Reinforced Negative Sampling over Knowledge Graph for Recommendation

Xiang Wang
National University of Singapore
xiangwang@u.nus.edu

Yaokun Xu
Southeast University
xuyaokun98@gmail.com

Xiangnan He
University of Science and Technology of China
xiangnanhe@gmail.com

Yixin Cao
National University of Singapore
caoyixin2011@gmail.com

Meng Wang
HeFei University of Technology
eric.mengwang@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

Properly handling missing data is a fundamental challenge in recommendation. Most present works perform negative sampling from unobserved data to supply the training of recommender models with negative signals. Nevertheless, existing negative sampling strategies, either static or adaptive ones, are insufficient to yield high-quality negative samples — both informative to model training and reflective of user real needs.

In this work, we hypothesize that item knowledge graph (KG), which provides rich relations among items and KG entities, could be useful to infer informative and factual negative samples. Towards this end, we develop a new negative sampling model, *Knowledge Graph Policy Network* (KGPolicy), which works as a reinforcement learning agent to explore high-quality negatives. Specifically, by conducting our designed exploration operations, it navigates from the target positive interaction, adaptively receives knowledge-aware negative signals, and ultimately yields a potential negative item to train the recommender. We tested on a matrix factorization (MF) model equipped with KGPolicy, and it achieves significant improvements over both state-of-the-art sampling methods like DNS [39] and IRGAN [30], and KG-enhanced recommender models like KGAT [32]. Further analyses from different angles provide insights of knowledge-aware sampling. We release the codes and datasets at https://github.com/xiangwang1223/kgpolicy.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommendation, Knowledge Graph, Negative Sampling

ACM Reference Format:
Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380098

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380098

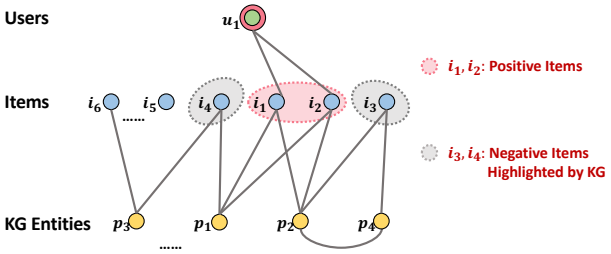


Figure 1: A toy example of distilling negative signals from KG. Having overlapping KG entities with i_1 and i_2, i_3 and i_4 are more likely to be of less interest to u_1.

1 INTRODUCTION

Recommender systems have been widely adopted in real-world applications to improve user satisfaction and engagement. To train a recommender model from historical user-item interactions, both positive and negative user feedback are required to ensure that the model generates reasonable personalized ranking [13, 23, 33]. Nevertheless, most interactions are in the form of implicit feedback, *e.g.,* clicks and purchases, which provide only the signals on positive feedback. This brings in the fundamental challenge to recommender model learning — how to distill negative signals from the positive-only data — which is also known as the one-class problem [20].

As negative signals are latent in unobserved data, a prevalent solution is to perform negative sampling [23], which is more efficient and versatile than treating all unobserved interactions as negative [13, 20]. Existing strategies on negative sampling can be categorized into three types — static sampler [3, 7, 23], adaptive sampler for hard negatives [22, 30, 37, 39], and enhanced sampler with extra behaviors [7, 19]. However, each type of method suffers from some inherent limitations:

- Static sampler applies fixed distribution to sample from missing data, *e.g.,* uniform [12, 23] and popularity-biased distribution [3, 4]. The main limitation is that it is independent of model status, making it easier to yield low-quality negative samples. For example, if a user-item instance is already scored lowly by the current model, sampling it as a negative example will have minor change on model parameters (as gradients are close to zero [22]).
- Adaptive sampler prefers hard negatives, since training on them can bring large change on current model parameters [22, 30, 39]. For example, DNS [39] picks the one scored highest by the current model among some random unobserved samples, and IRGAN [30] optimizes this process under a generative adversarial

framework. While being effective from the perspective of numerical optimization, these hard negatives are likely to be true positives (in future testing data), which degrades model effectiveness.
- Some recent works incorporate extra behavior data, *e.g.,* viewed but non-clicked [6, 7] and clicked but non-purchased [19], to enhance the negative sampler. Although such data provides certain signal on true negatives, it is of limited scale as compared with the vast amount of missing data. As such, only using them as negative feedback is rather insufficient (even performs worse than the uniform sampler [6]). For this type of method, it still needs strategies to effectively distill negative signal from the massive missing data.

Given the fundamental role of negative sampling and the limitations of existing methods, we focus on negative sampling in this work, aiming to improve its quality by introducing other types of data. We claim that high-quality negative samples should satisfy two requirements: 1) *informative*, meaning that the current model rates them relatively high, such that updating them as negative feedback will change model parameters significantly, and 2) *factual*, meaning that they are true negatives, *i.e.,* the user has known them before (exposed by the system or through other ways) but did not choose them. As the requirement of informative can be achieved by adaptive sampler, the key challenge lies in discovering factual negatives from missing data, which lacks ground-truth by nature and has not been well addressed by previous work.

In this work, we hypothesize that knowledge graph (KG), which introduces extra relations among items and real-world entities (from item attributes or external knowledge), could be useful to infer factual negative samples from unobserved data. Although incorporating KG into recommendation has recently been extensively researched [27, 32, 38], these studies only leverage KG to build the predictive model, and none of previous works has considered using it to enhance the negative sampler. In particular, they assume that items, which have overlapping KG entities with historical items, would be unexposed but of interest to the target user. However, in real-world scenarios, a user is often aware of these items through some ways (*e.g.,* searching, mouth marketing, or advertising systems); hence, she does not adopt them, suggesting that she might be truly not interested in these items. An example is shown in Figure 1, where user u_1 watches movies i_1 and i_2, both of which are directed by the same person p_1 and of the same genre p_2. We may infer that the combination of director p_1 and genre p_2 is an important factor of u_1's interest. Thus u_1 is highly likely to have known other movies (*e.g.,* i_4) directed by p_1 but with different genres, but be less interested on them. As such, from the perspective of negative sampling, i_4 could offer high-quality negative signals.

Despite great potential, it is highly challenging to exploit KG to guide negative sampling. First, when exploring KG towards possible negative items, the scale of exploring paths (*e.g.,* $\{i_1, i_2\} \rightarrow p_1 \rightarrow i_4$ in Figure 1) increases dramatically, since many edges are continually added at each forward step. Thus, there is a strong need for an intelligent sampler to effectively traverse KG. Second, due to the lack of ground-truth, it requires the sampler to distinguish negative signals carried by the exploring paths — more specifically, differentiate the confidence of KG entity (*e.g.,* p_1) being exposed to the target user and estimate the probability of possible item (*e.g.,* i_4) being negative.

Towards this end, we propose a new negative sampling model, KGPolicy (short for *Knowledge Graph Policy Network*), which employs a reinforcement learning (RL) agent to explore KG to discover high-quality negative examples. At the core is the designed **exploration operation**, which navigates from the positive item, picks two sequential neighbors (*e.g.,* one KG entity and one item) to visit. Such a two-hop path captures the knowledge-aware negative signals. We devise a neighbor attention module to achieve this goal, which specifies varying importance of one- and two-hop neighbors conditioned on the positive user-item pair, so as to adaptively capture personal tastes on KG entities and yield potential items. By conducting such exploration recursively, KGPolicy learns to select potential negative items for a target positive interaction. Moreover, the path history works as a support evidence revealing why the selected item is being treated as a negative instance. To demonstrate our method, we employ a simple linear model, matrix factorization (MF), as the recommender and co-train it with our KGPolicy. Empirically, MF equipped with KGPolicy achieves significant improvements over both state-of-the-art sampling methods like DNS [39] and IRGAN [30], and KG-enhanced recommender models like KGAT [32], on three benchmark datasets. Further analyses provide insights on how knowledge-aware negative samples facilitate the learning of recommender *w.r.t.* two requirements — informative (evidence in the training process *w.r.t.* gradient magnitude, and the performance *w.r.t.* sparsity level) and reflective of personal tastes (evidences in case study). It is worth highlighting that KGPolicy is recommender-agnostic and can works as a plug-and-play sampler for arbitrary recommenders.

In a nutshell, this work makes the following main contributions:

- To the best of our knowledge, we are the first to incorporate knowledge graph into negative sampling, with the aim of selecting high-quality negative samples to pair with a positive user-item interaction.
- We develop a reinforcement learning agent for negative sampling, KGPolicy, which effectively learns to navigate towards high-quality negative items with multi-hop exploring paths.
- We conduct extensive experiments on three benchmark datasets, demonstrating the advantages of KGPolicy on the effectiveness of sampling and the usage of knowledge entries.

2 TASK FORMULATION

We first present interaction data and knowledge graph, formulate our task, and emphasize negative signals within multi-hop paths.

Interaction Data. Let $O^+ = \{(u, i)|u \in \mathcal{U}, i \in \mathcal{I}\}$ be the implicit feedback, where each (u, i) pair indicates a historical interaction between user u and positive item i, and $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items, respectively.

Knowledge Graph. Inspired by recent works [2, 32], we organize item attributes or external knowledge as well as interaction data in the form of knowledge graph (KG). As prior efforts [40] show, items in user-item interaction data can be aligned with the corresponding entities in KG. With such alignments, we can establish a KG, formalized as $\mathcal{G} = \{(e, e')|e, e' \in \mathcal{E}\}$, where $\mathcal{E}$ is the entity set

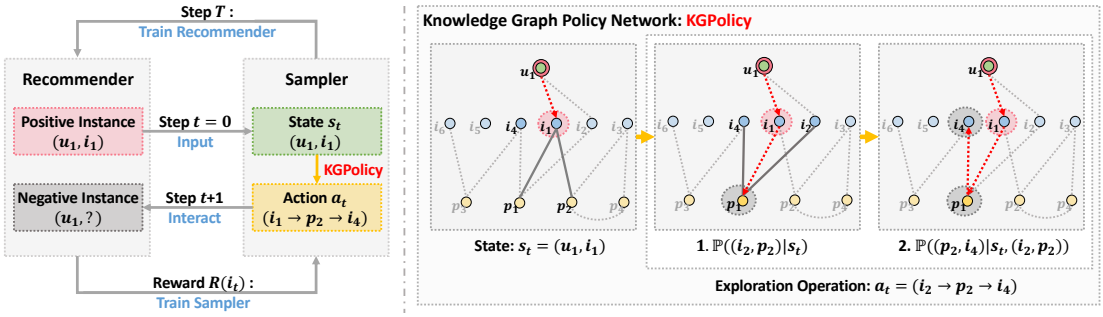


Figure 2: Illustration of the proposed knowledge-aware negative sampling framework. The left subfigure shows the model framework, and the right subfigure presents the proposed KGPolicy network. Best view in color.

unifying the user and item sets $\mathcal{U}$ and $\mathcal{I}$ with KG entity set $\mathcal{P}$, *i.e.*, $\mathcal{E} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{P}$. For example, (*item: Harry Potter*, *author: J. K. Rowling*) describes the fact that *J. K. Rowling* is the author of the book *Harry Potter*. Here we simplify the entity-relation-entity KG triples as the edges between conceptualized entities, merging the semantic relations into the entity instances and leaving the explicit modeling of them in future work.

Task Description. Having established user behaviors and item knowledge, we aim to exploit such rich information to guide the sampler learning. Our goal is the knowledge-aware negative sampling, formulated as:

$$j \sim f_S(u, i, \mathcal{G}), \tag{1}$$

where $f_S(\cdot)$ is the sampler parameterized with Θ_S. It generates the empirical distribution over unobserved items to yield knowledge-aware negative item j, which is expected to be informative and reflective of personal tastes. Furthermore, the exclusive KG entities of positive i, $\{p|(i,p) \in \mathcal{G}, (j,p) \notin \mathcal{G}\}$, helps explain why the target user u is less interested on negative j. For example, $\{p_2\}$ might be the reason on u_1's behavior difference between i_1 and i_4.

Negative Signals in Multi-hop Paths. Towards that, we aim to explore the structural information of KG, especially high-order connectivity among nodes, to discover suitable negative items. For a positive (u, i) interaction, we can traverse paths rooted at node i, terminate at a unobserved item j, and view multi-hop connections as the relationships between i and j. However, it is obvious that various paths have different confidences in the discovered items being negative, and not all paths are useful for distilling negative signals. We hence heuristically define the atomic path type as:

- $i \to e' \to j$ with $e' \in \mathcal{E}$,

which is 1) **informative**, since two items i and j share the same KG entity e', they might have similar representations and their pairwise comparison might provide large gradients on the recommender parameters; and 2) **reflective of user real tastes**, since if e' is an important factor of u's interest, j might have been exposed to u through other means (*e.g.*, searching, mouth marketing, or advertising systems). However, u consumed i rather than j, suggesting that she might be truly less interested in j compared to i. As a result, (u, j) is expected as a better negative sample to train the recommender. Moreover, if j is estimated with lower confidence being negative, we can continue the exploration by extending such atomic paths. For example, item j' can be discovered from $i \to e \to j' \to e' \to j$ with higher confidence of being negative.

3 METHODOLOGY

We now present knowledge-aware negative sampling. Figure 2 shows the framework, which consists of one recommender and the proposed sampler. We then elaborate our sampler, KGPolicy, with the target of learning to navigate towards informative negatives on KG. Specifically, there are three main components conducting the exploration operations: 1) graph learning module, which prepares high-quality representations of nodes in advance; 2) neighbor attention module, which utilizes two attention models to conduct path finding and determines which suitable node to visit next; and 3) neighbor pruning module, which reduce the search space to solve the computational overload in the foregoing module. Recursively performing such explorations, KGPolicy ultimately is able to yield a potential item to pair the positive target. Finally, KGPolicy and the recommender are co-trained for recommendation.

3.1 Recommender

To demonstrate the effectiveness of our knowledge-aware sampler, we employ a linear and simple model, matrix factorization (MF) [23], as the recommender. To be more specific, MF parameterizes ID information of users and items as embeddings, and uses inner product of user and item embeddings as the predictive function to estimate how likely user u would consume item i. The holistic goal is formulated as:

$$\hat{y}_{ui} = f_R(u, i) = \mathbf{r}_u^\top \mathbf{r}_i, \tag{2}$$

where $\hat{y}_{ui}$ is the prediction score for an (u, i) interaction; $f_R(\cdot)$ is abstracted as the interaction function with recommender parameters Θ_R; $\mathbf{r}_u \in \mathbb{R}^d$ and $\mathbf{r}_i \in \mathbb{R}^d$ are ID embeddings of user u and item i, respectively; d is the embedding size.

Following prior studies [12, 23, 34], we use the pairwise BPR loss [23] as the objective function to optimize and learn the parameters Θ_R. Specifically, it assumes that, for a target user, her historical items reflecting more personal interest should be assigned higher prediction scores, than that of unobserved items, as:

$$\min_{\Theta_R} \sum_{(u,i) \in O^+} \mathbb{E}_{j \sim f_S(u,i,\mathcal{G})} - \ln \sigma\Big(f_R(u, i) - f_R(u, j)\Big), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function. As such, the recommender as a critic judges whether the items i and j are truly consumed by user

u. Moreover, following previous work [22], the informativeness of a negative sample can be measured as the gradient magnitude, as:

$$\Delta_{u,i,j} = 1 - \sigma(f_R(u,i) - f_R(u,j)), \tag{4}$$

which reflects the contribution a pairwise preference (u,i,j) has for improving Θ_R. Low-quality negatives, which are assigned a smaller score than i, make gradient magnitude close to 0, hence contribute little to optimization. Hence, an informative negative is expected to have close prediction scores with the positive target.

3.2 Knowledge-aware Sampler

While adaptive samplers [22, 37, 39] achieve great success towards informative negatives, the discovery of factual negative signals are not fully explored. Towards this end, we take KG as the priors (or environment) of the sampler. This allows us to exploit rich relations, especially high-order connectivity, among items and KG entities for exploring more suitable negatives. As shown in the right side of Figure 2, the basic idea is to, conditioned on the target user, start from the positive item, learn to navigate over the KG structure, then yield the possible negatives along the exploring paths. Such paths are composed of the atomic paths defined in Section 2.

Enumerating possible paths towards all unobserved items is infeasible in large-scale KGs, since it requires labor-intensive feature engineering, being memory-consuming to store these paths and time-consuming to distill useful signals. Thus, we design an intelligent sampler as a reinforcement learning (RL) agent to conduct automatic exploration over KG.

3.2.1 ***Sampling as Reinforcement Learning.*** We cast sampling as a Markov Decision Process (MDP) $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$, where $\mathcal{A} = \{a\}$ is the set of actions derived from exploration operations, $\mathcal{S} = \{s\}$ is the set of states abstracting paths during exploration, $\mathcal{P}$ is the transition dynamics of states, and $\mathcal{R}$ is a reward function. We introduce these key elements of KG environment for RL as follows:

Exploration Operation. To obtain the atomic paths defined in Section 2, we define a new exploration operation involving two successive edges. Formally, at step t, $a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1})$ denotes a two-hop path rooted at item e_t towards a proposal e_{t+1}, where (e_t, e'_t), $(e'_t, e_{t+1}) \in \mathcal{G}$, and $e_t, e_{t+1} \in \mathcal{I}$ are connected via the internal node e'_t. Such operation considers the confidence of the KG entity e'_t being exposed to the target user u, and the confidence of item e_{t+1} being negative. With the budge of T exploration operations, the sampler $f_S(\cdot)$ generates a $2T$-hop paths $(a_1, \cdots, a_T)$, which sequentially yields T proposals as a policy:

$$\pi = (e_1, e_2, \cdots, e_T), \tag{5}$$

where the positive item i is the first node in a_0; e_t is the last item node of the t-th step. At the terminal step T, e_T is used as the final negative item to optimize the recommender. Varying the number of exploration operations T allows us to flexibly adjust the search space, so as to guarantee the diversity of negatives. We set T as 2 by default and evaluate the impact of T in Section 4.5.1.

State. At step t, the state s_t conditioned on the target user u is defined as a triple (u, e_t), where e_t is the node the sampler visits currently. As such, we can formalize the exploration process prior to step t as $\{s_0, a_1, s_1, \cdots, a_t, s_t\}$, where the initial state s_0 is $\{u, i\}$.

Action. The action space $\mathcal{A}_t$ of state s_t is composed of all possible exploration operations starting from node e_t, excluding the historical trajectory. As the state is changing during the exploration and its neighbors are different, the action space is dynamic.

State Transition Dynamics. Given an exploration operation a_t at state s_t, the transition to the next state s_{t+1} is determined as:

$$\mathbb{P}\Big(s_{t+1} = (u, e_{t+1}) | s_t = (u, e_t), a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1})\Big) = 1. \tag{6}$$

Reward. The reward $\mathcal{R}_{e_t}$ measures the quality of the proposal item e_t *w.r.t.* the pairwise preference (u,i,j) at step t. However, without ground truth of negative signals to confirm whether u is truly less interested on e_t, we rely on the feedback from the recommender to define the soft reward function. Here we consider two factors:

- **Prediction Reward:** the prediction of negative e_t is the typical reward in adversarial sampling methods [21, 30], accounting for the matching score between e_t and u. The sampler is encouraged by the recommender to yield items with higher predictions. From the perspective of informativeness, an item ranked close to the positive is able to offer larger gradient magnitude.
- **Similarity Reward:** intuitively, if e_t is similar to positive item i, e_t is more likely to be exposed to user u. The recommender enforces the sampler to care exposed but less interested items.

Considering these two factors, we design a reward function as:

$$\mathcal{R}(e_t) = f_R(u, e_t) + g_R(i, e_t), \tag{7}$$

where $f_R(u, e_t) = \mathbf{r}_u{}^\top \mathbf{r}_{e_t}$ and $g_R(i, e_t) = \mathbf{r}_i{}^\top \mathbf{r}_{\mathbf{e_t}}$ separately denote prediction and similarity rewards; $\mathbf{r}_u$, $\mathbf{r}_i$, and $\mathbf{r}_{e_t}$ are the recommender's ID embeddings of u, i, and e_t, respectively. Here we set equal importance for the two reward components, leaving their linear combination controlled by the hyper-parameter future. We verify the rationality of the two reward functions in Section 4.5.2.

Objective Function. Towards learning a stochastic policy π to optimize the sampler parameters Θ_S, we maximize the expected cumulative discounted reward as follows:

$$\max_{\Theta_S} \sum_{(u,i) \in O^+} \mathbb{E}_\pi [\sum_{t=1}^{T} \lambda^{t-1} \mathcal{R}(e_t)], \tag{8}$$

where γ is the decay factor; the expectation of π is to make the likelihood of a proposal pair as close to that of the possible interaction as possible. In what follows, we elaborate our policy network towards obtaining the probability of e_t being negative in π, *i.e.,* $\mathbb{P}(a_t|s_t)$, at t-th step.

3.3 Knowledge Graph Policy Network.

In this section, we introduce a network to generate policy π, as well as the confidence for each action. First, we describe a graph learning module, which generates representation for each node, and then build a neighbor attention module upon the representations to pick a suitable neighbor as a proposal to visit, which is coupled with a neighbor pruning module to reduce the exploration space.

3.3.1 ***Graph Learning Module.*** Inspired by recent graph neural networks (GNNs) [10, 17, 26, 34] which are powerful to generate representations for graph data, we employ GraphSage [10] on $\mathcal{G}$ and the user-item bipartite graph O^+, to embed user, item, and KG entity

nodes. In particular, at the l-th graph convolutional layers, a node e receives the information being propagated from its neighbors to update its representation, as:

$$\mathbf{h}_e^{(l)} = \rho\Big(\mathbf{W}^{(l)}(\mathbf{h}_e^{(l-1)}||\mathbf{h}_{\mathcal{N}_e}^{(l-1)})\Big), \tag{9}$$

where $\mathbf{h}_e^{(l)} \in \mathbb{R}^{d_l}$ is the representation after l steps of embedding propagation, d_l denotes the embedding size, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times 2d_{l-1}}$ is the weight matrix to distill useful information; $||$ is the concatenation operation and ρ is a nonlinear activation function set as LeakyReLU [10, 26] here; $\mathbf{h}_e^{(0)}$ represents ID embedding, while $\mathbf{h}_{\mathcal{N}_e}^{(l-1)}$ is the information being propagated from e's neighbors as:

$$\mathbf{h}_{\mathcal{N}_e}^{(l-1)} = \sum_{e' \in \mathcal{N}_e} \frac{1}{\sqrt{|\mathcal{N}_e||\mathcal{N}_{e'}|}} \mathbf{h}_{e'}^{(l-1)}, \tag{10}$$

where $\mathcal{N}_e = \{e'|(e,e') \in \mathcal{G} \text{ or } (e,e') \in O^+\}$ is the set of nodes which are connected with e. After stacking such L layers, we obtain the final representation for each node $\mathbf{h}_e = \mathbf{h}_e^{(L)}$. Compared with the initial ID embeddings, the GNN model injects graph structure into representation learning, so as to facilitate further exploration.

3.3.2 ***Neighbor Attention Module.*** At state $s_t = (u, e_t)$, having established representations for node e_t and its neighbors $\mathcal{N}_{e_t}$, we need to effectively search relevant actions towards potential items. In particular, we decompose an exploration operation $a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1})$ into two steps: 1) choosing an outgoing edge from e_t to the internal node e'_t, *i.e.,* (e_t, e'_t), and 2) determining the third node e_{t+1} conditioned on the historical steps. Such process is as:

$$\mathbb{P}(a_t|s_t) = \mathbb{P}((e_t, e'_t)|s_t) \cdot \mathbb{P}((e'_t, e_{t+1})|s_t, (e_t, e'_t)), \tag{11}$$

where $\mathbb{P}(a_t|s_t)$ represents the confidence or probability of e_{t+1} being negative; $\mathbb{P}(e_t, e'_t)$ and $\mathbb{P}(e'_t, e_{t+1})$ separately model the confidence of each exploration step. Here we implement these two exploration steps via two attention models.

Attentive KG Neighbors. For the current item node e_t, we need to estimate how likely its related KG entities $\mathcal{N}_{e_t}$ are exposed to user u. Intuitively, a user pays different attentions on various KG entities. For example, for a movie, a user might care more about the *Director* entity than the *Writer* entity. This indicates that other movies having the same *Director* entity are more likely to be exposed to the user. Furthermore, for different movies, her points of interest could change and be dynamic. For instance, a user watched two movies, because she is attracted by the *Director* and *Star* entities, respectively. Therefore, we devise an attention model to adaptively specify importance of neighbors, which are sensitive to the current state, *i.e.,* user u and current item e_t.

Formally, for each outgoing edge from e_t to neighbor $e'_t \in \mathcal{N}_{e_t}$, we generate its representation as $\mathbf{h}_{e_t} \odot \mathbf{h}_{e'_t}$, and then include user representation $\mathbf{h}_u$ to formulate its importance as:

$$p(e_t, e'_t) = \mathbf{h}_u{}^\top \rho(\mathbf{h}_{e_t} \odot \mathbf{h}_{e'_t}), \tag{12}$$

where $\odot$ is the element-wise product. Such importance score is dependent on the affinity between user u, item e_t, and KG entity e'_t. Thereafter, we employ a softmax function to normalize the scores across all neighbors as:

$$\mathbb{P}((e_t, e'_t)|s_t) = \frac{\exp\left(p(e_t, e'_t)\right)}{\sum_{e''_t \in \mathcal{N}_{e_t}} \exp\left(p(e_t, e''_t)\right)}. \tag{13}$$

Following such distribution, the KG entity e'_t can be selected from the set of candidates $\mathcal{N}_{e_t}$ with the corresponding probability.

Attention Item Neighbors. Having selected KG entity e'_t, we employ another attention model to decide yield which item from its neighbors $\mathcal{N}_{e'_t}$ as the proposal. Typically, different unobserved item neighbors have varying confidence of being negative, conditioned on KG entity e'_t and user u. Here we model the confidence of each edge from e'_t to $e_{t+1} \in \mathcal{N}_{e'_t}$ as:

$$p(e'_t, e_{t+1}) = \mathbf{h}_u{}^\top \rho(\mathbf{h}_{e'_t} \odot \mathbf{h}_{e_{t+1}}). \tag{14}$$

Then a softmax function is followed to generate the selection probability of item e_{t+1} as:

$$\mathbb{P}((e'_t, e_{t+1})|s_t, (e_t, e'_t)) = \frac{\exp\left(p(e'_t, e_{t+1})\right)}{\sum_{e''_{t+1} \in \mathcal{N}_{e'_t}} \exp\left(p(e'_t, e''_{t+1})\right)}. \tag{15}$$

As a result, we can generate the negative probability of each exploration operation for a policy π (*cf.* Equation (5)).

3.3.3 ***Neighbor Pruning Module.*** While such exploration over KG has narrowed the search space from the whole item sets to multi-hop neighbors of positive items, the neighbor scale of some nodes (*e.g.,* popular items or general KG concept like the genre of *Drama*) easily reach thousands or even larger. It is very computational expensive to calculate an output distribution over neighbors of such a node and smooth the probability (via Equations (13) and (15)), further hindering the exploration performance. Thus, we get inspiration from DNS [39] and propose a pruning strategy that can effectively keep promising neighbors.

More specifically, for each training epoch, we first construct a subset of neighbors, $\hat{\mathcal{N}}_e \subset \mathcal{N}_e$, which consists of n_1 randomly sampled neighbors (via either sub-sampling or oversampling), so as to reduce the search space. Moreover, to guarantee the diversity of samples, we additionally introduce some nodes randomly sampled from the whole space. Thereafter, we design a scoring function to select potential neighbors from $\hat{\mathcal{N}}_e$ to build the set $\tilde{\mathcal{N}}_e \subset \hat{\mathcal{N}}_e$ involving n_2 candidate neighbors, heuristically filtering useless nodes out. Formally, the scoring function is formalized as $g(u, e') = \mathbf{h}_e^\top \mathbf{h}_{e'}$, which is modeled as the representation similarity between e and $e' \in \hat{\mathcal{N}}_e$. Having obtained the confidence scores, we generate the ranking list over $\hat{\mathcal{N}}_e$ and select the top n_2 neighbors to construct $\tilde{\mathcal{N}}_e$. With such neighbor pruning strategy, we can use $\tilde{\mathcal{N}}_{e_t}$ and $\tilde{\mathcal{N}}_{e'_t}$ to replace the original set $\mathcal{N}_{e_t}$ and $\mathcal{N}_{e'_t}$ in Section 3.3.2, respectively. As a result, we can effectively solve the high time complexity of neighbor attention module (evidence from Section 3.4.4).

3.4 Model Optimization

Finally, we adopt the iteration optimization to train the recommender (*i.e.,* MF) and the sampler (*i.e.,* KGPolicy), where the recommender and sampler parameters are $\Theta_R = \{\mathbf{r}_u, \forall u \in \mathcal{U}, \mathbf{r}_i, \forall i \in \mathcal{I}\}$ and $\Theta_S = \{\mathbf{h}_e^{(0)}, \forall e \in \mathcal{E}, \mathbf{W}^{(l)}, \forall l \in \{1, \cdots, L\}\}$, respectively.

3.4.1 ***Recommender Optimization.*** We first freeze Θ_S, sample one negative item $j = e_T$ to pair one positive interaction and feed them into the recommender (*i.e.,* Equation (3)), and update Θ_R via stochastic gradient descent (SGD) [23].

3.4.2 ***Sampler Optimization.*** We then freeze Θ_R and update Θ_S. However, SGD cannot be directly applied to do the optimization, since the sampler involves discrete sampling steps which block the gradients when performing differentiation. A common solution is the policy gradient based RL (REINFORCE) [25] method, such that the gradients of $\mathcal{L}_S$ *w.r.t.* Θ_S are calculated as:

$$\nabla_{\Theta_S}\mathcal{L}_S = \nabla_{\Theta_S}\sum_{(u,i)\in O^+}\mathbb{E}_\pi[\sum_{t=1}^{T}\lambda^{t-1}\mathcal{R}(e_t)] \tag{16}$$
$$\simeq \sum_{(u,i)\in O^+}\frac{1}{T}\sum_{t=1}^{T}[\gamma^{t-1}R(e_t)\nabla_{\Theta_S}\log\mathbb{P}(a_t|s_t)].$$

Moreover, following prior studies [31], we also subtract a baseline b from the policy gradient to reduce the variance. Wherein, the baseline b is set as the average reward of the recently generated negative interactions. As such, in each iteration, we alternatively optimize the objective functions of the recommender and sampler.

3.4.3 ***False Negative Issue.*** It is hard for stochastic sampling approaches to avoid the false negative issue — some items sampled as negative by the sampler during training are truly positive during inference in further test dataset. Compared to adaptive samplers, KGPolicy benefiting from item knowledge could empirically alleviate the issue to some extent (evidences in Section 4.4.1).

3.4.4 ***Time Complexity Analysis.*** For the module of graph learning (*cf.* Section 3.3.1), establishing representations for nodes has computational complexity $O(\sum_{l=1}^{L}(|\mathcal{G}| + |O^+|)d_l d_{l-1})$. For the module of neighbor attention module (*cf.* Section 3.3.2), the time complexity which mainly comes from the calculation of attention scores is $O(2T|O^+|\tilde{\mathcal{N}}_e|d^2)$. As a result, the time cost of the whole training epoch is $O(\sum_{l=1}^{L}(|\mathcal{G}| + |O^+|)d_l d_{l-1} + 2T|O^+|n_2 d^2)$. Empirically, RNS, DNS, IRGAN, AdvIR, NMRN, and KGPolicy cost around 22s, 116s, 155s, 175s, 172s, and 232s per training epoch on the largest Yelp2018 dataset, respectively. Hence, KGPolicy has comparable complexity to adaptive samplers, especially the adversarial ones (IRGAN, AdvIR, and NMRN).

4 EXPERIMENT

We evaluate our proposed KGPolicy on three public datasets, aiming to answer the following research questions:

- **RQ1**: How does KGPolicy perform, compared with existing methods *w.r.t.* two dimensions — the effectiveness of negative sampling and the usage of knowledge entires?
- **RQ2**: How do different components (*e.g.,* number of exploration operations, reward functions) affect KGPolicy?
- **RQ3**: Can KGPolicy provide in-depth analyses of negative samples?

4.1 Dataset Description

We use three publicly available datasets: Amazon-book, Last-FM, and Yelp2018, released by KGAT [32]. Each dataset is composed of two components, the user-item interactions and KG derived from Freebase (Amazon-book and Last-FM) or local business information network (Yelp2018), as summarized in Table 1. Specifically, each knowledge-aware fact is represented as a conceptual edge (item: i_1, author: p_1). Note that, we omit the explicit modeling of semantic relations among KG entities and merge the entity type into the entity instances; we leave it for future work.

Table 1: Statistics of the datasets.

		Amazon-book	Last-FM	Yelp2018
User-Item Interaction	#Users	70, 679	23, 566	45, 919
	#Items	24, 915	48, 123	45, 538
	#Interactions	847, 733	3, 034, 796	1, 185, 068
Knowledge Graph	#Entity Types	39	9	42
	#KG Entities	88, 572	58, 266	90, 961
	#Edges	2, 557, 746	464, 567	1, 853, 704

We use the same training and test sets as that of KGAT [32]. That is, for each user, the interaction history is split into the training and test parts with the ratio of 80% : 20%. In the training set, we view each observed user-item interaction as a positive instance, while using the sampler to sample a negative item to pair the same user, and build the recommender.

4.2 Baselines

We compare our proposed method, KGPolicy, with two groups of the state-of-the-art baselines, as follows:

4.2.1 ***Negative Sampling Methods.*** To verify the effectiveness of negative sampling, we select the state-of-the-art sampling methods as baselines, covering the static (RNS and PNS), adaptive (DNS, IRGAN, AdvIR, and NMRN), and KG-based (RWS) samplers:

- **RNS** [23]: Such **r**andom **n**egative **s**ampling (RNS) is one prevalent technique to sample negative items with uniform probability. For a fair comparison, we use MF as the recommender.
- **PNS** [3, 4]: The MF recommender is equipped with the **p**opularity-biased **n**egative **s**ampling (PNS).
- **DNS** [39]: This is a state-of-the-art sampling strategy, dynamic negative sampling (DNS), which adaptively picks the negative item scored highest by the current MF recommender among some random missing samples. DNS is known as one of the most effective sampler for BPR loss, and empirically outperforms [22].
- **IRGAN** [30]: Such model is an adversarial sampler, which conducts a minimax game between the recommender and the sampler towards selecting better negative items.
- **AdvIR** [21]: This is a state-of-the-art sampler, which exploits both adversarial sampling and training (*i.e.,* adding perturbation) to generate negatives. In particular, the recommender and the sampler use the identical MF models.
- **NMRN** [31]: This sampler uses the translation-based CF models as the interaction and reward functions in the recommender and sampler, to adversarially sample negatives.
- **RWS** [18]: This **r**andom **w**alk **s**ampling (RWS) depends on the topology of KG merely to select negative items to assist MF.

4.2.2 ***KG-enhanced Recommender Methods.*** As KGPolicy presents a new way that uses KG in the sampler, we select the state-of-the-art KG-based recommenders as the competitors, ranging from supervised learning-based (NFM), regularization-based (CKE), path-based (RippleNet) to GNN-based (KGAT):

- **NFM** [11]: This recommender factorizes historical behaviors and item knowledge as the representations of a user-item interaction pair and feeds into a neural network to conduct predictions.
- **CKE** [38]: Such recommender uses KG embeddings to enhance item representations and further assist MF.
- **RippleNet** [27]: Such model leverages multi-hop paths rooted at each user in KG to enrich their representations, and employs MF on representations to do the predictions.
- **KGAT** [32]: This is a state-of-the-art KG-based recommender, which employs GNN on KG to generate representations of users and items and use inner product to do predictions.

4.3 Experimental Settings

4.3.1 ***Evaluation Metrics.*** We use two widely-used metrics of top-K recommendation and preference ranking tasks [12, 32]: recall@K and ndcg@K. By default, we set K to be 20. For each user in the test sets, we view the items she has adopted as positive items, and evaluate how well the recommenders rank the positive items over the whole item space. We report the average metrics of all users in each test set.

4.3.2 ***Parameter Settings.*** We implement our KGPolicy model in Pytorch and release our code and datasets at https://github.com/xiangwang1223/kgpolicy. As the datasets, data splits, evaluation metrics, and KG-based baselines (*cf.* Section 4.2.2) are exactly the same as that used in KGAT [32], we hence directly copy their performance from the original paper [32]. As for KGPolicy and the negative sampling baselines (*cf.* Section 4.2.1), we fix the embedding size for all recommenders and samplers as 64, and set the optimizer as Adm [16]. We use Xavier [8] to initialize sampler parameters; meanwhile, as suggested in [21, 30], a trained MF with RNS is used to initialize the recommenders paired with IRGAN, AdvIR, and KGPolicy, so as to stabilize and speed up the model training. For hyperparameters, we conduct a grid search to find the optimal settings for each model: the learning rates for the recommender and the sampler are searched in $\{0.0001, 0.0005, 0.001, 0.005\}$, and the coefficients of L_2 regularization in the recommender is tuned in $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$.

Other hyperparameters of KGPolicy are set as follows: we apply two graph convolutional layers to perform graph representation learning, *i.e.,* $L = 2$ in Equation (9); we search the number of exploration operations, T, in $\{1, 2, 3, 4\}$ and report its effect in Section 4.5.1; moreover, the size of pruned neighbors n_2 is searched in $\{4, 8, 16, 32, 64\}$ where n_1 is fixed as 64. Without specification, we set T as 2 by default.

4.4 Performance Comparison (RQ1)

We first report the empirical results *w.r.t.* negative sampling, and the analyze the comparison *w.r.t.* the usage of KG.

4.4.1 ***Empirical Results w.r.t. Negative Sampling.*** Through analyzing the overall results summarized in Table 2, we have the following observations:

- Our proposed KGPolicy consistently outperform all baselines across three datasets in all measures. In particular, KGPolicy achieves remarkable improvements over the strongest baselines *w.r.t.* ndcg@20 by 9.77%, 6.59%, and 5.42% in Yelp2018, Last-FM, and Amazon-Book, respectively. We attribute such improvements to the following aspects — 1) by exploiting the rich facts in KG, KGPolicy is more likely to discover high-quality negative items, than the static and adaptive samplers that are guided by limited information; 2) such discovered negatives are close to the target positive interactions, so as to offer meaningful gradients for recommender learning (evidence from the better capacity and representation ability of MF); and 3) by taking advantage of the KG structure and the neighbor pruning strategy, KGPolicy is able to effectively narrow the large-scale search space down to potential items.
- By jointly analyzing the results across the three datasets, we find that the improvement of KGPolicy on Yelp2018 is the most significant, while that in Amazon-book is the least. This may be caused by the quality of knowledge, since KG in Yelp2018 is constructed by using the local business information and hence is more accurate and targeted as compared to the others.
- Static samplers (*i.e.,* RNS and PNS) make MF perform poor on three datasets. In addition, PNS achieves worse performance than that of RNS in Yelp2018. Such findings are consistent to prior studies [22], suggesting that the samples from uniform or popularity-biased distribution, are easily discriminated by the recommender, and contribute varnishy (close-to-zero) gradients to update the recommender parameters. This also emphasizes the great need in informative negative instances.
- Compared with that of RNS and PNS, the results of adaptive samplers (*i.e.,* DNS, IRGAN, AdvIR, and NMRN) verify the importance of adaptively selecting negatives. They make the sampling distributions dependent on recommender status and conditioned on the target user.
- DNS works well on the three datasets. This finding is consistent with previous works [7]. One possible reason is that DNS could effectively reduce the search space via the ranking-aware reject sampling mechanism [39], suggesting the positive effect of suitable pruning strategy.
- AdvIR and NMRN substantially outperform IRGAN in most cases. It is reasonable since the key-value attention network and the adversarial perturbations are separately involved in NMRN and AdvIR, which endow the recommender better representation and generalization abilities, respectively. This suggests that, the capacity of sampler have impact on the recommender.
- While leveraging the identical data to KGPolicy, RWS only achieves comparable performance to the static samplers. It makes sense since the paths generated by random walk usually are biased by the popularity of nodes. Moreover, such sampling

Table 2: Comparison with Negative Samplers.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
RNS	0.0465	0.0575	0.0661	0.1063	0.1153	0.0754
PNS	0.0166	0.0220	0.0668	0.0984	0.1127	0.0730
DNS	**0.0687**	**0.0839**	**0.0877**	**0.1381**	**0.1518**	**0.1033**
IRGAN	0.0628	0.0767	0.0642	0.1070	0.1253	0.0833
AdvIR	0.0590	0.0744	0.0810	0.1304	0.1427	0.0967
NMRN	0.0565	0.0691	0.0719	0.1151	0.1305	0.0875
RWS	0.0488	0.0611	0.0667	0.1076	0.1185	0.0760
KGPolicy	**0.0747***	**0.0921***	**0.0932***	**0.1472***	**0.1572***	**0.1089***
%Improv.	8.73%	9.77%	6.27%	6.59%	3.56%	5.42%

Table 3: Comparison with KG-based Recommenders.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
NFM	0.0660	0.0810	0.0829	0.1213	0.1366	0.0914
CKE	0.0657	0.0805	0.0736	0.1184	0.1344	0.0885
RippleNet	0.0664	0.0822	0.0791	0.1238	0.1336	0.0910
KGAT	**0.0712**	**0.0867**	**0.0870**	**0.1325**	**0.1489**	**0.1006**
KGPolicy	**0.0747***	**0.0921***	**0.0932***	**0.1472***	**0.1572***	**0.1089***
%Improv.	4.92%	6.22%	7.12%	11.09%	5.60%	8.25%

also remains unchanged with the recommender status. It again justifies that KGPolicy better leverages KG.

4.4.2 ***Empirical Results w.r.t. KG usage.*** Table 3 shows the performance comparison *w.r.t.* the usage of KG. We have the following findings:

- Clearly, we can observe the significant improvements brought from KGPolicy on all three datasets *w.r.t.* all evaluation metrics. For example, KGPolicy outperforms the strongest baseline, KAGT, *w.r.t.* ndcg@20 by 6.22%, 11.09%, and 8.25% on Yelp2018, Last-FM, and Amazon-book, respectively. This again validates the rationality of using KG in the sampling method, and verifies our hypothesis that KG can provide guiding signals towards high-quality negative items.
- Knowledge-reinforced negatives endows the recommender better representation ability. Specifically, with th except of NFM, all KG-based recommenders user inner product of user and item representations to do predictions; hence the representation ability directly determines how well the recommender perform. Compared with complex representation learning models (*e.g.,* path-based in RippleNet and GNN-based in KGAT), KGPolicy uses simple ID embeddings but achieves the best performance. This suggests that, using proper negative signals helps improve representation ability.
- Existing methods are mainly focusing on using KG to leverage positive signals — either enhancing semantic similarity among items (*i.e.,* CKE), propagating user preference (*i.e.,* RippleNet), or encoding high-order connectivity between users and items (*i.e.,* KGAT), while ignoring its potential for distilling negative signals. The superior performance of KGPolicy emphasizes that, working together with behavior differences of users, KG is beneficial to negative sampling.

4.5 Study of KGPolicy (RQ2)

We also perform ablation studies to get deep insights on KGPolicy. We start by exploring the influence of different KG components — user behaviors and item knowledge. We then investigate how the number of exploration operations affects the performance. In what follows, we also analyze the influence of reward functions.

4.5.1 ***Impact of Exploration Number.*** As the core of KGPolicy is the exploration operation, we hence investigate how the number of such operations affects the performance. In particular, we search the operation number, T, in the range of $\{1, 2, 3, 4\}$. Table 4 summarizes the experimental results, wherein KGPolicy-3 indicates the sampler trained with three exploration operations, similar notations for others. There are several interesting observations:

Table 4: Impact of exploration operation numbers.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
KGPolicy-1	0.0738	0.0878	0.0891	0.1409	0.1520	0.1053
KGPolicy-2	**0.0747***	**0.0921***	**0.0932***	**0.1472***	**0.1572***	**0.1089***
KGPolicy-3	0.0730	0.0879	0.0928	0.1450	0.1551	0.1076
KGPolicy-4	0.0729	0.0878	0.0919	0.1437	0.1546	0.1059

Table 5: Impacts of reward functions.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
S-Reward	0.0731	0.0865	0.0899	0.1411	0.1559	0.1071
P-Reward	0.0721	0.0859	0.0918	0.1443	0.1558	0.1068

- Increasing the number of exploration operations enhances the predictive results. Clearly, KGPolicy-3 outperforms KGPolicy-1 and KGPolicy-2 in most cases. We attribute such consistent improvements to the diversity of negative items: three-hop item neighbors derived from KGPolicy-3 naturally cover more items — beyond these exposed but less interested, and probably involving some unobserved and disliked negatives — than the one- and two-hop neighbors from KGPolicy-1 and KGPolicy-2.
- Continuing one more exploration beyond KGPolicy-3, we observe that KGPolicy-4 makes the performance worse across the board. This might be because conducting too many operations would introduce less-relevant items and result in vanishing gradients to the recommender training.
- Jointly comparing results across Tables 2 and 4, KGPolicy with varying exploration operations is superior to other methods consistently. This again empirically shows the rationality and effectiveness of knowledge-aware negative sampling.

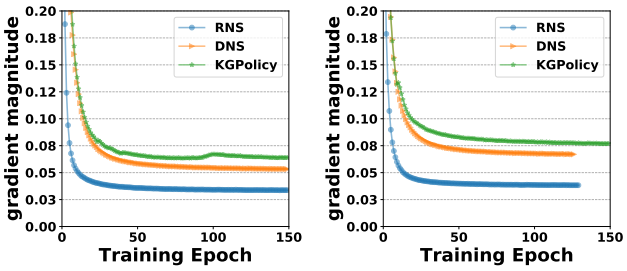
4.5.2 ***Impact of Reward Functions.*** To verify the influence of reward functions, we do ablation study by considering two variants of KGPolicy. In particular, we employ the prediction reward function (*cf.* Equation (7)) to build the variant P-Reward, while using the similarity reward function merely to construct the variant P-Reward. The results of comparison are shown in Table 5.

Only using S+Reward or P+Reward degrades the performance, indicating that they make the selected items suboptimal as negative. There is no exact winner between S-Reward and P-Reward on all datasets. To be more specific, S-Reward performs better in Yelp2018, while outperforming P-Reward in Last-FM. Such slight differences might be caused by the datasets. Therefore, it validates the rationality of our design.

4.6 In-depth Analysis (RQ3)

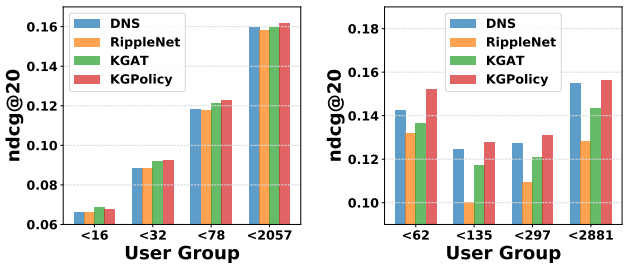
In this section, we get deep insights on how the knowledge-aware negative sampling facilitates the recommender learning. Towards the in-depth analysis, we perform additional experiments on the following aspects — training process *w.r.t.* gradient magnitude, and recommendation performance *w.r.t.* sparsity levels. In what follows, we present a case study to illustrate the quality of negative samples.

4.6.1 ***Training Process w.r.t. Gradient Magnitude.*** To study how informative the negative items are, we select RNS, DNS, and KGPolicy to represent the static, adaptive, and knowledge-aware sampling approaches, and use the average gradient magnitude in Equation (4) as the evaluation metric. We record the status of gradient magnitude at each epoch and illustrate the learning curves



(a) gradient magnitude in Yelp2018 (b) gradient magnitude in Last-FM

Figure 3: Training process *w.r.t.* average gradient magnitude.



(a) ndcg on Yelp2018 (b) ndcg on Last-FM

Figure 4: Performance comparison *w.r.t.* sparsity levels.

on Yelp2018 and Last-FM datasets in Figures 3. Here we omit the similar result in Amazon due to the limited space.

- Clearly, knowledge-aware sampling (KGPolicy) achieves larger gradient magnitudes than the static and adaptive strategies throughout all epochs. This qualitative results, together with the performance in Table 2, indicates that the negative items sampled by KGPolicy is more informative than that by RNS and DNS. This again verifies the rationality and effectiveness of incorporating KG into sampling.
- The gradient magnitudes of DNS are larger than that of RNS, which is consistent to the observation in [22]. In particular, the uniform sampler easily results in low-quality negative samples, making the gradients varnishing. The suboptimal performance of adaptive sampler emphasizes the importance of knowledge-aware guiding signals.

4.6.2 ***Performance w.r.t. Sparsity Levels.*** Inspired by early works [32, 38] which investigates whether KG helps to alleviate the sparsity issue, we would like to track KGPolicy's contributions on such issue. Note that, the user groups of different sparsity levels are exactly the same as that reported in KGAT, where users are divided into four groups based on the interaction number per user (*e.g.,* separately less than 62, 135, 297, and 2881 in Last-FM). We select the strongest baselines *w.r.t.* negative sampling (DNS) and KG-based recommendation (RippleNet and KGAT). Figures 4(a) and 4(b) show their performance *w.r.t.* ndcg@20 on Yelp2018 and Last-FM, respectively.

- Clearly, KGPolicy shows significant improvements over all competing methods in the third and fourth user groups. To be more specific, in Last-FM, the improvements over KGAT are 8.78% and 8.50% for <135 and <297 user groups, respectively. This promising finding again verifies the significance of high-quality negative signals, which can facilitate the simple linear interaction function (*i.e.,* MF) to achieve comparable or even better performance than the complex and nonlinear interaction modeling (*e.g.,* RippleNet and KGAT).
- KGPolicy, however, slightly outperforms KGAT in the sparsest group in Yelp2018, where users hold less than 16 interactions. We hence conclude that, knowledge-aware negative sampling is beneficial to the relatively active users, whereas extremely sparse behaviors are not sufficient to guide the sampler learning.

In a nutshell, jointly considering informative propagation over KG and knowledge-aware negative sampling might be a promising solution to sparsity issues. We leave such exploration for future.

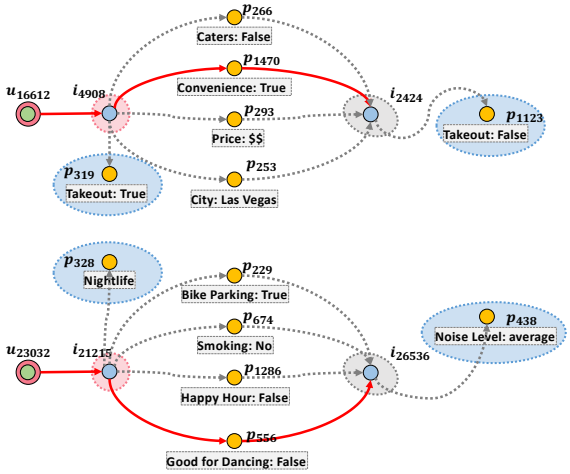


Figure 5: Real cases of exploring paths in Yelp2018.

4.6.3 ***Case Study.*** Exploring paths from the positive interaction to the negative item could be treated as the support evidence why a user holds opposite opinions on two items. We randomly selected two positive interactions, (u_{16612}, i_{4908}) and (u_{23032}, i_{21215}), and illustrate exploring paths derived from KGPolicy-1 in Figure 5. We have the following findings:

- KGPolicy is able to generate exploring paths for each positive interaction, such as $i_{4908} \rightarrow p_{1470} \rightarrow i_{2424}$ for u_{16612} and $i_{21215} \rightarrow p_{1286} \rightarrow i_{26536}$ for u_{23032}. Such exploring paths play a crucial role in negative sampling.
- Taking advantages of the designed neighbor attention module, KGPolicy successfully captures personal interest for each user, specifying varying importance of KG entities. There are many shared neighbors between the positive and negative items. For example, u_{16612} cares more about whether visiting a restaurant is convenient or not, rather than other general attributes; while, u_{23032} prefers the places with the KG entity p_{556}, *Good for Dancing: False*. As such, KGPolicy can narrow the searching space down to a set of candidates meeting personal tastes.
- Furthermore, the exploring paths, together with entity differences between positive and negative items, provides us with some insights of user real tastes. For example, when being aware of the overlapping attributes (*e.g.,* p_{266}, p_{1470}, p_{293}, p_{253}), u_{16612} visited i_{4908}, but ignore i_{2424}. Analyzing the attribute differences, we find that i_{4908} allows *TakeOut* (*cf.* p_{319}), which is disallowed in i_{2424}. Analogously, exclusive KG entities of i_{21215} are highly likely to make u_{23032} ignores the alternative i_{26536}. Therefore, KGPolicy

can help interpret why a user consumes the positive items but did not adopt the negatives.
- These observations also inspires us to involve some true negative signals, towards better explanations on user behaviors. We would like to perform user experience modeling in future work.

5 RELATED WORK

This work is relevant to two research lines: negative sampling and knowledge-graph based recommendation.

5.1 Negative Sampling for Recommendation

Towards solving one-class problem in implicit feedback, earlier recommendation methods [3, 5, 7, 20, 23] use negative sampling to subsample some from unobserved items as negatives, based on the predefined sampling distributions. For example, random (RNS) [7, 23] and popularity-biased (PNS) [3, 4] sampling are based on the uniform and popularity distribution, respectively. While being prevalent, such static sampling is independent of model status and unchanged for different users, thereby easily low-quality negative items and contributing little to recommender training. Later on, adaptive sampling, such as DNS [39], LambdaFM [37], and Adaptive Oversampling [22], is proposed. The basic idea is to devise additional measures to select hard negative items, accounting for model status. For example, DNS picks the item with the highest prediction scored by the current recommender as a negative sample. Recent studies [21, 30, 31] get inspiration from generative adversarial learning [9] to generate adversarial samples. For example, IRGAN [30] and NMRN [31] introduce a sampler model to play a minimax game with the recommender, so as to select close-to-observed negatives. While being effective from the perspective of numerical optimization, these hard negative samples are highly likely to be true positive in future testing data. Hence, more informative guiding signals are required to discover true negative feedback. Some recent efforts [6, 7, 19] consider extra behavior data to enhance the negative sampler. For instance, RNS [7] and multi-channel feedback BPR [19] exploit viewed but non-clicked and clicked but non-purchased data to help filter negatives. However, the scale of such behaviors is limited compared with the vast amount of missing data, which is insufficient to effectively distill negative signals.

Distinct from these methods, we hypothesize that knowledge graph (KG) of user behaviors and item knowledge is useful to infer informative and factual negative items from missing data.

5.2 Knowledge Graph-based Recommendation

KG-based recommendation has attracted increasing attention. At its core is the KG-enhanced interaction modeling, which use the structural knowledge to enrich relations among users and items in the recommender. A research line [11, 15, 28, 38] uses KG embeddings to improve the quality of item representations. For example, CKE [38] exploits KG embeddings to enhance MF, while DKN [28] treat semantic embeddings from KG as the content information of items. Another line extracts paths and meta paths [14, 24, 35] that connect the target user and item via KG entities, and then build the predictive models upon them. More recently, PGPR [36] further exploits a RL approach to explore items of interest for a target user. Another type of methods unify the foregoing ideas to encode first-order [1, 2] and higher-order connectivity [29, 32] of KG into the representations of users and items. For example, KTUP [2] and CFKG [1] jointly train the recommendation task with the KG completion problem, such that first-order connectivity involved in KG triples and user-item pairs can be captured via translation principles. Taking advantages of information propagation proposed by GNNs [17, 26], KGAT [32] and KGNN-LS [29] perform embedding propagation by applying multiple GNN layers over KG, so as to directly inject high-order connectivity into the representations.

However, to the best of our knowledge, existing recommenders only leverage KG to design more complex interaction functions and essentially distill better positive signals, but leave the negative signals unexplored. Our work differs from them in that, we focus on knowledge-aware negative sampling, towards discovering informative and factual negative feedback from missing data.

6 CONCLUSION AND FUTURE WORK

In this work, we initiated an attempt to incorporate knowledge graph into negative sampling to discover high-quality negative signals from implicit feedback. We devised a new knowledge-aware sampling framework, KGPolicy, which works as a reinforcement learning agent and effectively learns to navigate towards potential negative items for a positive user-item interaction. At its core is the proposed exploration operation, which is capable of adaptively selecting next neighbors to visit, accounting for user behavior- and item knowledge-based negative signals. Extensive experiments on three benchmark datasets demonstrate the rationality and effectiveness of knowledge-aware sampling. In particular, when equipped with KGPolicy, MF, such a simple and linear model exhibits significant improvements over both state-of-the-art sampler and KG-based recommender models.

Analysis on how knowledge graph facilitates the recommender learning provides us with some insights of the sampling process. Distilling high-quality negative signals from unobserved data is of crucial importance to make the use of positive-only data effectively. It greatly helps to establish better representations of users and items with limited data. Moreover, such guiding signals sort of interpret user intents and devise shaper preference distributions. This work hence opens up new research possibilities. In future work, we would like to involve more auxiliary information, such as social network and contextual information (*e.g.,* geo location and timestamp), together with user behaviors, to better mine negative user feedback. Furthermore, we plan to perform experiments on user experience modeling, with target of establishing ground truth on what a user likes and dislikes, as well as user-generated explanations. Exploiting such interpretable and explicit negative signals is beneficial to explainable recommendation.

ACKNOWLEDGMENTS

This research is part of NExT++ research, which is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative, and also supported by the National Natural Science Foundation of China (61972372, U19A2079).

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137.

[2] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. 151–161.

[3] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. 2018. Word2vec applied to recommendation: hyperparameters matter. In *RecSys*. 352–356.

[4] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On Sampling Strategies for Neural Network-based Collaborative Filtering. In *KDD*. 767–776.

[5] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan S. Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. In *WWW*. 639–648.

[6] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An Improved Sampler for Bayesian Personalized Ranking by Leveraging View Data. In *WWW*. 13–14.

[7] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*. 2230–2236.

[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NeurIPS*. 2672–2680.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1025–1035.

[11] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[13] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR*. 549–558.

[14] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. In *SIGKDD*. 1531–1540.

[15] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. 505–514.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[18] Jing Li, Feng Xia, Wei Wang, Zhen Chen, Nana Yaw Asabere, and Huizhen Jiang. 2014. ACRec: a co-authorship based random walk model for academic collaboration recommendation. In *WWW*. 1209–1214.

[19] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian Personalized Ranking with Multi-Channel User Feedback. In *RecSys*. 361–364.

[20] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *ICDM*. 502–511.

[21] Dae Hoon Park and Yi Chang. 2019. Adversarial Sampling and Training for Semi-Supervised Information Retrieval. In *WWW*. 1443–1453.

[22] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*. 273–282.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[24] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*. 297–305.

[25] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NeurIPS*. 1057–1063.

[26] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[27] 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 *CIKM*. 417–426.

[28] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *WWW*. 1835–1844.

[29] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *KDD*. 968–977.

[30] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*. 515–524.

[31] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural Memory Streaming Recommender Networks with Adversarial Training. In *KDD*. 2467–2475.

[32] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. 950–958.

[33] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item Silk Road: Recommending Items from Information Domains to Social Users. In *SIGIR*. 185–194.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[35] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*.

[36] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR*. 285–294.

[37] Fajie Yuan, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. LambdaFM: Learning Optimal Ranking with Factorization Machines Using Lambda Surrogates. In *CIKM*. 227–236.

[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.

[39] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR*. 785–788.

[40] Wayne Xin Zhao, Gaole He, Hong-Jian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2018. KB4Rec: A Dataset for Linking Knowledge Bases with Recommender Systems. *CoRR* abs/1807.11141 (2018).