarded as memory based, which directly works with values of recorded interactions, assuming no model, and are essentially based on nearest neighbours search via user-based [15, 25] or item-based [19, 27]. Later on, model based approaches have developed and been widely applied in recommender systems, assuming an underlying model should be able to explain the user-item interactions. Notable works on this include using different variants of matrix factorization such as singular value decomposition (SVD) [41, 20], probabilistic matrix factorization (PMF) [5, 4, 35], non-negative matrix approximation (NMF) [16], etc. These approaches all aim to learn a latent representation vector for each user and item, followed by calculating a similarity score between each user-item pair, then making recommendation based on the ranking of relevant scores. Nevertheless, recent studies on interpretability of recommender systems notice that it is of difficulty to make a clear and precise explanation for the results of collaborative filtering methods, since the recommendation is implemented in a latent space.

2.2 KG-based Recommendation

The advantage of incorporating Knowledge Graph (KG) into recommender systems have been noticed by community: it can help leverage side information and connectivity between entities, thus enhancing the quality of recommendation by considering more information than the traditional user-item interaction matrix. Previous work on KG-based recommendation methods can be divided into two categories: embedding-based methods and path-based methods.

In embedding-based methods, embeddings trained on the knowledge graph are directly enriched into the original representationss obtained from the user-item interaction matrix, followed by a typical matrix factorization method. For instance, Zhang *et al.* [38] proposed a Collaborative Knowledge Base Embedding (CKE) model to enrich the item embeddings by leveraging *structural*, *visual*, and *textual* information from an item knowledge base. Wang *et al.* [32] employed a deep knowledge-aware network (DKN) that incorporates knowledge graph representation to obtain side information into the task of news recommendation. They are all able to achieve a recommendation of good precision and recall but lack of clear interpretability: it is hard to find reasonable explanation path due to the latent representation space; even though some work acclaims reasonable explanations, they are totally post-hoc, since the recommendation process is not based on the reasoning, but on the ranking of similarity score. The path-based methods solve the aforementioned issue on interpretability by first defining some meta-paths or meta-graphs (a combination of meta-paths) and then doing recommendation based on these predefined paths [39, 37]. The explanation paths are absolutely clear, which are the meta-paths themselves; however, the limitation lies in the computational infeasibility for hand-defining all the possible paths when the knowledge graph is very large, which is always the case in reality.

Therefore, we regard the pure KG-based approaches on recommender systems as good techniques for improving quality of recommendation with respect to traditional evaluation metrics, such as precision and recall, but still lack of ideal interpretability.

2.3 Fairness in Recommender Systems

Fairness is always an important pursuit in human society and is also been investigated in community. Most previous works on fairness in recommender systems merely focus on one-side – the user side, which is also known as *Group Fairness* or *Demographic Parity*. For instance, Bose *et al.* [3] employed a flexible adversarial framework in social recommendation, allowing users to request their recommendations being invariant to sensitive features, e.g., age, gender, and religion. Yao *et al.* [36] proposed four new metrics that address different forms of unfairness among the group of users, such as males and females, then aimed to add these terms in the loss function to minimize them.

However, "users" is not the only community in recommender systems, there still exist other stakeholders behind "items" which are also human beings, such as the producers and suppliers. In reality, recommender systems always suffer from an inherent problem of "superstar economics" [26]: a small number of suppliers may receive the highest relevance score and popularity among most users. Therefore, those less popular producers and new coming suppliers do not have a fair opportunity of exposure, even though their products may have a high quality. Thus, a more general fairness in recommender systems defined over two sides is required.

To the best of our knowledge, the only paper investigates the fairness on a two-side marketplace is by Mehrotra *et al.* [21]. In our project, we extend their definition of fairness and propose our own fairness metrics.

2.4 Reasoning with Reinforcement Learning

Reinforcement learning (RL) has been investigated a lot in community, it is of good ability to understand the environment, thus intending to achieve the ideal behavior of a model within a complex context. RL techniques are also implemented in some recommendation tasks, such as news recommendation [40] and personalized ad recommendation [31], to optimize recommendations for long-term user engagement. Recently, some works use the RL idea with knowledge graph reasoning to do path-finding for Question Answering (QA) tasks and achieve a good performance [7, 17, 34]. However, as for the KG-based recommender systems, to the best of our knowledge, there only exist two works on this [33, 29] by using RL to do path finding and recommendation simultaneously, thus enhancing the interpretability of recommendation.

Our proposed model *Skywalker* is based on the techniques introduced by these two works, with the key differences being the multi-agent setting and fairness metrics we introduced.

3 Methodologies

We frame the recommendation problem as a multi-step reinforcement learning problem where multiple agents perform walks in a KG environment, guided by a learned optimal policy for their respective task. The entities and relations in the traversed paths themselves form the explanation reasoning behind the recommendations, with a clear and not post-hoc interpretability.

3.1 Problem Definition

We first formulate the traditional user-item interaction matrix as a bipartite graph as

$$G = \{(u, i) | u \in U, i \in I\}, \tag{1}$$

where U and I represent the user set and item set, respectively. Also, we have access to a knowledge graph defined over the entity set E and relation set R as

$$G_K = \{(e, r, e') | e, e' \in E, r \in R\}, \tag{2}$$

where each triplet indicates a fact that head entity e can reach tail entity e' by going through relation r. The relationship between G and G_K is that $U, I \subset E$. Finally, we combine the two graphs together and add a self-loop operation on all entities to allow the agent to remain stationary:

$$G' = \{(e, r, e') | e, e' \in E, r \in R'\}, \tag{3}$$

where $R' = R \cup \{Interaction(U, I)\} \cup \{selfloop(U)\}$.

Thereafter, the recommendation problem is defined as: given an initial user $u \in U$, our goal is to generate a path from u to a relevant item $i \in I$ on the knowledge graph G'. Such a path not only allows to find the relevant item but also offers good explanations.

3.2 Formulation as Markov Decision Process

Due to the aforementioned drawbacks in pure KG-based recommendation approaches, we introduce reinforcement learning into our recommendation problem in this project, aiming to let the agent do path reasoning and recommendation simultaneously, thus improving the interpretability of recommendation. We formally formulate our problem into a Markov Decision Process (MDP), whose components are defined as follows:

State. The state at step t is defined as the sequence of past traversed entities and relations: $s_t = (r_0, e_0, r_1, e_1, ..., r_t, e_t)$, where $r_i \in R$ and $e_i \in E$. Specifically, e_0 denotes the initial user and r_0 represents the self-loop on e_0. Therefore, the initial state is of the same formation, i.e., $s_0 = (r_0, e_0)$. And the terminal state is formulated as $s_T = (r_0, e_0, r_1, e_1, ..., r_T, e_T)$.

Action Space. The action space represents all the agent's choices for the next one-hop outgoing edge when it is in a state s_t. The formulation is denoted as: $A_t = \{a_t = (r', e') | (e_t, r', e') \in G', e \notin \{e_0, e_1, ..., e_{t-1}\}\}$ is the terminal tail entity in state s_t.

Since the size of each action space is based on the out-degree of each entity node and some out-degree can be very large, it requires a large spacial and computational resource to maintain and traverse each possible action. We thus implement an *action pruning strategy* to merely keep the actions which lead to most relevant items to the initial user, and we define the pruned action space conditioned on user u as:

$$\hat{A}_t(u) = \{a_t = (r, e) | \text{rank}(f_a(r, e) | u) \leq \alpha, (r, e) \in A_t\}, \tag{4}$$

where α is a hyperparameter which denotes the upper bound of the size of the action space, f_a is a scoring function defined on a dot product operation, i.e., $f_a = \langle \mathbf{e}_0 + \sum_{i=0}^{t} \mathbf{r}_t, \mathbf{e}_t \rangle$, where t is the current step. Additionally, in order to encourage the diversity of the paths, we still adopt dropout strategy on the pruned action space $\hat{A}_t(u)$.

Transition. One significant property of knowledge graph is that: the one-hop tail entity is determined when given the head entity and the one-hop relation passing through. When the agent in state s_t takes an action a_t from the action space, the next state s_{t+1} is obviously determined by simply concatenating the action into the previous state, i.e., $s_{t+1} = s_t \oplus a_t = (r_0, e_0, r_1, e_1, ..., r_t, e_t, r_{t+1}, e_{t+1})$, where "$\oplus$" denotes the concatenation operation. Therefore, we adopt a deterministic transition in our MDP as: $\mathcal{P}(s_{t+1}|s_t = (r_0, e_0, r_1, e_1, ..., r_t, e_t), a_t = (r_{t+1}, e_{t+1})) = 1$.

Reward. One challenge for the reward design in this recommendation problem is that there is no ground truth for the "correct" recommended items given an initial user. Therefore, we cannot force a binary reward to indicate whether the user is interacted with the recommended items or not. Instead, we define a soft reward only for the terminal state s_T based on a scoring function f_r, which is defined as the dot product of the initial user embedding and the terminal entity embedding, i.e., $f_r = \langle \mathbf{e}_0, \mathbf{e}_T \rangle$. Then the terminal reward R_T is defined as:

$$R_T = \begin{cases} max(0, \frac{f_r(u, e_T)}{max_{i \in I} f_r(u,i)}), & \text{if } e_T \in I, \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

where e_T is the tail entity in the terminal state. Also notice that, a reward will only be offered when the terminal entity $e_T \in I$ rather than any entities in E on the knowledge graph, since our aim is to recommend items to users.

3.3 Optimization with Policy Gradient

The goal of the recommendation MDP formulation is to learn a policy which can tell the agent how to do the path "walking" on the knowledge graph; thereafter, recommendation and path reasoning can be achieved simultaneously. We use the policy gradient approach to solve the proposed recommendation in this project.

Mathematically, the policy π_θ denotes the probability of each action a' given the current state s_t and the current pruned action space $\hat{A}_t$. We first obtain the state vector $\mathbf{s}_t$ by concatenating each representation of traversed entities and relations in state s_t, then define the policy π_θ as follows:

$$\pi_\theta(a'|s_t, \hat{A}_t) = \frac{exp(\mathbf{a}'^T \mathbf{x}_t)}{\sum_{a \in \hat{A}_t} exp(\mathbf{a}^T \mathbf{x}_t)}, \tag{6}$$

where $\mathbf{a}$ denotes the action vector, $\mathbf{x}_t$ represents the learned hidden features of the state s_t, which is calculated as $\mathbf{x}_t = W_2 ReLU(W_1 \mathbf{s}_t + b_1) + b_2$. It is obvious to notice that the aforementioned θ contains trainable parameters: $\{W_1, W_2, b_1, b_2\}$.

The training process is followed by to maximize the expected discounted cumulative reward, for which the objective function is defined as:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T-1} \gamma^t R_{t+1} | s_0 = (r_0, e_0)], \tag{7}$$

where T is the number of the maximum hop, γ is the discounted value, R is the reward function defined before. Finally, the optimization is to maximize $J(\theta)$ via gradient ascent, while the policy gradient is derived by the basic REINFORCE [30] algorithm as:

$$\bigtriangledown_\theta J(\theta) = E_{\pi_\theta}[\bigtriangledown_\theta \text{log}\pi_\theta(a'|s_t, \hat{A}_t) \cdot G], \tag{8}$$

where G is the discounted cumulative reward from state s to the terminal state s_T. Since we only define the reward on the final state, the G can also be replaced with R_T.

3.4 Multi-agent Strategy: *Skywalker*

We introduce a multi-agent strategy, named *Skywalker*. In our project, we train two agents independently using the same approach described before, leveraging the embeddings obtained from a *TransE* [2] model. The description for these two agents are as follows:

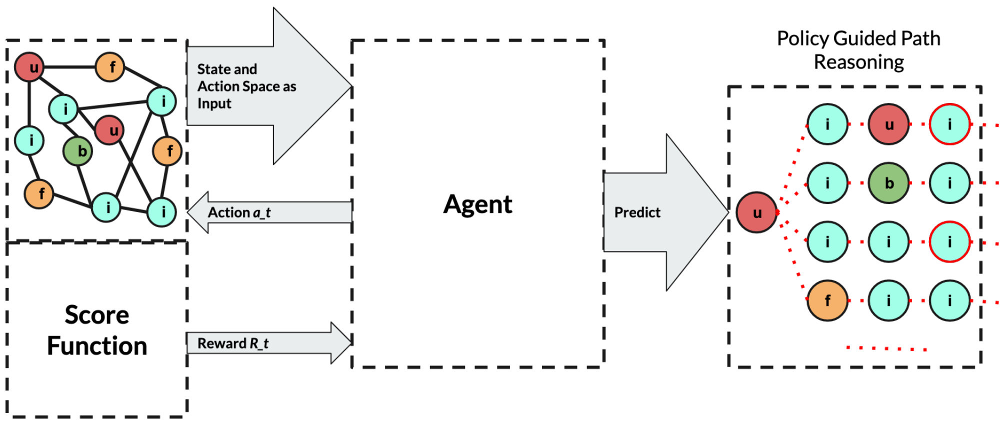

Figure 1: The training and prediction procedure for Agent-U. The training process aims to learn a policy to tell the agent how to "walk" on the knowledge graph. The prediction process is to use the policy learned before to make recommendation and generate interpretability simultaneously given an initial user, with beam search strategy.

- **Agent**$_U$: aims to recommend items to users. The output is a list of top-k recommended items given an initial user. The whole pipeline of agent training process and prediction process based on the idea aforementioned is illustrated in Fig. 1.
- **Agent**$_I$: aims to recommend users to items. In this project, we sets our Agent$_I$ to do k separate walks. Therefore, the output is a dictionary: each "key" represents an item id from the top-k recommendation of Agent$_U$, the corresponding "value" is a list of recommended users given the item in the "key".

Intuitively, *Skywalker* works by finding a set of recommendations that is favorable to multiple agents, while still prioritizing the main agent's objective - in this case Agent$_U$, which is optimizing for end-user satisfaction. The high level idea is shown in Fig. 2. The process is detailed as follows: given an initial user e_0, the Agent$_U$ will result a top-k recommendation: I_1, I_2, ..., I_k. For each item in the output of Agent$_U$, Agent$_I$ will do a similar "walk" on the knowledge graph to recommend a list of top-k most relevant users. Each item I_j $(j \leq k)$ will be added into the final recommendation of e_0 if e_0 occurs in Agent$_I$'s recommendation on I_j. If we traverse all the items and the final recommendation has not reached the required length of k, the rest of the recommendations will be padded based on the order they appear in the main agent's recommendations.

It is significant to notice that, if we only keep the top-k recommendation of Agent$_U$ and our goal is also to recommend k items, the final output will just be a permutation of Agent$_U$'s output by using *Skywalker*. Therefore, in order to introduce some new recommendations, we expand the size of Agent$_U$'s recommendation from k to $k \times \beta$ in practice, where β is a hyperparameter.

Additionally, one good property of *Skywalker* framework is that it allows considering multi-stakeholders in recommender systems by introducing multiple agents, even more than two. In our project, due to the information extracted from the Amazon E-commerce datasets, we've only obtained the users information and the brands information (indicating the suppliers), thus only introducing two agents in our model. The generalized version of *Skywalker* framework is described in **Algorithm 1**.

3.5 Fairness Metrics

Instead of focusing on the community fairness merely on the user side, we define two variations of fairness metrics in the context of a two-sided marketplace. Our aim is to consider fairness of multi-stakeholders in recommender systems.

Fairness@k. The first fairness metric is similar to the definition in Mehrotra *et al.*'s work [22]. We first split the set of all items into k equal volume groups $P_1, P_2, ..., P_k$ based on the purchased times,

Algorithm 1: Skywalker: Generalize to Multiple Stakeholders (Agents)

```
Result: A list of recommendations $S$
$k \leftarrow$ predefined final recommendation size
$S \leftarrow$ [null]
$e \leftarrow$ entity queried
$A \leftarrow$ [Agent$_1$, Agent$_2$, ...]
$M \leftarrow$ Agent$_0$.generateRecommendations($e$) $\times\beta$
foreach $m \in M$ do
    foreach $Agent \in A$ do
        $V \leftarrow Agent$.generateRecommendations($m$)
        if $e \in V$ then
            $S$.append($m$)
        end
    end
end
if $|S| \geq k$ then
    $S \leftarrow S[: k]$
end
while $|S| < k$ do
    $m \leftarrow$ None
    $m \leftarrow M$.nextItem()
    if $m \notin S$ then
        $S$.append($m$)
    end
end
return $S$
```

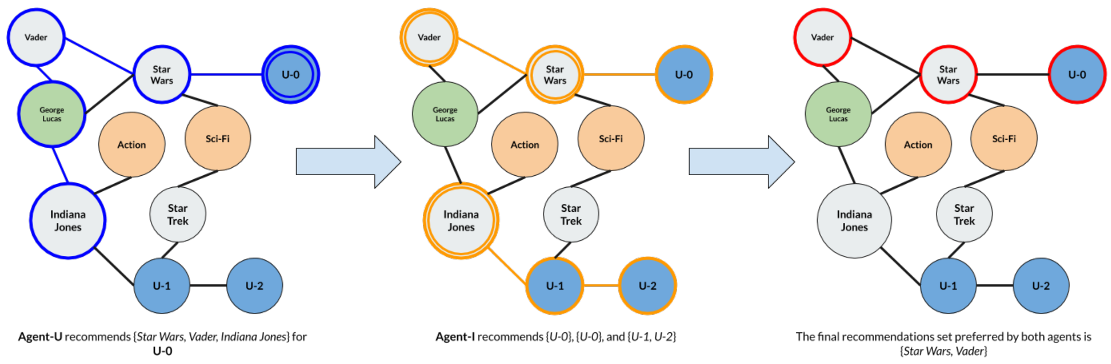
Vader
Star Wars
U-0
George Lucas
Action
Sci-Fi
Indiana Jones
Star Trek
U-1
U-2
Agent-U recommends {Star Wars, Vader, Indiana Jones} for U-0
Agent-I recommends {U-0}, {U-0}, and {U-1, U-2}
The final recommendations set preferred by both agents is {Star Wars, Vader}

Figure 2: Outline of *Skywalker*. The key idea is to force the final to satisfy both Agent$_U$ and Agent$_I$, while prioritizing the main agent - Agent$_U$.

where each group contains same number of items or brands (i.e., P_1 represents the least popular group, P_k represents the most popular group). Thereafter, given a slate of recommendations s, we hope the items come from as diverse groups as possible, thus forcing a fair exposed opportunity for those less popular items (or brands) in the marketplace. We calculate the fairness score by summing over the square root of the number of items each group contains in s as follow:

$$Fr(s) = \sum_{i=1}^{k} \sqrt{|item|_{\forall item \in s \cap P_i}}, \tag{9}$$

here s is also known as the *top-k recommendation*, and we set the size of s and the total number of groups sharing the same value k.

A simple case of study based on this definition is shown in Fig. 3. It is important to notice that, if all the recommended items come from the same group, we regard it as the least fair situation. Conversely, if all items come from different groups, it should be regarded of the highest fairness value. This is consistent with our definition, e.g., $\sqrt{1}+\sqrt{1}+\sqrt{1}>\sqrt{2}+\sqrt{1}>\sqrt{3}$.

nDCF@k. One of the notable drawbacks of the *Fairness@k* metric lies in not considering about the ranking of the recommendation; however, the ranking position always matters in recommendation, because users are likely to purchase the items which occur higher on the recommendation list. Still take the case study shown in Fig. 3 on cellphones, in the former definition of fairness, considering the brands of recommended items, a recommendation of $s_1 = \{\textit{Apple}, \textit{Xiaomi}, \textit{Oneplus}\}$ is of the same fairness value with $s_2 = \{\textit{Oneplus}, \textit{Xiaomi}, \textit{Apple}\}$. However, a user may still probably purchase *Apple* more than *Oneplus* in s_1 recommendation, since *Apple* occurs on the first place. By comparing s_1 and s_2, we regard s_2 should be more fair to the marketplace: it offers the less popular brands higher opportunities being exposed by ranking the corresponding items in higher orders, also it won't undermine those most popular brands (e.g., *Apple*) too much, since they are already of very high popularity and the slate of recommendation still contains their items (e.g., s_2 still contains the items from *Apple*).

Therefore, we would like to incorporate the ranking position in our second definition on fairness, and we borrow the idea from normalized discounted cumulative gain (*nDCG*) [12]. We first assign each group $P_1, P_2, ..., P_k$ a numerical value r from 1 to k respectively, and a higher value indicates a higher purchased times and popularity. Then we define our discounted cumulative fairness (*DCF*) as:

$$DCF = \sum_{i=1}^{|s|} \frac{k - r_{a_i}}{log_2(i+1)}, \tag{10}$$

where s is a slate of recommendation, r_{a_i} is the corresponding value r of the group containing the i^{th} item in s. We use "$k - r_{a_i}$" in the nominator to forcely offer a larger fairness score for recommending a less popular item.

In order to normalize the fairness value between 0 and 1, we first calculate the ideal DCF (IDCF) value as:

$$IDCF = \sum_{i=1}^{|\text{sorted}(s)|} \frac{k - r_{a_i}}{log_2(i+1)}, \tag{11}$$

where *sorted(s)* is a permutation of s based on the popularity value r_{a_i} from lowest to highest. Then, the normalized discounted cumulative fairness (nDCF) is calculated as:

$$nDCF = \frac{DCF}{IDCF}. \tag{12}$$

4 Experiments

In this section, we describe the process of our experiments in detail. We first introduce the statistics of the datasets we investigate and the evaluation metrics we use, followed by our choice of the baseline models for comparison and the implementation details.

4.1 Datasets

We evaluate our proposed model with the state-of-the-art methods on real-world datasets based on the Amazon 2014 *Metadata* and *5-core* datasets [10]. The datasets we use in our experiments contain the E-commerce record on three categories: $Beauty$, $Cellphones$, and $Clothing$. For each category, we generate a knowledge graph as the environment of our agent by extracting information from the *Metadata*, while doing the recommendation task based on the *5-core* dataset. Each *Metadata* contains 5 categories of entities with 8 different relations, for which the statistics are shown in Table 1. Notice that we follow the same strategy in paper [33] to adopt TF-IDF on the $Mention$ and $Described\_by$ relations to exclude less important word features, since both of them occupy a large proportion among all the relations. For each *5-core* dataset, we uniformly use 70% of which as the training set, and the rest as the testing set, uniformly keeping this setting for all the categories and models.

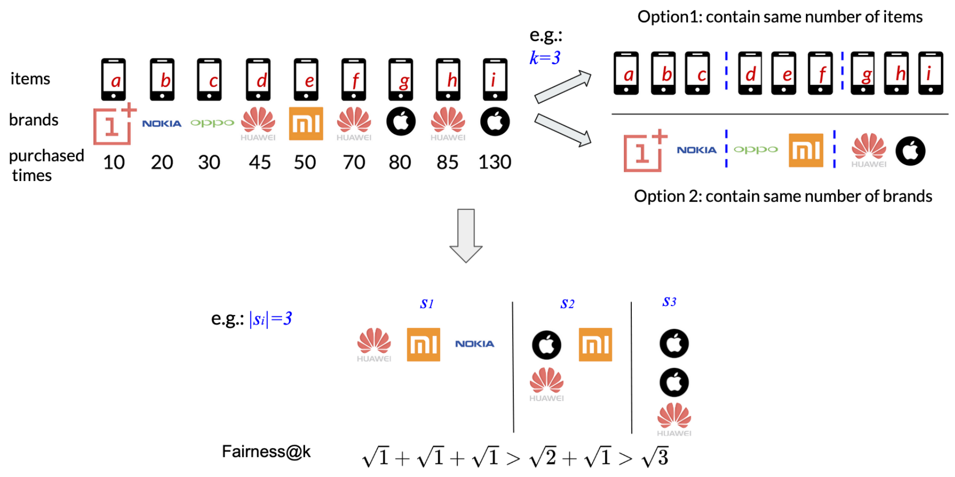
items
brands
purchased times
10
20
30
45
50
70
80
85
130
e.g.:
k=3
Option1: contain same number of items
Option 2: contain same number of brands
e.g.: |si|=3
s1
s2
s3
Fairness@k
$\sqrt{1}+\sqrt{1}+\sqrt{1} > \sqrt{2}+\sqrt{1} > \sqrt{3}$

Figure 3: Considering a cellphone marketplace which contains 9 cellphones of 6 different brands, the items are sorted by their purchased times. If we set $k = 3$, there are several ways to split the items into k different groups based on their purchased times: each group contains same number of items or same number of brands. For our study case, let's use option 2. For three slates of recommendations shown below, we display the brands information of each items and consider which group the corresponding brand belongs to, then we can calculate the Fairness@k score based on our definition. The ranking of fariness score is also consistent with the intuition.

			Amazon_Beauty	**Amazon_Cellphones**	**Amazon_Clothing**
Entities	**Description**		**Numbers**		
User	User in recommender systems		22363	27879	39387
Item	Product for recommendation		12101	10429	23033
Feature	Word in reviews for pruduct description		22564	22493	21366
Brand	Supplier information for products		2077	955	1182
Category	Different classification of products		248	206	1193
Relations	**Head Entity Type**	**Tail Entity Type**	**Average Numbers per Head**		
Purchase	*User*	*Item*	8.88	6.97	7.08
Mention	*User*	*Feature*	806.89	652.08	440.20
Described_by	*Item*	*Feature*	1491.16	1743.16	752.75
Belong_to	*Item*	*Category*	4.11	3.49	6.72
Produced_by	*Item*	*Brand*	0.83	0.52	0.17
Also_bought	*Item*	*Feature*	73.65	56.53	61.35
Also_viewed	*Item*	*Item*	12.84	1.24	6.29
Bought_together	*Item*	*Item*	0.75	0.81	0.69

Table 1: Details and statistics for three Amazon E-commerce datasets we use. Information rearranged from paper [33].

4.2 Evaluation Metrics

We evaluate our model versus other methods in terms of nDCG@k, Precision@k, Recall@k, and HR@k. For each user, nDCG@k is the normalized discounted cumulative gain at top-k recommended items, which takes the position of correctly recommended items into account. Precision@k indicates the proportion of recommended items in the top-k set that are relevant. Recall@k indicates what percentage of their rated items can emerge in the top-k recommended. HR@k represents the proportion of relevant items occurring in the top-k recommendation.

Also, we evaluate fairness scores of all the models based on the two variations of fairness metrics defined before: Fairness@k and nDCF@k .

All of the metrics are computed based on the choice of $k = 10$.

4.3 Methods Studied

In order to demonstrate the effectiveness of our model, we compare to the following baseline recommendation models: **BPR** [24], Bayesian Personalized Ranking, which is a basic model that learns latent vectors of users and items; **VBPR** [9], Visual Bayesian Personalized Ranking, which is built upon the BPR model but incorporates visual information of products; **WBPR** [8], Weighted Bayesian Personalized Ranking, which is another variation of BPR but takes into account the sampling probability of the negative items; **NCF** [11], Neural Collaborative Filtering, which replaces the inner product on the latent features of users and items in collaborative filtering with a neural architecture; **CKE** [38], Collaborative Knowledge Base Embedding, which leverages the information from an item knowledge base to enrich the item representations in the user-item matrix; **PGPR** [33], which combines knowledge graph and reinforcement learning to achieve an explainable recommendation (This is also the model our proposed ***Skywalker*** builds upon).

4.4 Implementation Details

We implement ***Skywalker*** in Pytorch [23]. In order to enhance the connectivity of the graph, we add reverse edges in our graph G': if $(e, r, e') \in G'$, we also add $(e', r, e) \in G'$. Then we use *TransE* [2] method to obtain all the representation vectors of entities and relations on graph G', with all the embedding size setting to be 100. For the formulation of MDP, we simplify the representation of state as: $\mathbf{s}_t = (\mathbf{e}_0, \mathbf{e}_{t-1}, \mathbf{r}_t, \mathbf{e}_t)$; therefore, the state vector is of size 400 by calculating the embeddings of its four components. For the action space pruning strategy, we set $\alpha = 250$, setting a upper bound as 250 possible actions in one action space. The dropout rate implemented on the pruned action space is set to be 0.5. The discount value γ in the discounted cumulative reward function is set to be 0.99. Since we regard shorter paths should indicate more relevant relations, we set the maximum hop step $T = 3$. As for the *Skywalker*, we set the recommendation size expansion coefficient β in $\{5, 4, 3, 2\}$. As for the training process, we use Adam [14] optimizer on all models for all three Amazon E-commerce datasets, with a learning rate of 0.0001 and a batch size of 32. All models are trained for 50 epochs. For the recommendation process, we follow the setting in Xian *et al.*'s work [33] by using beam search to generate paths for target users, and we keep the path with the highest probability for duplicate paths leading to the same item. Finally, we test and evaluate our models on a top-k recommendation, where k is set to be 10.

5 Results and Analysis

This section displays the results of all the models on three Amazon E-commerce datasets. We first compare the recommendation performance of each model and show the powerfulness of our proposed model *Skywalker*. Then we do a brief analysis on the fairness in recommendation, showing that there exits a trade-off between recommendation quality and fairness. Finally, we output all the three-hop reasoning paths, combining with several specific user cases to demonstrate the interpretability of our recommendation.

5.1 Analysis of Recommendation Performance

As shown in Fig. 2, our proposed model *Skywalker* consistently outperforms any other models in terms of all four metrics: nDCG, Precision, Recall, and HR. For instance, even compared with the other reinforcement learning and knowledge graph based approach, *PGPR*, *Skywalker_5* achieves an average performance improvement of 68.5% on Amazon_Beauty dataset, 34.3% on Amazon_-Cellphones dataset, and 91.2% on Amazon_Clothing dataset, while *Skywalker_2* achieves an average performance improvement of 30.0%, 19.5%, and 38.7%, respectively. These good performances demonstrate the effectiveness of the multi-agent setting in our model.

It is also important to notice that *CKE* outperforms the former baseline models, which proves the efficacy of using knowledge graph to incorporate more side information into the recommender systems. By comparing the results of *CKE* and *PGPR*, we can tell that it is more powerful to combine the KG-based approach with reinforcement learning idea, even not considering about the interpretability. The recommendation policy leaned by the agent tends to have a good understanding of the environment, thus leading to a higher quality of recommendation than just doing a relevance score matching.

Model	Amazon_Beauty				Amazon_Cellphones				Amazon_Clothing			
	nDCG@k	Precision@k	Recall@k	HR@k	nDCG@k	Precision@k	Recall@k	HR@k	nDCG@k	Precision@k	Recall@k	HR@k
BPR	2.657	1.026	2.905	8.323	0.724	0.543	1.316	5.764	0.492	0.192	0.699	1.567
VBPR	1.901	0.902	2.786	5.627	1.797	0.507	3.489	4.873	0.631	0.228	0.845	1.629
WBPR	2.604	1.034	4.362	8.765	2.751	0.747	4.376	6.793	2.83	0.298	2.49	3.679
NCF	2.656	1.308	5.943	9.342	2.452	0.691	4.034	8.273	1.668	0.315	1.932	3.736
CKE	3.396	1.546	5.949	10.645	3.364	0.965	5.515	9.415	2.378	0.329	2.892	4.073
PGPR	3.947	1.272	6.004	10.938	3.765	0.872	5.999	8.415	2.308	0.591	3.885	5.742
Skywalker_5	**5.102**	**2.432**	**10.805**	**18.991**	**4.334**	**1.243**	**8.407**	**11.733**	**3.239**	**1.276**	**8.022**	**11.587**
Skywalker_4	4.945	2.289	10.243	18.074	4.299	1.210	8.223	11.500	3.134	1.168	7.480	10.798
Skywalker_3	4.780	2.089	9.496	16.805	4.247	1.168	7.967	11.152	2.988	1.044	6.759	9.779
Skywalker_2	4.441	1.767	8.138	14.546	4.112	1.075	7.374	10.298	2.738	0.865	5.673	8.253

Model	Amazon_Beauty		Amazon_Cellphones		Amazon_Clothing	
	Fairness@k	nDCF@k	Fairness@k	nDCF@k	Fairness@k	nDCF@k
BPR	**6.823**	**0.893**	**7.121**	**0.917**	**7.098**	**0.923**
VBPR	5.423	0.883	6.753	0.903	7.032	0.918
WBPR	6.289	0.865	5.876	0.892	6.453	0.872
NCF	5.782	0.845	5.887	0.882	5.253	0.883
CKE	5.698	0.823	5.658	0.893	4.987	0.862
PGPR	3.592	0.721	3.603	0.758	3.589	0.743
Skywalker_5	2.523	0.384	2.226	0.324	2.400	0.311
Skywalker_4	2.761	0.401	2.546	0.346	2.654	0.376
Skywalker_3	2.879	0.437	2.801	0.404	2.834	0.422
Skywalker_2	3.198	**0.512**	3.097	**0.530**	3.128	**0.532**

Table 2: Comparisons of models on three Amazon E-commerce datasets. The results for nDCG@k, Precision@k, Recall@k, and HR@k are reported in percentage (%). All the metrics are evaluated on the top-k recommendation, where $k = 10$. The number n in "Skywalker_n" represents the recommendation size expansion coefficient β aforementioned.

5.2 Analysis of Fairness

The results on fairness strongly show that there is a trade-off between the fairness score and the quality of recommendation: recommendation of higher nDCG, Precision, Recall, and HR tends to have a poor fairness socre; conversely, models which perform not well on the four metrics of recommendation quality generally obtain high scores on fairness. From our perspective, this phenomenon is not surprising, since it is true that the marketplace in reality is not fair and "superstar" economy always happens: only a small number of items can receive the most highest relevant score to most users. Therefore, if a recommendation hopes to recommend more less popular items, the relevance score between the items and users will reduce, thus learning to the performance drop on traditional evaluation metrics.

However, it is still comforting to notice that *Skywalker_2* does not perform too bad on the fairness metrics by achieving an nDCF score over 50% on all three datasets. In regard of the notable improvement on the four quality metrics, we regard this model is successful on all the evaluations, in general.

5.3 Analysis of Interpretability

We display all the patterns of paths we find for the agents when setting $T = 3$, representing a path of three hops. The results are shown in Fig. 4. Instead of merely showing the global path reasoning patterns, we also offer three examples for specific user case, one for each dataset, which is shown in Fig. 5.

In the case study of *Amazon_Beauty* dataset, the user purchased a skin moisturizer, which is produced by Biotherm. One of the recommended items for this user is thus another product with the same brand. In the case study of *Amazon_Cellphones* dataset, the user purchased a screen protector. The recommendation agent first found another screen protector viewed by the user, and recommended a mobile phone charger, considering other people who purchased the second screen protector would also buy the charger. In the case study of *Amazon_Clothing*, the first user mentioned two features in his review which are "comfortable" and "perfect". The agent found s second user who bought a pair of shoes also mentioned that two features; therefore, the model recommended that pair of shoes to the first user.

In all, it is obvious to notice that our model is of clear and reasonable interpretability in our recommendation.

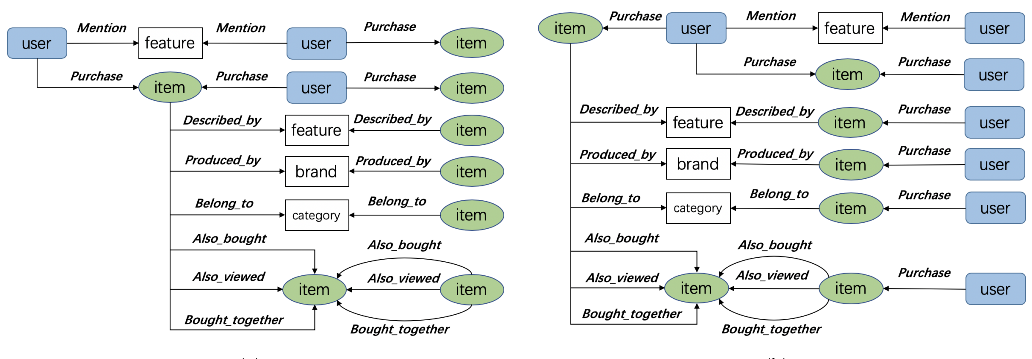

Figure 4: All three-hop paths found in our path reasoning. (a) the paths of Agent$_U$, which recommends items to a given user; (b) the paths of Agent$_I$, which does the reverse.

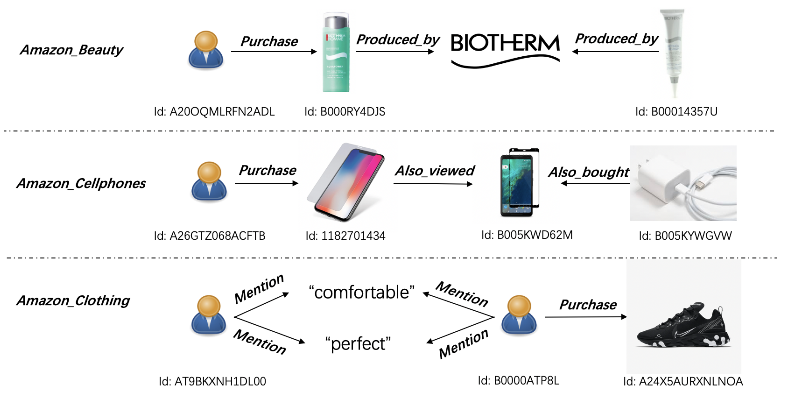

Figure 5: Case study of path reasoning recommendation for Amazon E-commerce datasets.

6 Conclusions and Future Work

We successfully achieved our goal of creating an explainable and fairness-aware multi-stakeholder recommender system. By incorporating the idea of knowledge graph, we are able to enrich the representation of users and items by using comprehensive side information and the relations between them; by using the reinforcement learning, the agent can do recommendation and simultaneously generate reasonable paths on the knowledge graph, which solves the limitation existing in traditional pure KG-based approaches. Two key contributions of this work lie into (1) proposing a multi-agent idea in the reinforcement learning architecture to enhance the quality of recommendation greatly and (2) defining novel fairness metrics on a two-sided marketplace, instead of just considering one side - the user side. In all, our proposed model *Skywalker* not only performs very well on typical evaluation metrics on recommendation quality, e.g., nDCG, Precision, Recall, and HR, but also achieves generating very clear and reasonable interpretability. And we regard *Skywalker_2* achieves the best balance between all of the evaluation metrics.

Future directions of this work include to investigate how to balance the trade-off between fairness and other quality evaluation of recommendation. A possible approach is to regard the problem as a multi-task learning, then jointly optimizing multiple loss objectives. This approach may also bring more stronger theoretical support. Other further extension and investigation includes: (1) Try to use more advanced embedding methods on knowledge graph rather than simple *TransE*. Other models such as *TransH* - which is more suitable for modeling one-to-many/many-to-one relations [18], or *TransD* - which models entities and relations in two distinct spaces (rather than assuming they are in the same semantic space) [13], are likely to improve results; (2) There are practical limitations to the number of stakeholders n and the size of the recommendation list k, as the time complexity to make a recommendation changes from a constant $O(1)$ to $O(nk)$.

References

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, Sep 2018.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[3] Avishek Joey Bose and William L. Hamilton. Compositional fairness constraints for graph embeddings, 2019.

[4] Y Cao, W Li, and D Zheng. *An Improved Neighborhood-Aware Unified Probabilistic Matrix Factorization Recommendation*, volume Wireless Pers Commun 102. 2015.

[5] Allison J.B. Chaney, David M. Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, page 43–50, New York, NY, USA, 2015. Association for Computing Machinery.

[6] Jianguo Chen, Kenli Li, Huigui Rong, Kashif Bilal, Nan Yang, and Keqin Li. A disease diagnosis and treatment recommendation system based on big data mining and cloud computing. *Information Sciences*, 435:124–149, 2018.

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

[8] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Personalized ranking for non-uniformly sampled items. In Gideon Dror, Yehuda Koren, and Markus Weimer, editors, *Proceedings of KDD Cup 2011*, volume 18 of *Proceedings of Machine Learning Research*, pages 231–247. PMLR, 21 Aug 2012.

[9] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback, 2015.

[10] Ruining He and Julian McAuley. Ups and downs. *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, 2016.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. *Proceedings of the 26th International Conference on World Wide Web*, Apr 2017.

[12] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[13] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, 2015.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[15] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, March 1997.

[16] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 535–541, Cambridge, MA, USA, 2000. MIT Press.

[17] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping, 2018.

[18] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[19] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[20] Chih-Chao Ma. A guide to singular value decomposition for filtering. 2008.

[21] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th acm international conference on information and knowledge management*, pages 2243–2251, 2018.

[22] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 2243–2251, New York, NY, USA, 2018. Association for Computing Machinery.

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

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback, 2012.

[25] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, page 175–186, New York, NY, USA, 1994. Association for Computing Machinery.

[26] Sherwin Rosen. In *The economics of superstars. The American economic review 71, 5*, page 845–858. 1981.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 285–295, New York, NY, USA, 2001. Association for Computing Machinery.

[28] Ben Schafer, Dan Frankowski, and Shilad Sen. Filtering recommender systems, 2007.

[29] Weiping Song, Zhijian Duan, Ziqing Yang, Hao Zhu, Ming Zhang, and Jian Tang. Explainable knowledge graph-based recommendation via deep reinforcement learning. *arXiv preprint arXiv:1906.09506*, 2019.

[30] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. mit press. In *Twenty-ninth AAAI conference on artificial intelligence*, 2018.

[31] Georgios Theocharous, Philip S. Thomas, and Mohammad Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1806–1812. AAAI Press, 2015.

[32] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 2018.

[33] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294, 2019.

[34] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[35] Xiwang Yang, Harald Steck, and Yong Liu. Circle-based recommendation in online social networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 1267–1275, New York, NY, USA, 2012. Association for Computing Machinery.

[36] Sirui Yao and Bert Huang. Beyond parity: Fairness objectives for collaborative filtering. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2921–2930. Curran Associates, Inc., 2017.

[37] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, page 283–292, New York, NY, USA, 2014. Association for Computing Machinery.

[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 353–362, New York, NY, USA, 2016. Association for Computing Machinery.

[39] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 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, page 635–644, New York, NY, USA, 2017. Association for Computing Machinery.

[40] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 167–176, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[41] X Zhou, J He, G Huang, and Y Zhang. *SVD-based incremental approaches for recommender systems*, volume 81. 2015.