Reinforcement Knowledge Graph Reasoning for Explainable Recommendation

Tuo Liu

August 7, 2022

1 Main Contribution

The authors proposed a method called Policy-Guided Path Reasoning (PGPR), which couples recommendation and interpretability by providing actual paths in a knowledge graph. And their contributions include four aspects:

1. They highlight the significance of incorporating knowledge graphs into recommendation to formally define and interpret the reasoning process.
2. They propose a reinforcement learning (RL) approach featuring an innovative soft reward strategy, user-conditional action pruning and a multi-hop scoring function.
3. They design a policy-guided graph search algorithm to efficiently and effectively sample reasoning paths for recommendation.
4. They extensively evaluate their method on several large-scale real-world benchmark datasets, obtaining favorable results compared with state-of-the-art methods.

2 Methodology

2.1 Problem Formulation

Given a knowledge graph G_R, user $u \in U$ and integers K and N , the goal is to find a recommendation set of items $\{i_n\}_{n\in[N]} \subseteq I$ such that each pair (u, i_n) is associated with one reasoning path $p_k(u, i_n)(2 \leq k \leq K)$, and N is the number of recommendations.

2.2 Formulation as Markov Decision Process

The start of the method is to formalize the KGRE-Rec problem as a Markov Decision Process. Details are as follows:

1. State: the state s_t at step t is defined as a tuple (u, e_t, h_t), where $u \in U$ is the starting user entity, e_t is the entity the agent has reached at step t, and h_t is the history prior to step t.
2. Action: the complete action space A_t of state s_t is defined as all possible outgoing edges of entity e_t excluding history entities and relations. Formally, $A_t = \{(r, e)|(e_t, r, e) \in G_R, e \notin \{e_0, \cdots, e_{t-1}\}\}$. And to solve the problem of the large out-degrees, the authors introduce the user-conditional action pruning strategy.
3. Reward: given any user, there is no pre-known targeted item in the KGRE-Rec problem, so it is unfeasible to consider using binary rewards. Instead, the authors consider to give a soft reward only for the terminal state $s_T = (u, e_T, h_T)$ based on another scoring function $f(u, i)$. The terminal reward R_T is defined as

$$R_T = \begin{cases} \max\left(0, \frac{f(u,e_T)}{\max_{i\in I} f(u,i)}\right), & \text{if } e_T \in \mathcal{I} \\ 0, & \text{otherwise} \end{cases}$$

   where the value of R_T is normalized to the range of [0,1].

4. Transition: given a state $st = (u, e_t, h_t)$ and an action $at = (r_{t+1}, e_{t+1})$, the transition to the next state s_{t+1} is:

$$P\left[s_{t+1} = (u, e_{t+1}, h_{t+1}) \mid s_t = (u, e_t, h_t), a_t = (r_{t+1}, e_{t+1})\right] = 1$$

The initial state is an exception. For simplicity, the authors assume the prior distribution of users follows a uniform distribution so that each user is equally sampled at the beginning.

5. Optimization: the goal of the MDP is to learn a stochastic policy π that maximizes the expected cumulative reward for any initial user u:

$$J(\theta) = E_\pi \left[\sum_{t=0}^{T-1} \gamma^t R_{t+1} \mid s_0 = (u, u,) \right]$$

The authors solve the problem through REINFORCE by designing a policy network and a value network that share the same feature layers. The structures of the two networks are as follows:

$$\begin{aligned} \mathbf{x} &= \text{dropout}\left(\sigma\left(\text{dropout}\left(\sigma\left(\mathbf{s}\mathbf{W}_1\right)\right)\mathbf{W}_2\right)\right) \\ \pi\left(\cdot \mid \mathbf{s}, \tilde{\mathbf{A}}_u\right) &= \text{softmax}\left(\tilde{\mathbf{A}}_u \odot \left(\mathbf{x}\mathbf{W}_\text{p}\right)\right) \\ \hat{v}(\mathbf{s}) &= \mathbf{x}\mathbf{W}\mathbf{v}_\mathbf{v} \end{aligned}$$

where x is the learned hidden features of the state, the element of A_u is to be 0 or 1(invalid action for 0 and valid for 1), and the $\odot$ means multiplying the corresponding elements, which can mask invalid actions. Finally, the policy gradient $\nabla_\Theta J(\Theta)$ is defined as:

$$\nabla_\Theta J(\Theta) = E_\pi \left[\nabla_\Theta \log \pi_\Theta \left(\cdot \mid \mathbf{s}, \tilde{\mathbf{A}}_u\right) \left(G - \hat{v}(\mathbf{s})\right) \right]$$

2.3 Multi-Hop Scoring Function

Define a general multi-hop scoring function $f\left(e_0, e_k \mid \tilde{r}_{k,j}\right)$ of two entities e_0, e_k given 1-reverse k-hop pattern $\tilde{r}_{k,j}$ as follows.

$$f\left(e_0, e_k \mid \tilde{r}_{k,j}\right) = \left\langle \mathbf{e}_0 + \sum_{s=1}^{j} \mathbf{r}_\text{s}, \mathbf{e}_\mathbf{k} + \sum_{s=j+1}^{k} \mathbf{r}_\text{s} \right\rangle + b_{e_k}$$

Then, the scoring function for action pruning is defined as

$$f((r, e) \mid u) = f\left(u, e \mid \tilde{r}_{k_e,j}\right)$$

where k_e is the smallest k such that $\tilde{r}_{k,j}$ is a valid pattern for entities (u, e). And the scoring function for reward is defined as

$$f(u, i) = f\left(u, i \mid \tilde{r}_{1,1}\right)$$

because the authors simply use the 1-hop pattern between user entity and item entity. As for learning the scoring function, the goal is to maximize the conditional probability of $P\left(e' \mid e, \tilde{r}_{k,j}\right)$, which is defined as

$$P\left(e' \mid e, \tilde{r}_{k,j}\right) = \frac{\exp\left(f\left(e, e' \mid \tilde{r}_{k,j}\right)\right)}{\sum_{e'' \in \mathcal{E}} \exp\left(f\left(e, e'' \mid \tilde{r}_{k,j}\right)\right)}$$

However, due to the huge size of the entity set E, the authors adopt a negative sampling technique to approximate $\log P\left(e' \mid e, \tilde{r}_{k,j}\right)$. In the end, the goal is to maximize the objective function $J\left(G_\text{R}\right)$, defined as:

$$J\left(G_\text{R}\right) = \sum_{e, e' \in \mathcal{E}} \sum_{k=1}^{K} 1\left\{\left(e, \tilde{r}_{k,j}, e'\right)\right\} \log P\left(e' \mid e, \tilde{r}_{k,j}\right)$$

where $1\left\{\left(e, \tilde{r}_{k,j}, e'\right)\right\}$ is 1 if $\tilde{r}_{k,j}$ is a valid pattern for entities (e, e') and 0 otherwise.

2.4 Policy-Guided Path Reasoning

The authors propose to employ beam search guided by the action probability and reward to explore the candidate paths as well as the recommended items for each user. It takes as input the given user u, the policy network $\pi\left(\cdot \mid \mathbf{s}, \tilde{\mathbf{A}}_u\right)$, horizon T, and predefined sampling sizes at each step, denoted by $K1, ..., K_T$. As output, it delivers a candidate set of $T-hop$ paths P_T for the user with corresponding path generative probabilities Q_T and path rewards R_T.

3 Experiments

All experiments are conducted on the Amazon e-commerce datasets collection, consisting of product reviews and meta information from Amazon.com. The datasets include four categories: CDs and Vinyl, Clothing, Cell Phones and Beauty. The authors adopt TF-IDF to eliminate less salient features in the preprocessing stage.

The authors use Normalized Discounted Cumulative Gain (NDCG), Recall, Hit Ratio (HR) and Precision (Prec.) as measures. Experiments conducted on the datasets show that PGPR can achieve better results compared to BPR, BPR-HFT, VBPR, TransRec, DeepCoNN, CKE and JRL.

4 Future Work

1. PGPR approach is a flexible graph reasoning framework and can be extended to many other graph-based tasks such as product search and social recommendation.
2. PGPR approach can also be extended to model time-evolving graphs so as to provide dynamic decision support.

5 My View

This paper is really novel in using reinforcement learning method to do the recommendation. Specifically, an agent starts from a given user, and learns to navigate to the potential items of interest, such that the path history can serve as a genuine explanation for why the item is recommended to the user.

Reinforcement Learning is really hot in recent years, and considering the goal of explainable recommendation is to find a path from user to item on the knowledge graph, we can skillfully combine these two things together, by using the agent as a navigator.