NEURAL GRAPH REASONING FOR EXPLAINABLE DECISION-MAKING

By

YIKUN XIAN

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

S. Muthukrishnan

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October 2021

ABSTRACT OF THE DISSERTATION

Neural Graph Reasoning for Explainable Decision-Making

by **YIKUN XIAN**

Dissertation Director:

S. Muthukrishnan

Researchers have been seeking to develop intelligent systems with the ability to behave like humans by autonomously making accurate and reasonable decisions for real-world tasks. It now becomes imaginable and achievable with the help of advanced artificial intelligence (AI), especially the deep learning technique that is known for its superior representation and predictive power. Such deep learning based decision-making systems have been shown to be surprisingly effective in delivering accurate predictions, but at the price of lack of explainability due to the "black-box" of deep neural networks. However, explainability plays a pivotal role in practical human-involved applications such as user modeling, digital marketing and e-commerce platforms. Explanations can be leveraged to not only assist model developers to understand and debug the working mechanism of the decision-making process, but also facilitate better engagement and trustworthiness for the end users who consume the results produced by the systems.

In this thesis, we concentrate on one category of explainable decision-making system that relies on external heterogeneous graphs to generate accurate predictions accompanied with faithful and comprehensible explanations, which is also known as the neural graph reasoning for explainable decision-making. Unlike existing work on explainable machine

learning that mainly yields model-agnostic explanations for deep neural networks, we attempt to develop intrinsically interpretable models based on graphs with the guarantee of both accuracy and explainability. The meaningful and versatile graph structures (e.g., knowledge graphs) are shown to be effective in improving model performance, and more importantly, make it possible for an intelligent decision-making system to conduct explicit reasoning over graphs to generate predictions. The benefit is that the resulting graph paths can be directly regarded as the explanations to the prediction results because the traceable facts along the paths reflect the decision-making process and can also be easily understood by humans.

To this end, our goal is to develop neural graph reasoning approaches to generate such path-based explainable results by marrying the merits of predictive power by deep neural models and the interpretability of graph structures. Specifically, we propose four methods from different perspectives: (i) a fundamental graph reasoning framework based on reinforcement learning, (ii) a neural-symbolic model featured by its self-explaining and compositional neural symbolic modules, (iii) a neural logic model that explicitly learns personalized and explainable reasoning rules, and (iv) an imitation learning based method that learns to distinguish the quality of explainable paths from demonstrations. These approaches are extensively evaluated on real-world benchmarks across different applications such as e-commerce recommendation and column annotation in digital marketing. The experimental results demonstrate the effectiveness of the proposed methods in achieving satisfying prediction accuracy and providing users with faithful and understandable path-based explanations.

ACKNOWLEDGMENTS

First of all, I would like to express my truthful, accumulated and deep-seated gratitude to my advisor Prof. Shan Muthukrishnan for his academic, mental, financial, supports during my PhD study. He has been encouraging and motivating me to explore and work on any research topics I am interested in. I am still impressed by his wisdom, kindness and open-mindedness when he guided me to discover, think of and formulate my research problems. Without his guidance, it would be impossible for me to complete Ph.D and this dissertation.

I want to thank Prof. Yongfeng Zhang, Prof. Gerard de Melo and Dr. Lihong Li for serving on my defense committee and providing me with insightful feedback on my thesis work and presentation, and thank Prof. Pranjal Awasthi and Prof. Amélie Marian for serving on my qualifying committee and providing me with insightful feedback on my research.

I would like to give special thanks to Prof. Yongfeng Zhang who provided me with valuable guidance for my first first-author paper in top conferences and supported me to get involved in academic events and build connection with industry. It was a pleasure to work with him during the past few years. I would like to thank Prof. Abdeslam Bolarias. As my first-year adviser at Rutgers, he introduced me to the area of deep reinforcement learning and provided valuable guidance at my early stage of PhD research. I also would like to thank Prof. William Steiger for his kindness and support in my first year at Rutgers. I would also give my gratitude to all the school staff who helped me during my study at Rutgers.

I feel honored and thankful to work with all my talented co-authors, including Zuohui Fu who is the chief collaborator of my recent work, Qiaoying Huang, Yingqiang Ge, Zhou Qin, Ruoyuan Gao, Shijie Geng, Xu Chen, Yaxin Zhu, Hoa T. Vu, Tak Yeon Lee, Sungchul Kim, Ryan Rossi, Branislav Kveton, Zheng Wen, Saied Mahdian, Christos

Faloutsos, George Karypis, Luna Dong, Jin Li, Jun Ma, Andrey Kan. I also would like to thank all my intern mentors: Qiang Ma and Ming-Jun Chen from Google, Handong Zhao from Adobe Research and Tong Zhao from Amazon. They provided me with professional mentorship, valuable and unique advice and research guidance during my internships. The experience of working with these talents also contributes to the completion of this thesis.

I also thank my friends for sharing my long journey in Rutgers including but not limited to Zhihan Fang, Xiaoyang Xie, Liyang Wang, Guang Wang, Zhaomeng Niu, Yu Yang, Shuhan Wang, Jie Wang, Yizhe Zhu, Ji Zhang, Kun Wang, Guangzhi Tang, Chaolun Xia, Priya Govindan and many others.

Finally, I want to give my utmost thanks to all my family members for their unconditional support across these years.

DEDICATION

To my family, especially my parents and grandparents who encouraged and supported me over the duration of my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview of Explainable Decision Making

The last decade has witnessed huge success of deep learning techniques that significantly push the boundaries of real-world decision-making systems in delivering more accurate predictions. Such deep neural models are known to be powerful at feature representation learning from large amount of data with the complicated and non-linear operations and underlying structures. The achieved performance gain brings tremendous benefits to various applications in practice. For instance, modern recommender systems [14, 144, 39] tend to learn user and item representations via deep neural networks to capture latent relationships of user-item interactions and make increasingly accurate recommendations to improve user engagement. Data management systems [76, 99, 10, 53] powered by deep neural models are capable of automating the data cleaning process (e.g., annotating table columns and linking entities among rows) and largely saving time and effort for humans in labeling and evaluation. Other applications such as advertising [39, 64, 65], user profiling [25, 101], social networking [140, 75, 74], conversational systems [31, 30] and medical diagnosis [51, 50] are also benefited from the powerfulness of deep neural networks.

However, due to the blackbox nature of deep neural networks, such neural decision-making systems may lack explainability and transparency, leading to negative consequences in many human-centered scenarios [37, 88, 138]. For instance, if a medical-diagnosis system fails to provide supportive evidence to justify why the prediction is correct or not, the doctor can hardly adopt the automated decision even if its actual accuracy is high. Same situation can be applied to other domains such as finance, e-commerce, digital marketing, etc. Therefore, explanations are critical and useful in modern accuracy-driven automated

decision-making system. For model developers, the explanation can assist them to debug models and understand how the results are derived given certain inputs. For end users who consume the results delivered from automated decision-making systems, the additional explanations can better convince them why such decision is made so as to provide more trustable results. Take personalized recommender system as an example [147]. An explanation can help a user understand the process of why the model recommends the candidate items based on his/her historical behavior, and also benefit machine learning developers to understand the mechanism behind the model and figure out how to further improve the performance. Moreover, the importance of providing explainable prediction results is also emphasized by General Data Protection Regulation (GDPR) [35]. It regulates that each individual has the right of explanation – requesting an explanation of the inference result by an automated model especially when the decision largely affects the individual in finacial, mental and legal aspects. Therefore, both accuracy and explainability are supposed to be simultaneously and carefully considered when we utilize machine learning technique to make automated decisions.

Two basic requirements of the generated explanation include faithfulness and understandability, which have been commonly adopted by previous works [67, 120, 132]. The faithfulness refers to the consistency of explanation that should reflect the actual decision-making process of the model. The understandability means that resulting explanation can be understood by human regardless of the presentation forms. Both properties are supposed to help model developers understand how the model actually works by examining the generated explanation.

1.2 Categorization of Explanations

Existing explainable methods can be classified along two orthogonal dimensions: the generation paradigm and the explanation form.

Model-agnostic vs model-specific. The generation paradigm of existing approaches can be either model-agnostic or model-specific. The model-agnostic methods aim to provide post-hoc explanations regardless of the specific choice of machine learning models. Representative algorithms including LIME [107], Anchor [108] and SHAP [82] are able to identify the importance of input features that contribute most to the prediction. One advantage of model-agnostic method is that it can be applied to any trained prediction models, which is especially useful in the case where models have been deployed into industrial production environment and can hardly be changed. However, the drawback is that the generated explanation may be not faithful because it does not reflect the actual decision-making process of the black-box model. On the contrary, model-specific methods are self-explaining, i.e., the decision-making process is transparent and understandable by humans and the explanation comes along with the prediction and hence can be regarded as the byproduct of the result. The disadvantage is obvious because the prediction model is bundled with the explainer and not applicable to existing trained models. However, the model itself is explainable and usually achieves comparable performance with the accuracy-driven neural models.

Explanation forms In parallel, the explainable methods can also be categorized according to the form of output explanation.

- Feature importance [82, 19, 107, 108]: A plethora of explainable machine learning algorithms fall into this category, seeking to discover important features of input instances that make significant contribution to the prediction result. Such explanation is commonly generated for tabular data since each input dimension is comprehensible to human.

- Nearest-neighbor [117, 124]: These methods aim to find the most similar training examples to the given test sample based on certain properties of the data and prediction results.

- Visual explanation [12, 48, 96]: It is often related to images as input where heatmap can be imposed on the image to highlight the area that is relevant to the prediction. Some methods can also plot the change of a model prediction for non-image data [15, 27].

- Textual explanation [148, 128, 17]: The goal of this set of methods is to provide users with natural language sentences to justify the prediction results. Some previous works rely on predefined templates and slot-filling to constitute an explanation, while the others directly generate word sequences via natural language generation models.

- Rule sets [3, 68]: These methods target to extract interpretable rules from data such as decision trees and logical rules.

- Graph-based explanation [97, 139]: These methods are mainly designed for graph-structured data such as social network and knowledge base, etc, and the output form can be a set of nodes, edges, paths or subgraphs that summarizes the properties of the input–output pairs.

Note that the combinations of two aforementioned dimensions for explanation categorization elicit various research directions in explainable AI. For example, model-agnostic feature importance and image-based explanation generation have attracted many researchers in recent years, and a plethora of algorithms have been proposed to mitigate the explainability issue of deep neural networks for image and text classification, ranking and search, etc. Model-agnostic methods are known for its applicability and adaptability to any predictors and hence receive more attention than the model-specific ones. However, it is important to realize that even though the generated explanation can somehow describe the relationship between inputs and outputs, these model-agnostic explainers still fail to make the decision-making process transparent and understandable to humans due the black-box nature of neural networks. Therefore, in this thesis, we try to explore self-explaining model-specific approaches that can not only achieve comparable accuracy with the existing black-box neu-

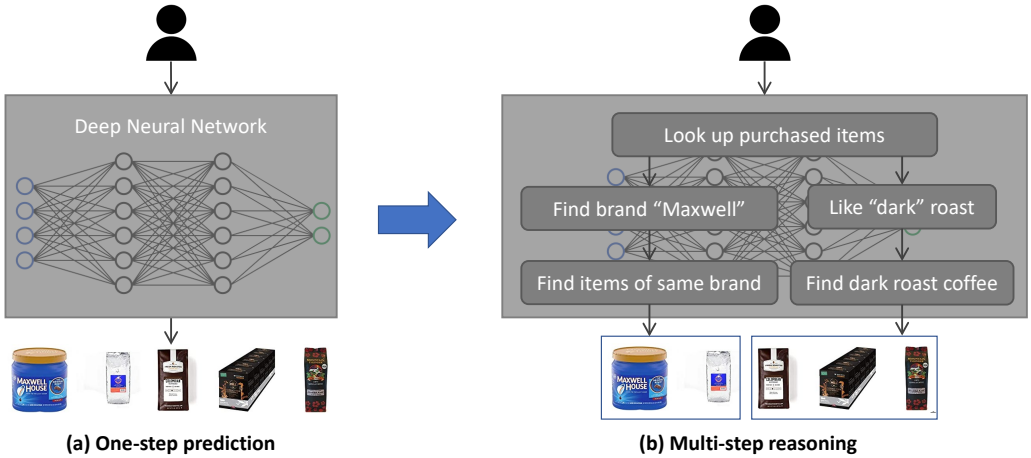


Figure 1.1: One-step prediction vs multi-step reasoning in making recommendations.

ral models, but also provide faithful and understandable explanations even with transparent and interpretable model architecture.

1.3 One-step Prediction vs Multi-step Reasoning

Most existing decision-making models only make one-step prediction by directly mapping the input data into the output space via opaque and complex operations. To achieve model-specific explanation, we aim to decompose the single-step black-box prediction into a multi-step reasoning process such that each step is comprehensible and transparent to human and the results are subsequently produced once all reasoning steps are accomplished. As an example shown in Figure 1.1, a conventional neural recommendation model will take user information as input and directly output a set of items without revealing how the recommendations are derived. In comparison, our idea is to enforce the model to conduct multi-step reasoning, where each step represents a meaningful action towards the final recommendation, e.g., to look up purchased items of the user or to find items of similar brand. In this way, the transparency of the decision-making process is increased since we know how the prediction is made via multiple comprehensible actions.

To this end, we explore to leverage heterogeneous graphs to conduct multi-step reasoning and make explainable decisions. The major reasons of choosing graphs are as follows.

- Heterogeneous graph data are ubiquitous in the real world and prior researches have shown the effectiveness of using graphs to further improve the prediction performance.
- Graph-based neural models have been widely studied but the explainability is less noticeable compared to other types of explanation.
- Graph structures naturally provide a medium to conduct multi-hop path reasoning for decision-making.

1.4 Neural Graph Reasoning for Explainable Decision-Making

We assume the heterogeneous graphs for multi-step reasoning consists of three types of nodes: source nodes, target nodes and other nodes. The source nodes refer to the input instances such as users, images and documents that are fed into the model for prediction. The target nodes can be items, labels, topics, which are the output entities to be generated by the model. The rest of nodes can be the features extracted from source and target nodes or additional knowledge to describe entities. The edges are also allowed to have different types or meanings such as the relations in knowledge graphs.

We formulate a decision-making problem under the graph reasoning framework. Given a source node on a heterogeneous graph, the goal is to predict a set of target nodes that are relevant to the source node and a set of qualified paths connecting each pair of the source and the predicted target nodes. The output target nodes are regarded as the decision and the corresponding paths are the accompanied explanation for the decision. Note that a wide range of machine learning problems can be unified under the graph reasoning framework including classification, ranking, recommendation, link prediction, etc. Take recommendation as an example. The source and target nodes refer to users and items, and the decision is equivalent to finding relevant items for recommendation. To make the graph more versatile, external knowledge graphs can also be incorporated into the user-item bipartite graph

for further extension.

There are two major lines of researches on graph reasoning. One line is to learn the graph representation such as node and edge embeddings, which can later be used to make predictions via a similarity function. For example, various embedding techniques are developed for knowledge graph representation learning [56] such as TransE [5] and RotatE [121], which aim to capture relationship among entities and relations in the low-dimensional continuous space. Meanwhile, graph neural networks [133] have become popular on learning representation for more general graph structures via multiple layers of message propagation. The representation learned by these methods can be leveraged to make accurate predictions on different tasks. However, one drawback is that the derived node or edge vectors are not explainable, i.e., it is impossible to understand the meaning of each dimension of the vector. Other research focuses on conducting multi-hop reasoning to make predictions. For example, DeepPath [139] and MINERVA [18] utilizes reinforcement learning to walk over the knowledge graph from a source node towards an unknown target node. The advantage of such methods is that they explicitly conduct multi-step reasoning for making predictions, which provide a prototype of explainable graph reasoning. However, they only focus on objective tasks such as question answering, but ignore large amount of human-centered tasks. Meanwhile, their prediction performance is not guaranteed compared to the embedding-based models using deep learning techniques.

In order to guarantee both accuracy and explainability, we attempt to make prediction by marrying both merits of powerful representation ability of neural networks and the naturally interpretable graph structures. We term such problem as *neural graph reasoning for explainable decision-making*. The major challenges of this problem are listed below.

1. *Unknown targets.* The target nodes are unknown prior to the decision-making process, which makes it very challenging to find a reliable path leading to a potentially correct target node.

2. *Large search space.* The search space at each step depends on the node degree, which

can be very large in real graph-based applications and may affect the efficiency in performing multi-hop graph reasoning.

3. *Sparse signal.* The original graph may not provide strong signals to guide the model to walk towards an unknown target, so it is important to leverage both graph structure and semantic information as auxiliary signals for path finding.

4. *Understandable explanation.* The explainable paths are supposed to be composed of comprehensible features that are relevant to the source and target nodes.

5. *Faithful explanation.* Since the resulting paths are used as the explanation to the decision, they must be faithful to the actual decision-making process, i.e., the paths can be used to trace back the multi-hop reasoning.

6. *Path quality.* Multi-faceted quality of explainable paths should be considered such as path length, diversity, patterns, etc.

7. *Model interpretability.* Some existing works proposed complicated neural networks to generate path-based explanation, which still fail to resolve the issue of model interpretability. Thus, it would be promising to make the neural model self-explaining and understandable to human.

8. *Explainability evaluation.* Unlike the accuracy evaluation, the explainability is hard to quantify and evaluate in practice and let alone how to measure the quality of path-based explanation.

1.5 Research Directions

In this thesis, we contribute to the answers to developing neural graph reasoning approaches from various perspectives including model design and implementation, interpretable architecture, data acquisition and explainability evaluation. The proposed methods are expected to work on different applications including recommendation and digital marketing.

The first research direction is how to design and implement the basic framework of neural graph reasoning for explainable decision-making, considering unknown targets, large search space, sparse signal and understandable explanation (challenges 1, 2, 3 and 4). In chapter 2, we propose a reinforcement learning based graph reasoning framework by training a neural graph walker that can reason over the graph to simultaneously generate ad-hoc explainable path and prediction results.

The second research direction is how to make the graph walker itself interpretable rather than a black-box neural network (challenge 7). In chapter 3, we decompose the model into a set of compositional neural relation modules, each of which represents a comprehensible relation in knowledge graph and can be repeatedly used to conduct single hop reasoning.

The third research direction is how to leverage knowledge graphs to generate faithful explanation (challenge 5). In chapter 4, we propose to first generate interpretable logical rules via Markov Logic Network, and the rules can be further used to guide the path reasoning process.

The last but not least research direction is how to guarantee the quality of explainable paths during graph reasoning (challenges 3, 6 and 8). In chapter 5, we consider a case where graph paths can be manually labeled by crowdsourced workers so that the explanations are available during training. We accordingly propose an inverse reinforcement learning approach to learn both reward function and policy network such that the policy-based graph walker is trained guided by the learned rewards that incorporating both accuracy and explainability factors of high-quality paths.

CHAPTER 2
REINFORCEMENT GRAPH REASONING

Recent advances in personalized recommendation have sparked great interest in the exploitation of rich structured information provided by knowledge graphs. Unlike most existing approaches that only focus on leveraging knowledge graphs for more accurate recommendation, we perform explicit reasoning with knowledge for decision making so that the recommendations are generated and supported by an interpretable causal inference procedure. In this chapter, we propose a method called Policy-Guided Path Reasoning (PGPR), which couples recommendation and interpretability by providing actual paths in a knowledge graph. Our contributions include four aspects. We first highlight the significance of incorporating knowledge graphs into recommendation to formally define and interpret the reasoning process. Second, we propose a reinforcement learning approach featuring an innovative soft reward strategy, user-conditional action pruning and a multi-hop scoring function. Third, we design a policy-guided graph search algorithm to efficiently and effectively sample reasoning paths for recommendation. Finally, we extensively evaluate our method on several large-scale real-world benchmark datasets, obtaining favorable results compared with state-of-the-art methods.

2.1 Introduction

Equipping recommendation systems with the ability to leverage knowledge graphs (KG) not only facilitates better exploitation of various structured information to improve the recommendation performance, but also enhances the explainability of recommendation models due to the intuitive ease of understanding relationships between entities [147]. Recently, researchers have explored the potential of KG reasoning in personalized recommendation. One line of research focuses on making recommendations using KG embedding models,

such as TransE [5] and node2vec [36]. These approaches align the KG in a regularized vector space and uncover the similarity between entities by calculating their representation distance [143]. However, pure KG embedding methods lack the ability to discover multi-hop relational paths. Ai *et al*. [1] proposed to enhance the collaborative filtering (CF) method over KG embedding for personalized recommendation, followed by a soft matching algorithm to find explanation paths between users and items. However, one issue of this strategy is that the explanations are not produced according to the reasoning process, but instead are later generated by an empirical similarity matching between the user and item embeddings. Hence, their explanation component is merely trying to find a post-hoc explanation for the already chosen recommendations.

Another line of research investigates path-based recommendation. For example, Gao *et al*. [33] proposed the notion of meta-paths to reason over KGs. However, the approach has difficulty in coping with numerous types of relations and entities in large real-world KGs, and hence it is incapable of exploring relationships between unconnected entities. Wang *et al*. [130] first developed a path embedding approach for recommendation over KGs that enumerates all the qualified paths between every user–item pair, and then trained a sequential RNN model from the extracted paths to predict the ranking score for the pairs. The recommendation performance is further improved, but it is not practical to fully explore all the paths for each user–item pair in large-scale KGs.

We believe that an intelligent recommendation agent should have the ability to conduct explicit reasoning over knowledge graphs to make decisions, rather than merely embed the graph as latent vectors for similarity matching. In this chapter, we consider KGs as a versatile structure to maintain the agent's knowledge about users, items, other entities and their relationships. The agent starts from a user and conducts explicit multi-step path reasoning over the graph, so as to discover suitable items in the graph for recommendation to the target user. The underlying idea is that if the agent draws its conclusion based on an explicit reasoning path, it will be easy to interpret the reasoning process that leads to each recom-

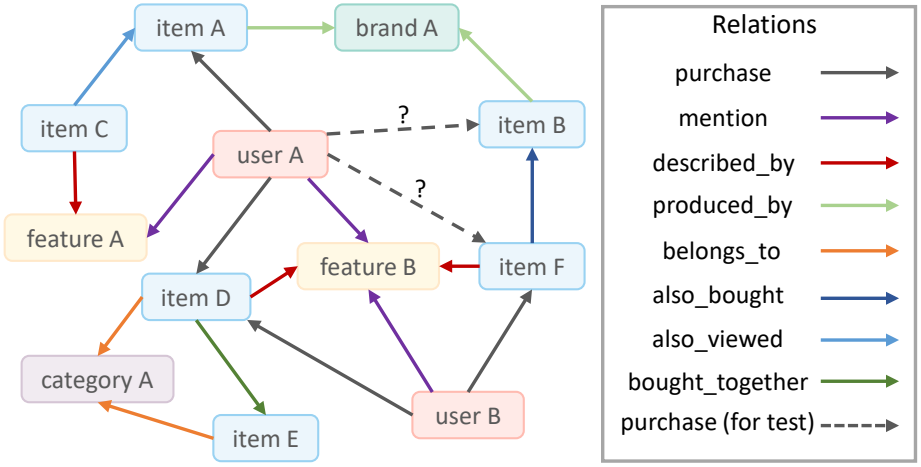


Figure 2.1: Illustration of the Knowledge Graph Reasoning for Explainable Recommendation (KGRE-Rec) problem.

mendation. Thus, the system can provide causal evidence in support of the recommended items. Accordingly, our goal is not only to select a set of candidate items for recommendation, but also to provide the corresponding reasoning paths in the graph as interpretable evidence for why a given recommendation is made. As an example illustrated in Figure 2.1, given user A, the algorithm is expected to find candidate items B and F, along with their reasoning paths in the graph, e.g., $\{\text{User } A \rightarrow \text{Item } A \rightarrow \text{Brand } A \rightarrow \text{Item } B\}$ and $\{\text{User } A \rightarrow \text{Feature } B \rightarrow \text{Item } F\}$.

In this chapter, we propose an approach that overcomes the shortcomings of the previous work. Specifically, we cast the recommendation problem as a deterministic Markov Decision Process (MDP) over the knowledge graph. We adopt a Reinforcement Learning (RL) approach, in which 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.

The main challenges are threefold. First, it is non-trivial to measure the correctness of an item for a user, so careful consideration is needed regarding the terminal conditions and RL rewards. To solve the problem, we design a soft reward strategy based on a multi-hop scoring function that leverages the rich heterogeneous information in the knowledge

graph. Second, the size of the action space depends on the out-degrees in the graph, which can be very large for some nodes, so it is important to conduct an efficient exploration to find promising reasoning paths in the graph. In this regard, we propose a user-conditional action pruning strategy to decrease the size of the action spaces while guaranteeing the recommendation performance. Third, the diversity of both items and paths must be preserved when the agent is exploring the graph for recommendation, so as to avoid being trapped in limited regions of items. To achieve this, we design a policy-guided search algorithm to sample reasoning paths for recommendation in the inference phase. We conduct several case studies on the reasoning paths to qualitatively evaluate the diversity of explainations for recommendation.

The major contributions of this work can be outlined as follows.

- We highlight the significance of incorporating rich heterogeneous information into the recommendation problem to formally define and interpret the reasoning process.
- We propose an RL-based approach to solve the problem, driven by our soft reward strategy, user-conditional action pruning, and a multi-hop scoring strategy.
- We design a beam search-based algorithm guided by the policy network to efficiently sample diverse reasoning paths and candidate item sets for recommendation.
- We extensively evaluate the effectiveness of our method on several Amazon e-commerce domains, obtaining strong results as well as explainable reasoning paths.

2.2 Related Work

2.2.1 Collaborative Filtering

Collaborative Filtering (CF) has been one of the most fundamental approaches for recommendation. Early approaches to CF consider the user–item rating matrix and predict ratings via user-based [106, 61] or item-based [112, 79] collaborative filtering methods. With the

development of dimension reduction methods, latent factor models such as matrix factorization gained widespread adoption in recommender systems. Specific techniques include singular value decomposition [62], non-negative matrix factorization [70] and probabilistic matrix factorization [111]. For each user and item, these approaches essentially learn a latent factor representation to calculate the matching score of the user–item pairs. Recently, deep learning and neural models have further extended collaborative filtering. These are broadly classified into two sub-categories: the similarity learning approach and the representation learning approach. Similarity learning adopts fairly simple user/item embeddings (e.g., one-hot vectors) and learns a complex prediction network as a similarity function to compute user–item matching scores [45]. In contrast, the representation learning approach learns much richer user/item representations but adopts a simple similarity function (e.g., inner product) for score matching [146]. However, researchers have noticed the difficulty of explaining the recommendation results in latent factor or latent representation models, making explainable recommendation [148, 147] an important research problem for the community.

2.2.2 Recommendation with Knowledge Graphs

Some previous efforts have made recommendations to users with the help of knowledge graph embeddings [95, 5]. One research direction leverages knowledge graph embeddings as rich content information to enhance the recommendation performance. For example, Zhang *et al*. [143] adopted knowledge base embeddings to generate user and item representations for recommendation, while Huang *et al*. [49] employed memory networks over knowledge graph entity embeddings for recommendation. Wang *et al*. [125] proposed a ripple network approach for embedding-guided multi-hop KG-based recommendation. Another research direction attempts to leverage the entity and path information in the knowledge graph to make explainable decisions. For example, Ai *et al*. [1] incorporated the learning of knowledge graph embeddings for explainable recommendation.

However, their explanation paths are essentially post-hoc explanations, as they are generated by soft matching after the corresponding items have been chosen. Wang *et al*. [130] proposed an RNN based model to reason over KGs for recommendation. However, it requires enumerating all the possible paths between each user–item pair for model training and prediction, which can be impractical for large-scale knowledge graphs.

2.2.3 Reinforcement Learning

Reinforcement Learning has attracted substantial interest in the research community. In recent years, there have been a series of widely noted successful applications of deep RL approaches (e.g., AlphaGo [118]), demonstrating their ability to better understand the environment, and enabling them to infer high-level causal relationships. There have been attempts to invoke RL in recommender systems in a non-KG setting, such as for ads recommendation [123], news recommendation [152] and post-hoc explainable recommendation [131]. At the same time, researchers have also explored RL in KG settings for other tasks such as question answering (QA) [139, 18, 77], which formulates multi-hop reasoning as a sequential decision making problem. For example, Xiong *et al*. [139] leveraged reinforcement learning for path-finding, and Das *et al*. [18] proposed a system called MINERVA that trains a model for multi-hop KG question answering. Lin *et al*. [77] proposed models for end-to-end RL-based KG question answering with reward shaping and action dropout. However, to the best of our knowledge, there is no previous work utilizing RL in KGs for the task of recommendation, especially when the KG has an extremely large action space for each entity node as the number of path hops grow.

2.3 Methodology

In this section, we first formalize a new recommendation problem called *Knowledge Graph Reasoning for Explainable Recommendation*. Then we present our approach based on reinforcement learning over knowledge graphs to solve the problem.

2.3.1 Problem Formulation

In general, a knowledge graph $\mathcal{G}$ with entity set $\mathcal{E}$ and relation set $\mathcal{R}$ is defined as $\mathcal{G} = \{(e, r, e') \mid e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet (e, r, e') represents a fact of the relation r from head entity e to tail entity e'. In this chapter, we consider a special type of knowledge graph for explainable recommendation, denoted by $\mathcal{G}_{\text{R}}$. It contains a subset of a *User* entities $\mathcal{U}$ and a subset of *Item* entities $\mathcal{I}$, where $\mathcal{U}, \mathcal{I} \subseteq \mathcal{E}$ and $\mathcal{U} \cap \mathcal{I} = \varnothing$. These two kinds of entities are connected through relations r_{ui}. We give a relaxed definition of k-hop paths over the graph $\mathcal{G}_{\text{R}}$ as follows.

Definition 2.3.1. (k-hop path) A k-hop path from entity e_0 to entity e_k is defined as a sequence of $k+1$ entities connected by k relations, denoted by $p_k(e_0, e_k) = \left\{ e_0 \overset{r_1}{\leftrightarrow} e_1 \overset{r_2}{\leftrightarrow} \cdots \overset{r_k}{\leftrightarrow} e_k \right\}$, where $e_{i-1} \overset{r_i}{\leftrightarrow} e_i$ represents either $(e_{i-1}, r_i, e_i) \in \mathcal{G}_{\text{R}}$ or $(e_i, r_i, e_{i-1}) \in \mathcal{G}_{\text{R}}$, $i \in [k]$.

Now, the problem of *Knowledge Graph Reasoning for Explainable Recommendation (KGRE-Rec)* can be formalized as below.

Definition 2.3.2. (KGRE-Rec Problem) Given a knowledge graph $\mathcal{G}_{\text{R}}$, user $u \in \mathcal{U}$ and integers K and N, the goal is to find a recommendation set of items $\{i_n\}_{n \in [N]} \subseteq \mathcal{I}$ 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.

In order to simultaneously conduct item recommendation and path finding, we consider three aspects that result in a good solution to the problem. First, we do not have pre-defined targeted items for any user, so it is not applicable to use a binary reward indicating whether the user interacts with the item or not. A better design of the reward function is to incorporate the uncertainty of how an item is relevant to a user based on the rich heterogeneous information given by the knowledge graph. Second, out-degrees of some entities may be very large, which degrades the efficiency of finding paths from users to potential item entities. Enumeration of all possible paths between each user and all items is

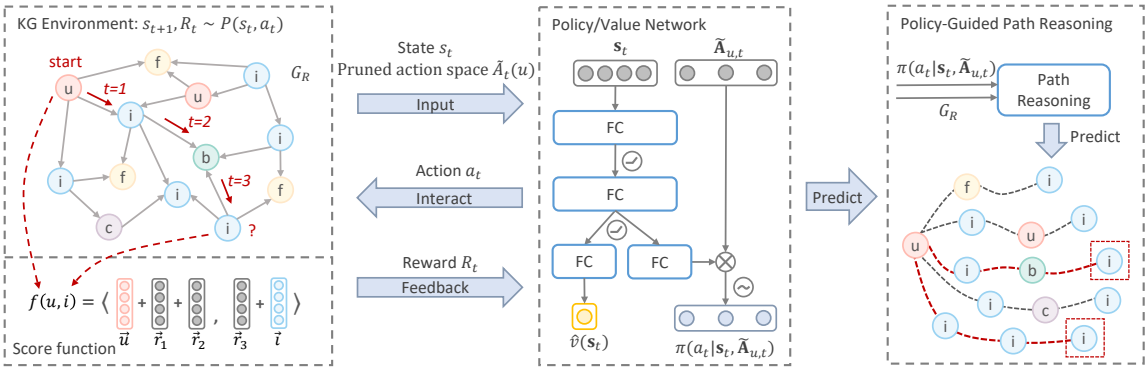


Figure 2.2: Pipeline of our Policy-Guided Path Reasoning method for recommendation. The algorithm aims to learn a policy that navigates from a user to potential items of interest by interacting with the knowledge graph environment. The trained policy is then adopted for the path reasoning phase to make recommendations to the user.

unfeasible on very large graphs. Thus, the key challenge is how to effectively perform edge pruning and efficiently search relevant paths towards potential items using the reward as a heuristic. Third, for every user, the diversity of reasoning paths for recommended items should be guaranteed. It is not reasonable to always stick to a specific type of reasoning path to provide explainable recommendations. One naive solution is post-hoc recommendation, which first generates candidate items according to some similarity measure, followed by a separate path finding procedure from the user to candidate items within the graph. The major downsides of this are that the recommendation process fails to leverage the rich heterogeneous meta-data in the knowledge graph, and that the generated paths are detached from the actual decision-making process adopted by the recommendation algorithm, which remains uninterpretable.

In the following sections, we introduce our *Policy-Guided Path Reasoning* method (PGPR) for explainable recommendation over knowledge graphs. It solves the problem through reinforcement learning by making recommendations while simultaneously searching for paths in the context of rich heterogeneous information in the KG. As illustrated in Figure 2.2, the main idea is to train an RL agent that learns to navigate to potentially "good" items conditioned on the starting user in the knowledge graph environment. The agent is then exploited to efficiently sample reasoning paths for each user leading to the

recommended items. These sampled paths naturally serve as the explanations for the recommended items.

2.3.2 Formulation as Markov Decision Process

The starting point of our method is to formalize the KGRE-Rec problem as a Markov Decision Process (MDP) [122]. In order to guarantee path connectivity, we add two special kinds of edges to the graph $\mathcal{G}_{\mathrm{R}}$. The first one are reverse edges, i.e., if $(e, r, e') \in \mathcal{G}_{\mathrm{R}}$, then $(e', r, e) \in \mathcal{G}_{\mathrm{R}}$, which are used for our path definition. The second are self-loop edges, associated with the no operation (NO-OP) relation, i.e., if $e \in \mathcal{E}$, then $(e, r_{\mathrm{noop}}, e) \in \mathcal{G}_{\mathrm{R}}$.

State The state s_t at step t is defined as a tuple (u, e_t, h_t), where $u \in \mathcal{U}$ 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. We define the k-step history as the combination of all entities and relations in the past k steps, i.e., $\{e_{t-k}, r_{t-k+1}, \ldots, e_{t-1}, r_t\}$. Conditioned on some user u, the initial state is represented as $s_0 = (u, u, \varnothing)$. Given some fixed horizon T, the terminal state is $s_T = (u, e_T, h_T)$.

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, $A_t = \{(r, e) \mid (e_t, r, e) \in \mathcal{G}_{\mathrm{R}}, e \notin \{e_0, \ldots, e_{t-1}\}\}$. Since the out-degree follows a long-tail distribution, some nodes have much larger out-degrees compared with the rest of nodes. It is fairly space-inefficient to maintain the size of the action space based on the largest out-degree. Thus, we introduce a *user-conditional action pruning strategy* that effectively keeps the promising edges conditioned on the starting user based on a scoring function. Specifically, the scoring function $f((r, e) \mid u)$ maps any edge (r, e) $(\forall r \in \mathcal{R}, \forall e \in \mathcal{E})$ to a real-valued score conditioned on user u. Then, the user-conditional pruned action space of state s_t, denoted by $\tilde{A}_t(u)$, is defined as:

$$\tilde{A}_t(u) = \{(r, e) \mid \mathrm{rank}(f((r, e) \mid u)) \leq \alpha, (r, e) \in A_t\}, \tag{2.1}$$

where α is a pre-defined integer that upper-bounds the size of the action space. The details of this scoring function $f((r, e) \mid u)$ will be discussed in the next section.

Reward Given any user, there is no pre-known targeted item in the KGRE-Rec problem, so it is unfeasible to consider binary rewards indicating whether the agent has reached a target or not. Instead, the agent is encouraged to explore as many "good" paths as possible. Intuitively, in the context of recommendations, a "good" path is one that leads to an item that a user will interact with, with high probability. To this end, we 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

$$R_T = \begin{cases} \max\left(0, \frac{f(u,e_T)}{\max_{i \in \mathcal{I}} f(u,i)}\right), & \text{if } e_T \in \mathcal{I} \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

where the value of R_T is normalized to the range of $[0, 1]$. $f(u, i)$ is also introduced in the next section.

Transition Due to the graph properties, a state is determined by the position of the entity. Given a state $s_t = (u, e_t, h_t)$ and an action $a_t = (r_{t+1}, e_{t+1})$, the transition to the next state s_{t+1} is:

$$\mathbb{P}\left[s_{t+1} = (u, e_{t+1}, h_{t+1}) | s_t = (u, e_t, h_t), a_t = (r_{t+1}, e_{t+1})\right] = 1 \tag{2.3}$$

One exception is that the initial state $s_0 = (u, u, \varnothing)$ is stochastic, which is determined by the starting user entity. For simplicity, we assume the prior distribution of users follows a uniform distribution so that each user is equally sampled at the beginning.

Optimization Based on our MDP formulation, our goal is to learn a stochastic policy π that maximizes the expected cumulative reward for any initial user u:

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t R_{t+1} \,\middle|\, s_0 = (u, u, \varnothing) \right]. \tag{2.4}$$

We solve the problem through *REINFORCE with baseline* [122] by designing a policy network and a value network that share the same feature layers. The policy network $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$ takes as input the state vector $\mathbf{s}$ and binarized vector $\tilde{\mathbf{A}}_u$ of pruned action space $\tilde{A}(u)$ and emits the probability of each action, with zero probability for actions not in $\tilde{A}(u)$. The value network $\hat{v}(\mathbf{s})$ maps the state vector $\mathbf{s}$ to a real value, which is used as the baseline in REINFORCE. The structures of the two networks are defined as follows:

$$\mathbf{x} = \text{dropout}(\sigma(\text{dropout}(\sigma(\mathbf{s}\mathbf{W}_1))\mathbf{W}_2)) \tag{2.5}$$

$$\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u) = \text{softmax}(\tilde{\mathbf{A}}_u \odot (\mathbf{x}\mathbf{W}_\text{p})) \tag{2.6}$$

$$\hat{v}(\mathbf{s}) = \mathbf{x}\mathbf{W}_\text{v} \tag{2.7}$$

Here, $\mathbf{x} \in \mathbb{R}^{d_f}$ are the learned hidden features of the state, $\odot$ is the Hadamard product, which is used to mask invalid actions here, and σ is a non-linear activation function, for which we use an Exponential Linear Unit (ELU). State vectors $\mathbf{s} \in \mathbb{R}^{d_s}$ are represented as the concatenation of the embeddings u, e_t and history h_t. For the binarized pruned action space $\tilde{\mathbf{A}}_u \in \{0, 1\}^{d_A}$, we set the maximum size d_A among all pruned action spaces. The model parameters for both networks are denoted as $\Theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_\text{p}, \mathbf{W}_\text{v}\}$. Additionally, we add a regularization term $H(\pi)$ that maximizes the entropy of the policy in order to encourage the agent to explore more diverse paths. Finally, the policy gradient $\nabla_\Theta J(\Theta)$ is defined as:

$$\nabla_\Theta J(\Theta) = \mathbb{E}_\pi \left[\nabla_\Theta \log \pi_\Theta(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u) \left(G - \hat{v}(\mathbf{s})\right) \right], \tag{2.8}$$

where G is the discounted cumulative reward from state s to the terminal state s_T.

2.3.3 Multi-Hop Scoring Function

Now we present the scoring function for the action pruning strategy and the reward function. We start with some relevant concepts.

One property of the knowledge graph $\mathcal{G}_{\mathrm{R}}$ is that given the type of a head entity and a valid relation, the type of tail entity is determined. We can extend this property by creating a chain rule of entity and relation types: $\{e_0, r_1, e_1, r_2, \ldots, r_k, e_k\}$. If the types of entity e_0 and all relations $r_1, \ldots, r_k$ are given, the types of all other entities $e_1, \ldots, e_k$ are uniquely determined. According to this rule, we introduce the concept of *patterns* as follows.

Definition 2.3.3. (k-hop pattern) A sequence of k relations $\tilde{r}_k = \{r_1, \ldots, r_k\}$ is called a valid k-hop pattern for two entities (e_0, e_k) if there exists a set of entities $\{e_1, \ldots, e_{k-1}\}$ whose types are uniquely determined such that $\{e_0 \overset{r_1}{\leftrightarrow} e_1 \overset{r_2}{\leftrightarrow} \cdots \overset{r_{k-1}}{\leftrightarrow} e_{k-1} \overset{r_k}{\leftrightarrow} e_k\}$ forms a valid k-hop path over $\mathcal{G}_{\mathrm{R}}$.

One caveat with pattern is the direction of each relation, provided that we allow reverse edges in the path. For entities e, e' and relation r, $e \overset{r}{\leftrightarrow} e'$ represents either $e \overset{r}{\rightarrow} e'$ or $e \overset{r}{\leftarrow} e'$ in the path. We refer to the relation r as a *forward* one if $(e, r, e') \in \mathcal{G}_{\mathrm{R}}$ and $e \overset{r}{\rightarrow} e'$, or as a *backward* one if $(e', r, e) \in \mathcal{G}_{\mathrm{R}}$ and $e \overset{r}{\leftarrow} e'$.

In order to define the scoring functions for action pruning and reward, we consider a special case of patterns with both forward and backward relations.

Definition 2.3.4. (*1-reverse* k-hop pattern) A k-hop pattern is *1-reverse*, denoted by $\tilde{r}_{k,j} = \{r_1, \ldots, r_j, r_{j+1}, \ldots, r_k\}$ $(j \in [0, k])$, if $r_1, \ldots, r_j$ are forward and $r_{j+1} \ldots r_k$ are backward.

In other words, paths with a *1-reverse* k-hop pattern have the form of $e_0 \overset{r_1}{\rightarrow} \cdots \overset{r_j}{\rightarrow} e_j \overset{r_{j+1}}{\leftarrow} e_{j+1} \overset{r_{j+2}}{\leftarrow} \cdots \overset{r_k}{\leftarrow} e_k$. Note that the pattern contains all backward relations when $j = 0$, and all forward relations when $j = k$.

Now we define a general multi-hop scoring function $f(e_0, e_k \mid \tilde{r}_{k,j})$ of two entities

e_0, e_k given *1-reverse* k-hop pattern $\widetilde{r}_{k,j}$ as follows.

$$f(e_0, e_k \mid \widetilde{r}_{k,j}) = \left\langle \mathbf{e_0} + \sum_{s=1}^{j} \mathbf{r_s},\ \mathbf{e_k} + \sum_{s=j+1}^{k} \mathbf{r_s} \right\rangle + b_{e_k}, \tag{2.9}$$

where $\langle \cdot, \cdot \rangle$ is the dot product operation, $\mathbf{e}$, $\mathbf{r} \in \mathbb{R}^d$ are d-dimensional vector representations of the entity e and relation r, and $b_e \in \mathbb{R}$ is the bias for entity e. When $k = 0$, $j = 0$, the scoring function simply computes the cosine similarity between two vectors:

$$f(e_0, e_k \mid \tilde{r}_{0,0}) = \langle \mathbf{e_0}, \mathbf{e_k} \rangle + b_{e_k}. \tag{2.10}$$

When $k = 1$,$j = 1$, the scoring function computes the similarity between two entities via translational embeddings [5]:

$$f(e_0, e_k \mid \widetilde{r}_{1,1}) = \langle \mathbf{e_0} + \mathbf{r_1},\ \mathbf{e_k} \rangle + b_{e_k} \tag{2.11}$$

For $k \geq 1, 1 \leq j \leq k$, the scoring function in Equation 2.9 quantifies the similarity of two entities based on a 1-reverse k-hop pattern.

Scoring Function for Action Pruning We assume that for user entity u and another entity e of other type, there exists only one *1-reverse* k-hop pattern $\tilde{r}_{k,j}$ for some integer k. For entity $e \notin \mathcal{U}$, we denote k_e as the smallest k such that $\tilde{r}_{k,j}$ is a valid pattern for entities (u, e). Therefore, the scoring function for action pruning is defined as $f((r, e) \mid u) = f(u, e \mid \tilde{r}_{k_e,j})$.

Scoring Function for Reward We simply use the 1-hop pattern between user entity and item entity, i.e., $(u, r_{ui}, i) \in \mathcal{G}_{\mathrm{R}}$. The scoring function for reward design is defined as $f(u, i) = f(u, i|\tilde{r}_{1,1})$.

Learning Scoring Function A natural question that arises is how to train the embeddings for each entity and relation. For any pair of entities (e, e') with valid k-hop pattern $\tilde{r}_{k,j}$, we seek to maximize the conditional probability of $\mathbb{P}(e' \mid e, \tilde{r}_{k,j})$, which is defined as

$$\mathbb{P}(e' \mid e, \tilde{r}_{k,j}) = \frac{\exp\left(f(e, e' \mid \tilde{r}_{k,j})\right)}{\sum_{e'' \in \mathcal{E}} \exp\left(f(e, e'' \mid \tilde{r}_{k,j})\right)}. \tag{2.12}$$

However, due to the huge size of the entity set $\mathcal{E}$, we adopt a negative sampling technique [87] to approximate $\log \mathbb{P}(e' \mid e, \tilde{r}_{k,j})$:

$$\log \mathbb{P}(e|e', \tilde{r}_{k,j}) \approx \log \sigma\left(f(e, e' \mid \tilde{r}_{k,j})\right) + m\mathbb{E}_{e''}\left[\log \sigma\left(-f(e, e'' \mid \tilde{r}_{k,j})\right)\right] \tag{2.13}$$

The goal is to maximize the objective function $J(\mathcal{G}_{\mathrm{R}})$, defined as:

$$J(\mathcal{G}_{\mathrm{R}}) = \sum_{e,e' \in \mathcal{E}} \sum_{k=1}^{K} \mathbb{1}\{(e, \tilde{r}_{k,j}, e')\} \log \mathbb{P}(e'|e, \tilde{r}_{k,j}), \tag{2.14}$$

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

2.3.4 Policy-Guided Path Reasoning

The final step is to solve our recommendation problem over the knowledge graph guided by the trained policy network. Recall that given a user u, the goal is to find a set of candidate items $\{i_n\}$ and the corresponding reasoning paths $\{p_n(u, i_n)\}$. One straightforward way is to sample n paths for each user u according to the policy network $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$. However, this method cannot guarantee the diversity of paths, because the agent guided by the policy network is likely to repeatedly search the same path with the largest cumulative rewards. Therefore, we 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. The process is described as Algorithm 1. It takes as input the given user u, the policy network $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$, horizon T, and predefined sampling sizes at each step, denoted by $K_1, \ldots, K_T$.

Algorithm 1 Policy-Guided Path Reasoning

Input: u, $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$, T, $\{K_1, \ldots, K_T\}$
Output: path set $\mathcal{P}_T$, probability set $\mathcal{Q}_T$, reward set $\mathcal{R}_T$

Initialize $\mathcal{P}_0 \leftarrow \{\{u\}\}$, $\mathcal{Q}_0 \leftarrow \{1\}$, $\mathcal{R}_0 \leftarrow \{0\}$
for $t \leftarrow 1, \ldots, T$ **do**
    Initialize $\mathcal{P}_t \leftarrow \varnothing$, $\mathcal{Q}_t \leftarrow \varnothing$, $\mathcal{R}_t \leftarrow \varnothing$
    for $\hat{p} \in \mathcal{P}_{t-1}$, $\hat{q} \in \mathcal{Q}_{t-1}$, $\hat{r} \in \mathcal{R}_{t-1}$ **do** ▷ path $\hat{p} \doteq \{u, r_1, \ldots, r_{t-1}, e_{t-1}\}$
        Set $s_{t-1} \leftarrow (u, e_{t-1}, h_{t-1})$
        Get user-conditional pruned action space $\tilde{A}_{t-1}(u)$ from environment given state s_{t-1} ▷ $p(a) \doteq \pi(a \mid \mathbf{s}_{t-1}, \tilde{\mathbf{A}}_{u,t-1})$ and $a \doteq (r_t, e_t)$
        Actions $\mathcal{A}_t \leftarrow \left\{ a \mid \text{rank}(p(a)) \leq K_t,\ \forall a \in \tilde{A}_{t-1}(u) \right\}$
        for $a \in \mathcal{A}_t$ **do**
            Get s_t, R_{t+1} from environment given action a
            Save new path $\hat{p} \cup \{r_t, e_t\}$ to $\mathcal{P}_t$
            Save new probability $p(a)\,\hat{q}$ to $\mathcal{Q}_t$
            Save new reward $R_{t+1} + \hat{r}$ to $\mathcal{R}_t$
Save $\forall \hat{p} \in \mathcal{P}_T$ if the path $\hat{p}$ ends with an item
return filtered $\mathcal{P}_T, \mathcal{Q}_T, \mathcal{R}_T$

As output, it delivers a candidate set of T-hop paths $\mathcal{P}_T$ for the user with corresponding path generative probabilities $\mathcal{Q}_T$ and path rewards $\mathcal{R}_T$. Note that each path $p_T(u, i_n) \in \mathcal{P}_T$ ends with an item entity associated with a path generative probability and a path reward.

For the acquired candidate paths, there may exist multiple paths between the user u and item i_n. Thus, for each pair of (u, i_n) in the candidate set, we select the path from $\mathcal{P}_T$ with the highest generative probability based on $\mathcal{Q}_T$ as the one to interpret the reasoning process of why item i_n is recommended to u. Finally, we rank the selected interpretable paths according to the path reward in $\mathcal{R}_T$ and recommend the corresponding items to the user.

2.4 Experiments

In this section, we extensively evaluate the performance of our PGPR method on real-world datasets. We first introduce the benchmarks for our experiments and the corresponding experimental settings. Then we quantitatively compare the effectiveness of our model with other state-of-the-art approaches, followed by ablation studies to show how parameter vari-

		CDs & Vinyl	Clothing	Cell Phones	Beauty
Entities	**Description**		**Number of Entities**		
User	User in recommender system	75,258	39,387	27,879	22,363
Item	Product to be recommended to users	64,443	23,033	10,429	12,101
Feature	A product feature word from reviews	202,959	21,366	22,493	22,564
Brand	Brand or manufacturer of the product	1,414	1,182	955	2,077
Category	Category of the product	770	1,193	206	248
Relations	**Description**		**Number of Relations per Head Entity**		
Purchase	*User* $\xrightarrow{\text{purchase}}$ *Item*	14.58 ± 39.13	7.08 ± 3.59	6.97 ± 4.55	8.88 ± 8.16
Mention	*User* $\xrightarrow{\text{mention}}$ *Feature*	$2,545.92 \pm 10,942.31$	440.20 ± 452.38	652.08 ± 1335.76	806.89 ± 1344.08
Described_by	*Item* $\xrightarrow{\text{described\_by}}$ *Feature*	$2,973.19 \pm 5,490.93$	752.75 ± 909.42	$1,743.16 \pm 3,482.76$	$1,491.16 \pm 2,553.93$
Belong_to	*Item* $\xrightarrow{\text{belong\_to}}$ *Category*	7.25 ± 3.13	6.72 ± 2.15	3.49 ± 1.08	4.11 ± 0.70
Produced_by	*Item* $\xrightarrow{\text{produced\_by}}$ *Brand*	0.21 ± 0.41	0.17 ± 0.38	0.52 ± 0.50	0.83 ± 0.38
Also_bought	*Item* $\xrightarrow{\text{also\_bought}}$ *Item*	57.28 ± 39.22	61.35 ± 32.99	56.53 ± 35.82	73.65 ± 30.69
Also_viewed	*Item* $\xrightarrow{\text{also\_viewed}}$ another *Item*	0.27 ± 1.86	6.29 ± 6.17	1.24 ± 4.29	12.84 ± 8.97
Bought_together	*Item* $\xrightarrow{\text{bought\_together}}$ another *Item*	0.68 ± 0.80	0.69 ± 0.90	0.81 ± 0.77	0.75 ± 0.72

Table 2.1: Descriptions and statistics of four Amazon e-commerce datasets: *CDs & Vinyl*, *Clothing*, *Cell Phones* and *Beauty*.

ations influence our model.

2.4.1 Data Description

All experiments are conducted on the Amazon e-commerce datasets collection [43], consisting of product reviews and meta information from Amazon.com. The datasets include four categories: *CDs and Vinyl*, *Clothing*, *Cell Phones* and *Beauty*. Each category is considered as an individual benchmark that constitutes a knowledge graph containing 5 types of entities and 7 types of relations. The description and statistics of each entity and relation can be found in Table 2.1. Note that once the type of head entity and relation are provided, the type of tail entity is uniquely determined. In addition, as shown in Table 2.1, we find that *Mention* and *Described_by* account for a very large proportion among all relations. These two relations are both connected to the *Feature* entity, which may contain redundant and less meaningful words. We thus adopt TF-IDF to eliminate less salient features in the preprocessing stage: For each dataset, we keep the frequency of feature words less than 5,000 with TF-IDF score > 0.1. We adopt the same data split rule as in previous work [146], which randomly sampled 70% of user purchases as the training data and took the rest 30% as test. The objective in the KGRE-Rec problem is to recommend items purchased by

users in the test set together with reasoning paths for each user–item pair.

2.4.2 Experimental Setup

Baselines & Metrics We compare our results against previous state-of-the-art methods. **BPR** [105] is a Bayesian personalized ranking model that learns latent embeddings of users and items. **BPR-HFT** [85] is a Hidden Factors and Topics (HFT) model that incorporates topic distributions to learn latent factors from reviews of users or items. **VBPR** [44] is the Visual Bayesian Personalized Ranking method that builds upon the BPR model but incorporates visual product knowledge. **TransRec** [42] invokes translation-based embeddings for sequential recommendation. It learns to map both user and item representations in a shared embedding space through personalized translation vectors. **DeepCoNN** or Deep Cooperative Neural Networks [153] are a review-based convolutional recommendation model that learns to encode both users and products with reviews assisting in rating prediction. **CKE** or Collaborative Knowledge base Embedding [145] is a modern neural recommender system based on a joint model integrating matrix factorization and heterogeneous data formats, including textual contents, visual information and a structural knowledge base to infer the top-N recommendations results. **JRL** [146] is a start-of-the-art joint representation learning model for top-N recommendation that utilizes multimodal information including images, text and ratings into a neural network. Note that we did not include [130] as a baseline because we are unable to enumerate all the possible paths between user–item pairs due to the large scale of our datasets.

All models are evaluated in terms of four representative top-N recommendation measures: **Normalized Discounted Cumulative Gain (NDCG)**, **Recall**, **Hit Ratio (HR)** and **Precision (Prec.)**. These ranking metrics are computed based on the top-10 predictions for every user in the test set.

Implementation Details The default parameter settings across all experiments are as follows. For the KGRE-Rec problem, we set the maximum path length to 3 based on the assumption that shorter paths are more reliable for users to interpret the reasons of recommendation. For models' latent representations, the embeddings of all entities and relations are trained based on the 1-hop scoring function defined in Equation 2.11, and the embedding size is set to 100. On the RL side, the history vector $\mathbf{h_t}$ is represented by the concatenation of embeddings of e_{t-1} and r_t, so the state vector $\mathbf{s_t} = (\mathbf{u}, \mathbf{e_t}, \mathbf{e_{t-1}}, \mathbf{r_t})$ is of size 400. The maximum size of the pruned action space is set to 250, i.e., there are at most 250 actions for any state. To encourage the diversity of paths, we further adopt action dropout on the pruned action space with a rate of 0.5. The discount factor γ is 0.99. For the policy/value network, $\mathbf{W}_1 \in \mathbb{R}^{400\times512}$, $\mathbf{W}_2 \in \mathbb{R}^{512\times256}$, $\mathbf{W}_{\text{p}} \in \mathbb{R}^{256\times250}$ and $\mathbf{W}_{\text{v}} \in \mathbb{R}^{256\times1}$. For all four datasets, our model is trained for 50 epochs using Adam optimization. We set a learning rate of 0.001 and a batch size of 64 for the *CDs & Vinyl* dataset, and a learning rate of 0.0001 and batch size of 32 for the other datasets. The weight of the entropy loss is 0.001. In the path reasoning phase, we set the sampling sizes at each step to $K_1 = 20, K_2 = 10, K_3 = 1$ for *CDs & Vinyl*, and $K_1 = 25, K_2 = 5, K_3 = 1$ for the other three datasets.

2.4.3 Quantitative Analysis

In this experiment, we quantitatively evaluate the performance of our model on the recommendation problem compared to other baselines on all four Amazon datasets. We follow the default setting as described in the previous section.

The results are reported in Table 2.2. Overall, our PGPR method consistently outperforms all other baselines on all datasets in terms of NDCG, Hit Rate, Recall and Precision. For example, it obtains a 3.94% NDCG improvement over the best baseline (JRL) on the *CDs & Vinyl* dataset, a significant improvement of 64.73% on *Clothing*, 15.53% on *Cell Phone*, and 23.95% on *Beauty*. Similar trends can be observed for Recall, Hit Rate and

Dataset	**CDs & Vinyl**				**Clothing**			
Measures (%)	NDCG	Recall	HR	Precision	NDCG	Recall	HR	Precision
BPR	2.009	2.679	8.554	1.085	0.601	1.046	1.767	0.185
BPR-HFT	2.661	3.570	9.926	1.268	1.067	1.819	2.872	0.297
VBPR	0.631	0.845	2.930	0.328	0.560	0.968	1.557	0.166
TransRec	3.372	5.283	11.956	1.837	1.245	2.078	3.116	0.312
DeepCoNN	4.218	6.001	13.857	1.681	1.310	2.332	3.286	0.229
CKE	4.620	6.483	14.541	1.779	1.502	2.509	4.275	0.388
JRL	5.378*	7.545*	16.774*	2.085*	1.735*	2.989*	4.634*	0.442*
PGPR (Ours)	**5.590**	**7.569**	**16.886**	**2.157**	**2.858**	**4.834**	**7.020**	**0.728**
Dataset	**Cell Phones**				**Beauty**			
Measures (%)	NDCG	Recall	HR	Precision	NDCG	Recall	HR	Precision
BPR	1.998	3.258	5.273	0.595	2.753	4.241	8.241	1.143
BPR-HFT	3.151	5.307	8.125	0.860	2.934	4.459	8.268	1.132
VBPR	1.797	3.489	5.002	0.507	1.901	2.786	5.961	0.902
TransRec	3.361	6.279	8.725	0.962	3.218	4.853	0.867	1.285
DeepCoNN	3.636	6.353	9.913	0.999	3.359	5.429	9.807	1.200
CKE	3.995	7.005	10.809	1.070	3.717	5.938	11.043	1.371
JRL	4.364*	7.510*	10.940*	1.096*	4.396*	6.949*	12.776*	1.546*
PGPR (Ours)	**5.042**	**8.416**	**11.904**	**1.274**	**5.449**	**8.324**	**14.401**	**1.707**

Table 2.2: Overall recommendation effectiveness of our method compared to other baselines on four Amazon datasets. The results are reported in percentage (%) and are calculated based on the top-10 predictions in the test set. The best results are highlighted in bold and the best baseline results are marked with a star ($*$).

Precision on all datasets. This shows that searching for reasoning paths over the knowledge graph provides substantial benefits for product recommendation and hence—even in the absence of interpretability concerns—is a promising technique for recommendation over graphs. Our path reasoning process is guided by the learned policy network that incorporates rich heterogeneous information from knowledge graphs and captures multi-hop interactions among entities (e.g., user mentions feature, feature is described by item, item belongs to category, etc.). This also largely contributes to the promising recommendation performance of our method. Besides, we notice that directly applying TransE for recommendation [1] slightly outperforms ours. It can be regarded as a single-hop latent matching method, but the post-hoc explanations do not necessarily reflect the true reason of generating a recommendation. In contrast, our methods generate recommendations through an

Dataset	# Valid Paths/User	# Items/User	# Paths/Item
CDs & Vinyl	173.38 ± 29.63	120.53 ± 25.04	1.44 ± 1.12
Clothing	60.78 ± 7.00	37.21 ± 7.23	1.63 ± 1.25
Cell Phone	117.22 ± 13.12	57.99 ± 14.29	2.02 ± 2.34
Beauty	59.95 ± 6.28	36.91 ± 7.24	1.62 ± 1.25

Table 2.3: Results of number of valid paths per user, number of unique items per user and number of paths per item.

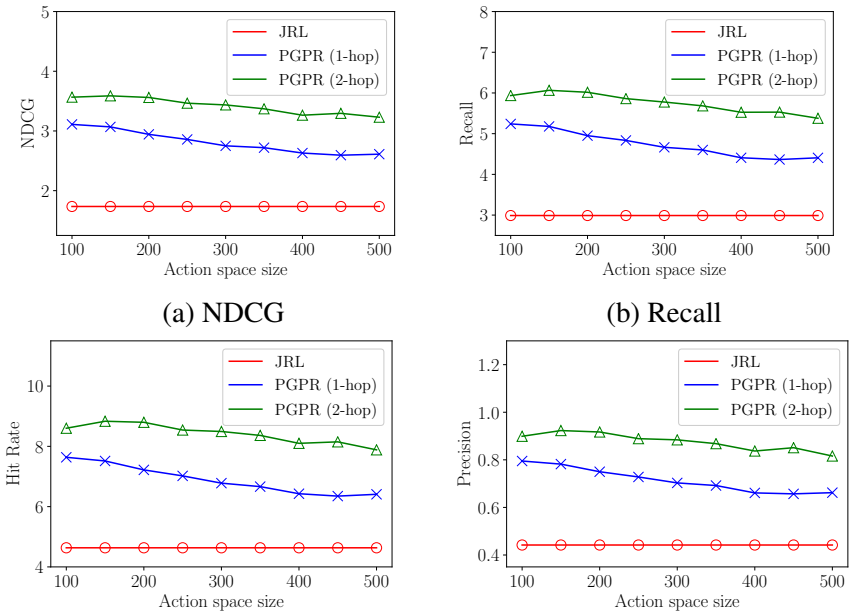
explicit path reasoning process over knowledge graphs, so that the explanations directly reflect how the decisions are generated, which makes the system transparent.

Furthermore, we examine the efficiency of our method in finding valid reasoning paths. A path is deemed *valid* if it starts from a user and ends at an item entity within three hops (i.e., at most four entities in a path). As shown in Table 2.3, we respectively report the average number of valid paths per user, the average number of unique items per user, and the average number of supportive paths per item. We observe two interesting facts. First, the success rate of our method to find valid paths is around 50%, which is calculated as the number of valid paths out of all sampled paths (200 paths for *CDs & Vinyl* dataset and 125 for others). Especially for the *Cell Phone* dataset, almost all paths are valid. Considering that the number of all possible paths from each user to items is very large and the difficulty of our recommendation problem is particularly high, these results suggest that our method performs very well in regard to path finding properties. Second, each recommended item is associated with around 1.6 reasoning paths. This implies that there are multiple reasoning paths that can serve as supportive evidence for each recommendation. One could hence consider providing more than one of these if users request further details.

2.4.4 Influence of Action Pruning Strategy

In this experiment, we evaluate how the performance of our model varies with different sizes of pruned action spaces.

Recall that conditioned on the starting user, the action space is pruned according to the



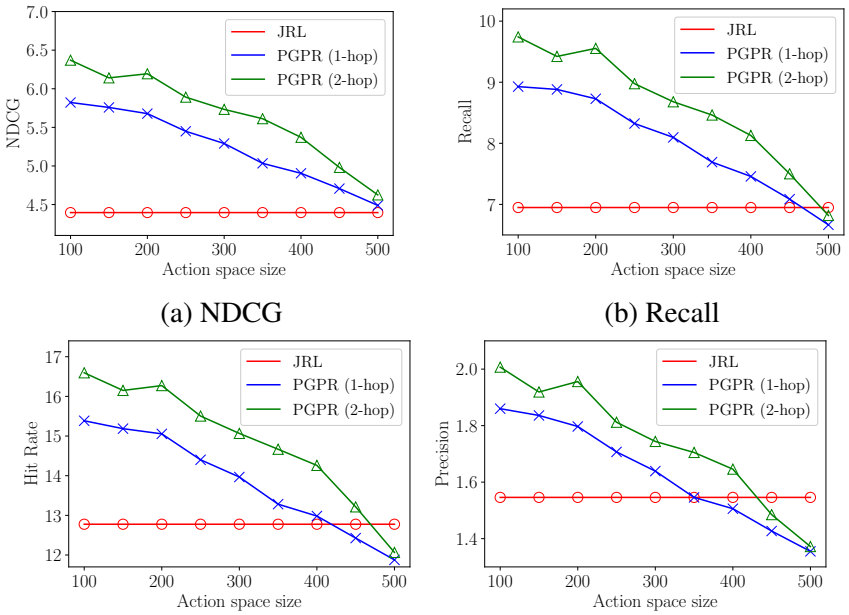
(a) NDCG (b) Recall

(c) Hit Rate (d) Precision

Figure 2.3: Recommendation effectiveness of our model under different sizes of pruned action spaces on the *Clothing* dataset. The results using multi-hop scoring function are also reported.

scoring function defined in Equation 2.9, where actions with larger scores are more likely to be preserved. In other words, larger action spaces contain more actions that are less relevant to the user. This experiment aims to show whether larger action spaces are helpful in exploring more reasoning paths to find potential items. We experiment on two selected datasets, *Beauty* and *Clothing*, and follow the default setting from the previous section, except that the size of the pruned action space is varied from 100 to 500 with a step size of 50. The results on two datasets are plotted in Figure 2.3 and Figure 2.4, respectively. The best baseline method JRL is also reported in the figures for comparison. Its performance does not depend on the action space.

There are two interesting observations in the results. First, our model outperforms JRL under most choices of pruned action space sizes. Take the *Clothing* dataset as an example. As shown in Figure 2.3, for any metric among NDCG, Recall, Hit Rate and Precision, the blue curve of our method is consistently above the red curve of JRL by a large margin for all sizes ranging from 100 to 500. The results further demonstrate the effectiveness of our



(a) NDCG (b) Recall

(c) Hit Rate (d) Precision

Figure 2.4: Recommendation effectiveness of our model under different sizes of pruned action spaces on the *Beauty* dataset. The results using multi-hop scoring function are also reported.

method compared to other baselines. Second, the performance of our model is slightly influenced by the size of the pruned action space. As shown in both figures, the common trend is that a smaller pruned action space leads to better performance. This means that the scoring function is a good indicator for filtering proper actions conditioned on the starting user. Another possible reason is that larger action spaces require more exploration in RL, but for fair comparison, we set the same parameters such as learning rate and training steps across all different choices of action space, which may lead to suboptimal solutions in cases of larger action spaces.

2.4.5 Multi-Hop Scoring Function

Besides action pruning and the reward definition, the scoring function is also used as a part of the objective function in training the knowledge graph representation. By default, we employ a 1-hop scoring function for representation learning. In this experiment, we explore whether multi-hop scoring functions can further improve the recommendation performance

of our method.

In particular, the 2-hop scoring function from Equation 2.9 is: $f(e_0, e_2|\tilde{r}_{2,2}) = \langle \mathbf{e_0} + \mathbf{r_1} + \mathbf{r_2}, \mathbf{e_2} \rangle + b_{e_2}$ for any valid 2-hop pattern $\tilde{r}_{2,2} = \{r_1, r_2\}$ between entities e_0 and e_2. This function is plugged into the objective function in Equation 2.14 for training entity and relation embeddings. All further settings are adopted from the previous action space experiment. We also plot the results in Figure 2.3 and Figure 2.4 with an additional green curve representing our new model trained by the 2-hop scoring function. Surprisingly, we find that our 2-hop PGPR method further outperforms the default model (blue curve). This improvement mainly stems from the effectiveness of the multi-hop scoring function, which captures interactions between entities with longer distance. For example, if a user purchases an item and the item belongs to a category, the 2-hop scoring function enhances the relevance between the *User* entity and the *Category* entity through the 2-hop pattern $\{$*Purchase*, *Belong_to*$\}$.

2.4.6 Sampling Size in Path Reasoning

In this experiment, we study how the sampling size for path reasoning influences the recommendation performance of our method.

We carefully design 9 different combinations of sampling sizes given a path length of 3. As listed in the first column of Table 2.4, each tuple (K_1, K_2, K_3) means that the we sample top K_t actions at step t as described in Algorithm 1. For fair comparison, the total number of sampling paths ($= K_1 \times K_2 \times K_3$) is fixed to 120 (except for the first case). We experiment on the *Clothing* and *Beauty* datasets and follow the default settings of other parameters. The recommendation results in terms of NDCG, Recall, Hit Rate and Precision are reported in Table 2.4. Interestingly, we observe that the first two levels of sampling sizes play a more significant role in finding good paths. For example, in the cases of $(25, 5, 1), (20, 6, 1), (15, 8, 1), (12, 10, 1), (10, 12, 1)$, our model performs much better than in the rest of cases. One explanation is that the first two selections of actions largely

Dataset	**Clothing**				**Beauty**			
Sizes	NDCG	Recall	HR	Precision	NDCG	Recall	HR	Precision
$25, 5, 1$	<u>2.858</u>	<u>4.834</u>	<u>7.020</u>	<u>0.728</u>	<u>5.449</u>	<u>8.324</u>	<u>14.401</u>	<u>1.707</u>
$20, 6, 1$	2.918	4.943	7.217	0.749	5.555	8.470	14.611	1.749
$20, 3, 2$	2.538	4.230	6.177	0.636	4.596	6.773	12.130	1.381
$15, 8, 1$	2.988	5.074	7.352	0.767	5.749	8.882	15.268	1.848
$15, 4, 2$	2.605	4.348	6.354	0.654	4.829	7.138	12.687	1.458
$12, 10, 1$	3.051	5.207	7.591	0.791	5.863	9.108	15.599	1.905
$12, 5, 2$	2.700	4.525	6.575	0.679	4.968	7.365	13.168	1.519
$10, 12, 1$	**3.081**	**5.271**	**7.673**	**0.797**	**5.926**	**9.166**	**15.667**	**1.920**
$10, 6, 2$	2.728	4.583	6.733	0.693	5.067	7.554	13.423	1.559

Table 2.4: Influence of sampling sizes at each level on the recommendation quality. The best results are highlighted in bold and the results under the default setting are underlined. All numbers in the table are given in percentage (%).

determine what kinds of items can be reached. After the first two steps are determined, the policy network tends to converge to selecting the optimal action leading to a good item. On the other hand, our model is quite stable if the sample sizes at the first two levels are large, which offers a good guidance for parameter tuning.

2.4.7 History Representations

Finally, we examine how different representations of state history influence our method. We consider three alternatives for $\mathbf{h}_t$: no history (0-step), last entity e_{t-1} with relation r_t (1-step), and last two entities e_{t-2}, e_{t-1} with relations r_{t-1}, r_t (2-step). Other settings are the same as in the previous experiments. As shown in Table 2.5, we find that the worst results are obtained in the 0-step case, which suggests that a state representation without history cannot provide sufficient information for the RL agent to learn a good policy. Apart from this, the performance of using 2-step history is slightly worse than that of 1-step history. One possible reason is that additional history information is redundant and even misleads the algorithm in the decision-making process.

Dataset	Clothing				Beauty			
History	NDCG	Recall	HR	Precision	NDCG	Recall	HR	Precision
0-step	1.972	3.117	4.492	0.462	3.236	4.407	8.026	0.888
1-step	2.858	4.834	7.020	0.728	5.449	8.324	14.401	1.707
2-step	2.786	4.702	6.865	0.710	5.342	8.181	14.168	1.669

Table 2.5: Results for different history representations of state. All numbers in the table are given in percentage (%).

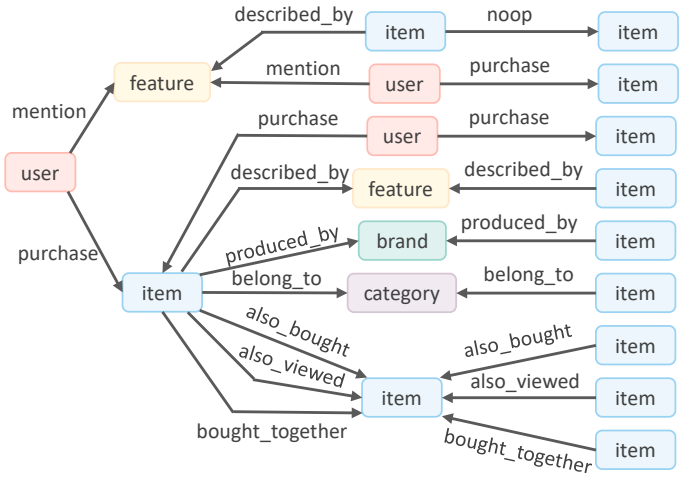


Figure 2.5: All 3-hop path patterns found in the results.

2.4.8 Case Study on Path Reasoning

To intuitively understand how our model interprets the recommendation, we give a case study here based on the results generated in the previous experiments. We first study the path patterns discovered by our model during the reasoning process, followed by various cases for recommendation.

Path patterns For a fixed path length of 3, we find that our method managed to discover 15 different path patterns, which are plotted in an aggregated form in Figure 2.5. Interesting, the pattern $\{user \xrightarrow{purchase} item \xleftarrow{purchase} user \xrightarrow{purchase} item\}$ is one kind of collaborative filtering.

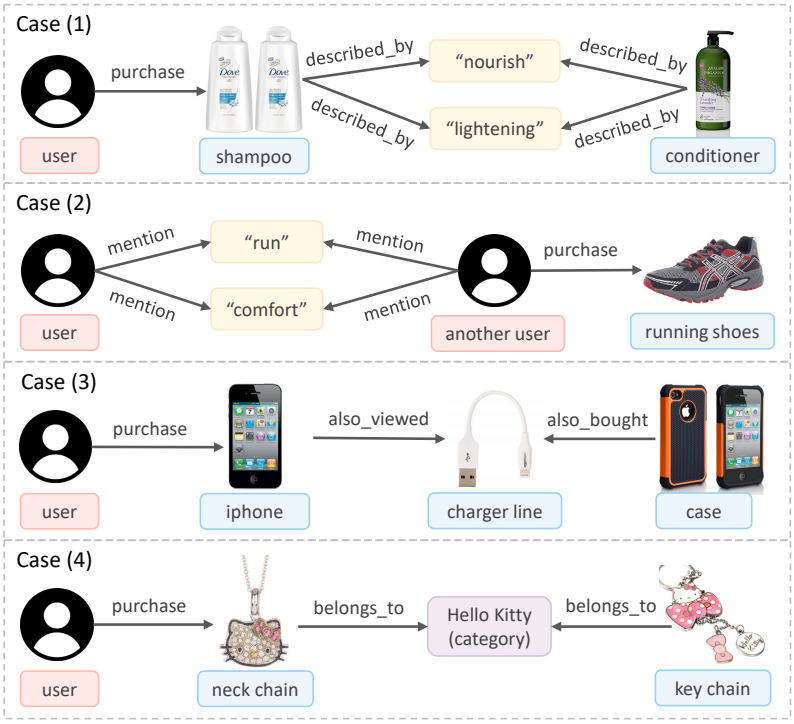


Figure 2.6: Real cases of recommendation reasoning paths.

Case study As shown in Figure 2.6, we provide several real-world examples of the reasoning paths generated by our method to illustrate how to interpret recommendations through paths.

The first example (Case 1) comes from the *Beauty* dataset, where a user purchased an item "shampoo" that was described by two feature words "nourish" and "lightening". Meanwhile, another item "conditioner" also contained these two features in some review. Therefore, our model recommended "conditioner" to this user. In the second example (Case 2), there are two users who both mentioned the feature words "run" and "comfort" in their reviews, so our method made a decision based on the purchase history of one user, by recommending the item "running shoes" purchased by the user to the other user. In the third example (Case 3), a user bought an item "iPhone" and also viewed another item "charger line". Considering that other users who purchased "phone case" would also buy "charger line", our method accordingly recommended "iPhone case" to the user. The last example (Case 4) depicts that one user purchased an item "neck chain", which belonged

to the category of “Hello Kitty”. Thus, our method recommended the user another item “key chain” that was also in the same category “Hello Kitty”. We conclude that our PGPR method not only achieves promising recommendation results, but also is able to efficiently find diverse reasoning paths for the recommendations.

2.5 Conclusion and Future Work

We believe that future intelligent agents should have the ability to perform explicit reasoning over knowledge for decision making. In this chapter, we propose RL-based reasoning over knowledge graphs for recommendation with interpretation. To achieve this, we develop a method called Policy-Guided Path Reasoning (PGPR). Based on our proposed soft reward strategy, user-conditional action pruning strategy, and a multi-hop scoring approach, our RL-based PGPR algorithm is not only capable of reaching outstanding recommendation results, but also exposes its reasoning procedure for explainability. We conduct extensive experiments to verify the performance of our approach compared with several state-of-the-art baselines. It should be noted that our 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, which will be explored in the future. We can also extend our PGPR approach to model time-evolving graphs so as to provide dynamic decision support.

CHAPTER 3
NEURAL SYMBOLIC REASONING

Recent research explores incorporating knowledge graphs (KG) into e-commerce recommender systems, not only to achieve better recommendation performance, but more importantly to generate explanations of why particular decisions are made. This can be achieved by explicit KG reasoning, where a model starts from a user node, sequentially determines the next step, and walks towards an item node of potential interest to the user. However, this is challenging due to the huge search space, unknown destination, and sparse signals over the KG, so informative and effective guidance is needed to achieve a satisfactory recommendation quality. To this end, we propose a CoArse-to-FinE neural symbolic reasoning approach (CAFE). It first generates user profiles as coarse sketches of user behaviors, which subsequently guide a path-finding process to derive reasoning paths for recommendations as fine-grained predictions. User profiles can capture prominent user behaviors from the history, and provide valuable signals about which kinds of path patterns are more likely to lead to potential items of interest for the user. To better exploit the user profiles, an improved path-finding algorithm called Profile-guided Path Reasoning (PPR) is also developed, which leverages an inventory of neural symbolic reasoning modules to effectively and efficiently find a batch of paths over a large-scale KG. We extensively experiment on four real-world benchmarks and observe substantial gains in the recommendation performance compared with state-of-the-art methods.

3.1 Introduction

Recommender systems on modern e-commerce platforms serve to support the personalization of the customer shopping experience by presenting potential products of interest to users [113, 34]. They draw on diverse forms of historical user behavior, including but

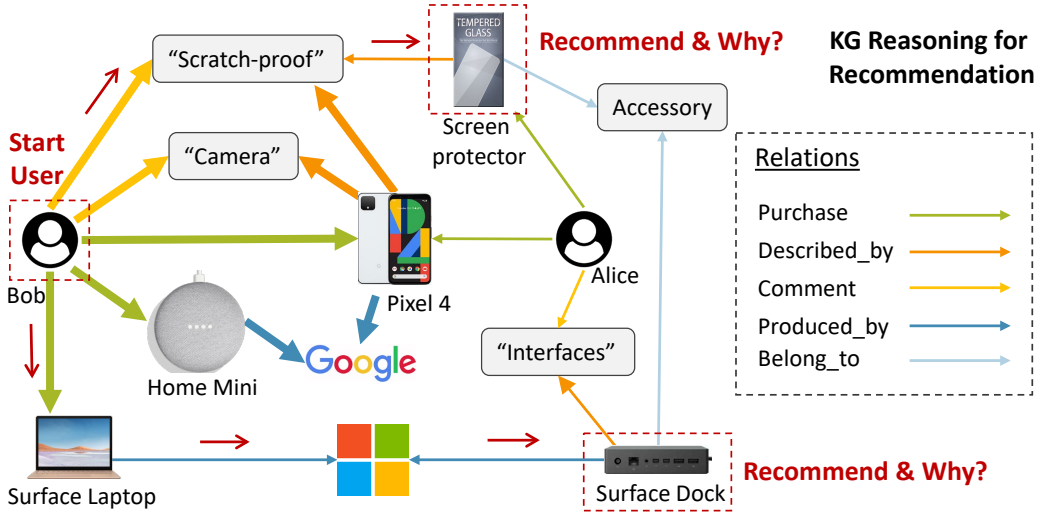


Figure 3.1: A motivating example of KG reasoning for e-commerce recommendation. Given the start user, the target destinations (i.e., items to recommend) are unknown beforehand. The goal is – guided by user behavior patterns (bold edges) – to sequentially determine the next step traversing the KG towards potential items of interest as recommendations (e.g., ***Screen protector*** and ***Surface Dock***). Two possible reasoning paths are marked with red arrows, which are taken as explanations to the recommendations.

not limited to past browsing and previously purchased products, written reviews, as well as added favorites [22]. The models are expected to capture customized patterns of user preference across products, and hence can be leveraged to provide more accurate recommendations [73]. In addition to accuracy-driven recommendation, it has become increasingly important in modern e-commerce systems to present auxiliary explanations of the recommendations [147], i.e., the system aims to supply customers with product recommendations accompanied by informative explanations about why those products are being recommended.

In this regard, knowledge graphs (KG) [47] have recently come to prominence to address both requirements. A KG can not only provide abundant information about users and items, but can also enable explainable recommendations via explicit KG reasoning [1, 136, 127]: Starting from a user node, the system sequentially determines the next-hop nodes, and moves towards potential items of interest for the user. The derived path explicitly traces the decision-making process and can naturally be regarded as an explanation for the

recommended item. For instance, as shown in Figure 3.1, one possible reasoning path is User $\xrightarrow{\text{Comment}}$ "Scratch-proof" $\xrightarrow{\text{Described\_by}^{-1}}$ "Screen protector", where the product "Screen protector" is directly used as a recommendation.

Although KG reasoning for explainable recommendation is promising, several issues still remain to be addressed. First, in order to make use of the reasoning paths to explain the decision-making process, the recommendations are supposed to be derived along with the KG reasoning. However, many existing approaches [1, 129] first predict the items to be recommended, and subsequently conduct a separate search for paths matching the user–item pair. Addressing these tasks in isolation means that the explanation may not reflect the actual decision making process for the recommendation. Moreover, this fails to allow the recommendation decision making to benefit from the KG reasoning process. We discuss this further in subsection 3.2.3.

Second, previous work on KG reasoning has largely neglected the diversity of user behavior in the historical activity data. Most approaches consider only item-side knowledge integrated from external sources, such as Freebase [150, 127] or product graphs [1, 23], restricting user-side information to simple user interactions (e.g., purchasing a product or rating a movie). However, in e-commerce recommendation, user purchases may be triggered by different aspects of past behavior. As an example, in Figure 3.1, the user having purchased product "Pixel 4" may contribute to the keyword "Camera" that the user mentioned in the comment, or to the brand ("Google") of some product ("Home Mini") owned by the user. User behavior patterns of this sort can be extracted to guide future recommendations ("Screen protectors" or "Surface Dock").

Last but not least, a lack of effective guidance on path reasoning makes it less efficient in finding potential paths in the large search space of the KG. Due to the large scale of the KG and the unknown destination before path-finding, in practice, it is infeasible to follow previous methods that enumerate paths among all user–item pairs to choose the best one. Other works [136, 77] adopt reward shaping from reinforcement learning [122] to alleviate

the issue. However, the reward signal is sparse and cannot effectively and efficiently guide the model to arrive at correct items for recommendation.

In this chapter, we seek to answer the following three questions regarding the task of KG reasoning for explainable recommendation: 1) Instead of isolating recommendation and path-finding, how to directly perform path reasoning to arrive at items of interest so that the derived paths can explain the recommendation process? 2) Besides rich item-side information, how to explicitly model diverse user behaviors from historical activities so that they can be exploited to provide good guidance in finding potential paths? 3) Upon modeling behavior, how to exploit the user model to conduct the path reasoning in a both effective and efficient manner?

To this end, we propose a CoArse-to-FinE neural symbolic reasoning method (CAFE), which first generates a coarse sketch of past user behavior, and then conducts path reasoning to derive recommendations based on the user model for fine-grained modeling. We draw inspiration from the literature in linguistics [100, 91], where the human writing process consists of multiple stages focusing on different levels of granularity. This has also been invoked in NLP tasks such as long review generation, where coarse-level aspects are first sketched to guide the subsequent long text generation [71, 21, 29]. In this work, we first compose a personalized user profile consisting of diverse user-centric patterns, each of which captures prominent coarse-grained behavior from historical user activities. Each profile can provide effective guidance on what patterns of reasoning paths may more likely lead to potential items of interest for a given user. To fully exploit the profile, we maintain an inventory of neural symbolic reasoning modules and accordingly design a path-finding algorithm to efficiently conduct batch path reasoning under the guidance of such profiles. Recommendations are consequently acquired from the batch of reasoning paths produced by the algorithm.

This work makes four key contributions.

- First, we highlight important shortcomings of past KG reasoning approaches for explain-

able recommendation, where path-reasoning and recommendation are addressed in isolation.

- Second, we introduce a coarse-to-fine paradigm to approach the problem by explicitly injecting diverse user behavior modeling into the KG reasoning process.
- Third, we propose a novel profile-guided path reasoning algorithm with neural symbolic reasoning modules to effectively and efficiently find potential paths for recommendations.
- Fourth, we experiment on four real-world e-commerce datasets showing that our model yields high-quality recommendation results and the designed components are effective.

3.2 Preliminaries

3.2.1 Concepts and Notations

In e-commerce recommendation, a *knowledge graph* (or *product graph*) denoted by $\mathcal{G}_{\mathrm{p}}$ is constructed to capture rich meta-information of products on the platform. It is defined to be a set of triples, $\mathcal{G}_{\mathrm{p}} = \{(e, r, e') \mid e, e' \in \mathcal{E}_{\mathrm{p}}, r \in \mathcal{R}_{\mathrm{p}}\}$, where $\mathcal{E}_{\mathrm{p}}$ and $\mathcal{R}_{\mathrm{p}}$ respectively denote the sets of entities and relations. A special subset of entities are called products (items), denoted by $\mathcal{I} \subseteq \mathcal{E}_{\mathrm{p}}$. Each triple $(e, r, e') \in \mathcal{G}_{\mathrm{p}}$ represents a fact indicating that head entity e interacts with tail entity e' through relation r.

At the same time, diverse user activities can also be modeled as a heterogeneous graph denoted by $\mathcal{G}_{\mathrm{u}} = \{(e, r, e') \mid e, e' \in \mathcal{E}_{\mathrm{u}}, r \in \mathcal{R}_{\mathrm{u}}\}$, where $\mathcal{E}_{\mathrm{u}}$ and $\mathcal{R}_{\mathrm{u}}$ are entity and relation sets satisfying that user set $\mathcal{U} \subseteq \mathcal{E}_{\mathrm{u}}$, item set $\mathcal{I} \subseteq \mathcal{E}_{\mathrm{u}}$, and user–item interaction $r_{ui} \in \mathcal{R}_{\mathrm{u}}$. When $|\mathcal{R}_{\mathrm{u}}| = 1$ and $\mathcal{E}_{\mathrm{u}} = \mathcal{U} \cup \mathcal{I}$, $\mathcal{G}_{\mathrm{u}}$ is a bipartite user–item graph. Here, we assume $\mathcal{G}_{\mathrm{u}}$ is the general user interaction graph consisting of diverse interactions and objects, e.g., a user can make comments as in Figure 3.1.

For convenience, we unify both product graph and user interaction graph into the same framework, which we call *User-centric KG*, denoted as $\mathcal{G} = \mathcal{G}_{\mathrm{p}} \cup \mathcal{G}_{\mathrm{u}}$ with combined entity

set $\mathcal{E} = \mathcal{E}_{\mathrm{p}} \cup \mathcal{E}_{\mathrm{u}}$ and relation set $\mathcal{R} = \mathcal{R}_{\mathrm{p}} \cup \mathcal{R}_{\mathrm{u}}$. In the remainder of this chapter, the term KG generally refers to this User-centric KG.

A *path* in the KG is defined as a sequence of entities and relations, denoted by $L = \{e_0, r_1, e_2, \ldots, r_{|L|}, e_{|L|}\}$ (or simply $L_{e_0 \rightsquigarrow e_{|L|}}$), where $e_0, \ldots, e_{|L|} \in \mathcal{E}$, $r_1, \ldots, r_{|L|} \in \mathcal{R}$ and $(e_{t-1}, r_t, e_t) \in \mathcal{G}$ for $t = 1, \ldots, |L|$. To guarantee the existence of paths, inverse edges are added into the KG, i.e., if $(e, r, e') \in \mathcal{G}$, then $(e', r^{-1}, e) \in \mathcal{G}$, where r^{-1} denotes the inverse relation with respect to $r \in \mathcal{R}$. One kind of path of particular interest is called a *user-centric path*. Such a path originates at a user entity ($e_0 \in \mathcal{U}$) and ends with an item entity ($e_{|L|} \in \mathcal{I}$). We also define a *user-centric pattern* π to be a relational path between a user and an item, $\pi = \{r_1, \ldots, r_{|\pi|}\}$. Hence, the relation sequence of any user-centric path forms a user-centric pattern. Such a pattern can be viewed as a semantic rule that describes a specific user behavior towards a product via some actions (relations) on the e-commerce platform. Additionally, we define the *user profile* $\mathcal{T}_u$ of user u to be an aggregation of user-centric patterns with weights, $\mathcal{T}_u = \{(\pi_1, w_1), \ldots, (\pi_{|\mathcal{T}_u|}, w_{|\mathcal{T}_u|})\}$, where $w_1, \ldots, w_{|\mathcal{T}_u|} \in \mathbb{N}$ are the weights of patterns. Each user profile distinctively characterizes prominent user behavior from the purchase history as well as diverse other activities, and can be leveraged to guide KG reasoning for recommendation (subsection 3.3.2).

3.2.2 Problem Formulation

In this work, we study the problem of KG reasoning for explainable recommendation in an e-commerce scenario [136]. By leveraging rich information in the KG, we aim to predict a set of items as recommendations for each user along with the corresponding user-centric paths as the explanation. The problem is formulated as follows.

Definition 3.2.1. (Problem Definition) Given an incomplete user-centric KG $\mathcal{G}$ and an integer K, for each user $u \in \mathcal{U}$, the goal is to generate

1. a set of K items $\left\{ i^{(k)} \middle| i^{(k)} \in \mathcal{I}, (u, r_{ui}, i^{(k)}) \notin \mathcal{G}, k \in [K] \right\}$, and

2. K corresponding user-centric paths $\{L_{u \rightsquigarrow i^{(k)}}\}_{k \in [K]}$.

3.2.3 A Coarse-to-Fine Paradigm

The general framework to approach the problem in Def. 3.2.1 consists of two parts: a recommendation component f_{rec} and a path inference component f_{path}. In most existing approaches [1, 125, 129, 83, 136], $f_{\text{rec}} : u, i \mapsto \mathbb{R}$ estimates a similarity score between user u and an item i using enriched information from the KG. $f_{\text{path}} : u, i \mapsto L$ outputs a user-centric path L given user u and item i (sometimes i is not necessary as input [136]). The major differences between existing works lie in 1) the technical implementation and 2) the composition and execution order of these components. Below we revisit the existing KG reasoning paradigms and highlight the benefits of the proposed coarse-to-fine paradigm.

Rec-First Paradigm One group of approaches [1, 125, 129] first makes recommendations via f_{rec}, followed by a separate process f_{path} to search paths that best match the predicted user–item pair:

$$\hat{i} = \underset{i \in \mathcal{I}}{\operatorname{argmax}} f_{\text{rec}}(u, i; \mathcal{G}), \quad \hat{L}_{u \rightsquigarrow \hat{i}} = f_{\text{path}}(u, \hat{i}; \mathcal{G}),$$

where $\hat{i}, \hat{L}_{u \rightsquigarrow \hat{i}}$ are the predicted item and path, respectively. Common choices of f_{rec} include KG embeddings [5, 125] and relational graph neural networks [114, 129]. f_{path} usually refers to a path ranking model [136, 28] or graph search algorithm [16, 1]. However, it is worth noting that one critical limitation of this paradigm is the isolation of recommendation f_{rec} and path selection f_{path}. This may degrade recommendation performance, as it is solely determined by f_{rec}, but fails to benefit from the post-hoc path-finding of f_{path}. More importantly, the reported path is not a genuine explanation of the actual recommendation process.

Path-Guided Paradigm Another line of work [136] first uses f_{path} to perform path-finding with unknown destination and the reached item is naturally adopted as the recommendation:

$$\hat{L}_{u \rightsquigarrow e_T} = f_{\text{path}}(u, -; \mathcal{G}_{\text{u}}), \quad \hat{i} = e_T,$$

where "$-$" means no item is required as input and e_T is the last entity of path $\hat{L}_{u \rightsquigarrow e_T}$. Here, f_{path} usually adopts a multi-step reasoning model such as a policy network [122, 77, 136] to sequentially pick the next step in the KG. Since the recommendation is derived along with the path inference results, f_{path} implicitly contains the recommendation process, and the resulting path can be used to track and explain the decision-making process. However, due to the challenges of unknown destinations and the huge search space of KG, the signals (e.g., rewards) are very sparse and cannot effectively guide the path inference to achieve satisfactory recommendation, in comparison with Rec-First approaches.

Coarse-to-Fine Paradigm To achieve direct path reasoning while simultaneously obtaining competitive recommendation performance, we propose a novel coarse-to-fine paradigm. In the coarse stage, we introduce a new component $f_{\text{profile}} : u \mapsto \mathcal{T}_u$ that composes a user profile $\mathcal{T}_u$ to capture prominent user behavior from historic data (details in subsection 3.3.1). Then, for fine-grained modeling, an improved variant of path inference component $f'_{\text{path}} : u, \mathcal{T}_u \mapsto L$ is developed to perform multi-step path reasoning guided by the composed user profile (details in subsection 3.3.2):

$$\mathcal{T}_u = f_{\text{profile}}(u; \mathcal{G}_{\text{u}}), \quad \hat{L}_{u \rightsquigarrow e_T} = f'_{\text{path}}(u, \mathcal{T}_u; \mathcal{G}_{\text{u}}), \quad \hat{i} = e_T. \tag{3.1}$$

The path reasoning relies on a one-step reasoner ϕ with learnable parameter Θ (see subsubsection 3.3.1). It determines the t^{th} step action by estimating the probability $P_\Theta(r_t, e_t | u, h_t)$ of choosing an outgoing edge (r_t, e_t) given user u and history trajectory $h_t = \{r_1, e_1, \ldots, r_{t-1}, e_{t-1}\}$.

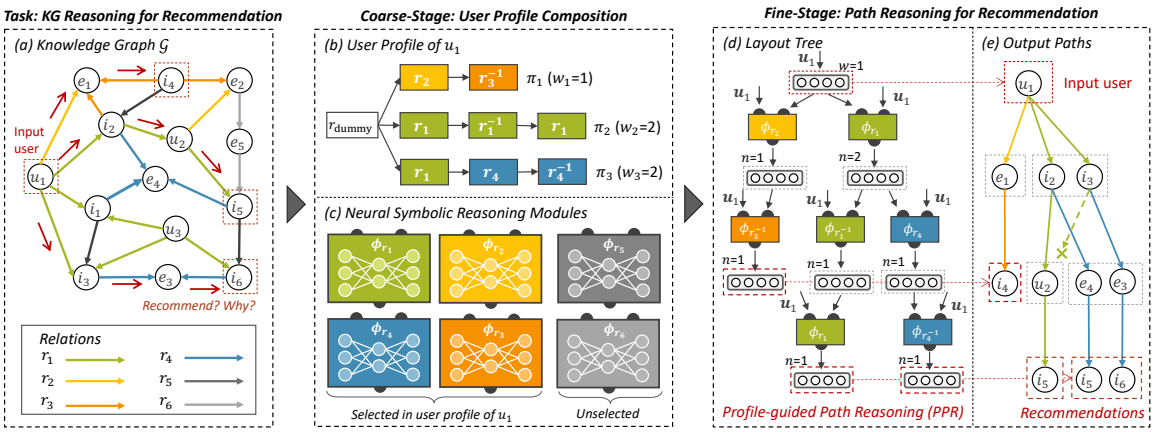


Figure 3.2: Illustration of CAFE, a coarse-to-fine KG reasoning approach. (a) Given a KG and a start user, the goal is to conduct multi-step path reasoning to derive recommendations. (b) In the coarse stage, a personalized user profile is constructed based on historic user behavior in the KG. (c) To make use of the user profile in path reasoning, an inventory of neural symbolic reasoning modules is maintained. (d) In the fine stage, a layout tree is composed with the modules based on the user profile, which is exploited by the proposed PPR algorithm (Algorithm 2) to produce (e) a batch of paths along with recommendations.

Therefore, we can estimate the probability of a multi-step path $L = \{u, r_1, e_1, \ldots, r_T, e_T\}$ being generated by ϕ:

$$\log P_\Theta(L|u) = \sum_{t=1}^{T} \log P_\Theta(r_t, e_t|u, h_t) \tag{3.2}$$

This paradigm has three notable benefits.

- Explicit user modeling from f_{profile} can detect prominent user-centric patterns, which assist the path reasoning process in arriving at potential items of interest to the user.
- Path inference via f'_{path} is conducted under the guidance of the user profile so as to improve both the effectiveness and efficiency of the path-finding process.
- The reasoner ϕ is decomposed into an inventory of neural reasoning modules, which can be composed on the fly based on the user profile to execute f'_{path}.

3.3 Methodology

Under the coarse-to-fine paradigm, we present a corresponding method called CAFE to approach the problem of KG reasoning for recommendation. As illustrated in Figure 3.2, given a KG (a), a user profile is first composed to capture prominent user-centric patterns in the coarse stage (b). To conduct multi-hop path reasoning guided by the user profile, we decompose the reasoner ϕ into an inventory of neural reasoning modules (c). In the fine stage, the selective neural symbolic reasoning modules are composed based on the user profile (d), which are exploited by a Profile-guided Path Reasoning (PPR) algorithm to efficiently perform batch path reasoning for recommendation (e).

3.3.1 Coarse-Stage: User Profile Composition

Given a user u, the goal of f_{profile} is to find a set of user-centric patterns that can distinctively characterize user behaviors, so that the potential paths with these patterns are more likely to arrive at items of interest to the given user. Since e-commerce KGs usually contain a large number of relations, we first adopt an off-the-shelf random walk based algorithm [68] to produce a candidate set of M user-centric patterns, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$, with maximum length H, from interacted user–item pairs in $\mathcal{G}$. To compose the user profile, one naive way is to assign the weights in proportion to the frequency of these retrieved patterns. However, this only provides overall information of user behavior towards items and is empirically shown not to achieve satisfying performance compared to personalized user profile (details in subsection 3.4.3).

Personalized Pattern Selection

The task of user profile composition now turns to selecting a subset from Π and assigning weights that reflect the prominent behaviors for each user. Formally, let $V_\Theta(u, \pi)$ be the prominence of a user-centric pattern π for user u. Intuitively, if π is prominent with a larger

value of $V_\Theta(u, \pi)$, it is more likely that the reasoning model ϕ can derive a path with pattern π from u to potential items of interest. Hence, we define $V_\Theta(u, \pi)$ to be the likelihood of "correct" paths being generated by ϕ:

$$V_\Theta(u, \pi) = \mathbb{E}_{L \sim D_\pi}[\log P_\Theta(L \mid u)], \tag{3.3}$$

where D_π denotes the set of paths with pattern π between the user u and interacted items in $\mathcal{G}_\mathrm{u}$, and $\log P_\Theta(L|u)$ is defined in Equation 3.2. Here, we assume the reasoner ϕ has been trained and the parameter Θ is fixed. The representation and model learning details will be discussed in subsubsection 3.3.1.

With the help of $V_\Theta(u, \pi)$, we propose a heuristic method to select prominent patterns to compose the profile for each user. Specifically, the goal is to determine the weights $\{w_1, \ldots, w_M\}$ of candidate patterns in Π and only the patterns with positive weights are kept. This can be formalized as an optimization problem:

$$\begin{aligned} \max_{w_1,\ldots,w_M} \quad & \sum_j w_j \, V_\Theta(u, \pi_j) \\ \text{s.t.} \quad & \sum_j w_j = K, \; 0 \le w_j \le K_j, j \in [M], \end{aligned} \tag{3.4}$$

where K_j is the upper bound of the quantity of pattern π_j to be adopted. The optimization problem corresponds to the well-studied bounded knapsack problem with equal weights 1 and can be easily solved [16]. Consequently, the user profile can be derived from Equation 3.4 by $\mathcal{T}_u = \{(\pi_j, w_j) \mid \pi_j \in \Pi, w_j > 0, j \in [M]\}$ (see example in Figure 3.2(b)). Each positive w_j specifies the number of paths with pattern π_j to be generated by f_path (subsection 3.3.2).

Modularized Reasoning Model

As introduced in subsection 3.2.3, the reasoner ϕ parametrized by Θ determines the next-step decision in path-finding. It maps the given user u and historic trajectory h_t to the conditional probability of choosing outgoing edge (r_t, e_t), i.e., $\phi : u, h_t \mapsto P_\Theta(r_t, e_t|u, h_t)$. Inspired by previous work [136], we can treat ϕ as a stochastic policy network [122]. However, instead of solving a reinforcement learning problem that requires a careful hand-crafted design of good reward functions, we train the model ϕ via behavior cloning [122] by reusing the sampled paths that are previously retrieved to produce candidate patterns Π.

Nevertheless, learning ϕ is still challenging due to the huge search space in the KG, where the out-degrees of nodes can be very large and the number of connecting edges varies from node to node. To address this, instead of representing ϕ as a deep and complex neural network to increase the reasoning capability, we propose to maintain an inventory of shallow *neural symbolic reasoning modules* ϕ_r with parameter Θ_r for each relation r in Π, as shown in Figure 3.2(c). Each $\phi_r(\mathbf{u}, \mathbf{h}; \Theta_r) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^d$ takes as input a user embedding $\mathbf{u}$ and a history embedding $\mathbf{h}$ and outputs the estimated vector of the next-hop entity. The network structure of each ϕ_r is defined as:

$$\phi_r(u, h; \Theta_r) = \sigma(\sigma([\mathbf{u}; \mathbf{h}]W_{r,1})W_{r,2})W_{r,3}, \tag{3.5}$$

where $[;]$ denotes concatenation, $\sigma(\cdot)$ is a nonlinear activation function (e.g., ReLU [90]), and $\Theta_r = \{W_{r,1}, W_{r,2}, W_{r,3}\}$ are the learnable parameters for the module ϕ_r.

With the module ϕ_{r_t}, we can compute the probability

$$P_\Theta(r_t, e_t \mid u, h_t) \approx \frac{1}{Z} \exp(\langle \phi_{r_t}(\mathbf{u}, \mathbf{h}_t; \Theta_{r_t}), \mathbf{e}_t \rangle), \tag{3.6}$$

where $Z = \sum_{e'_t} \exp(\langle \phi_{r_t}(\mathbf{u}, \mathbf{h}_t; \Theta_{r_t}), \mathbf{e}'_t \rangle)$ is the normalization term over all possible next-hop entities, and $\langle \cdot, \cdot \rangle$ is the dot product.

The benefits of this design are threefold. First, the total number of parameters of maintaining such small modules is smaller than that of a deep and complex neural network. Second, the space of next-hop actions is reduced from (r_t, e_t) (all outgoing edges) to e_t (only the edges of given relation), since the relation can be determined by the user profile. Third, outputting a continuous vector can effectively solve the issue of varying numbers of outgoing edges.

Objectives We consider the set of all parameters $\Theta = \{\mathbf{e}|\forall e \in \mathcal{E}\} \cup \{\Theta_r|\forall r \in \Pi\}$, where $\mathbf{e}$ denotes the entity embedding and is initialized with a pretrained KG embedding [5]. Given a positive path $L = \{u, r_1, e_1, \ldots, e_{T-1}, r_T, i^+\}$ with $(u, r_{ui}, i^+) \in \mathcal{G}$, the behavior cloning aims to minimize the following loss over Θ:

$$\ell_{\text{path}}(\Theta; L) = -\log P_\Theta(L|u) = -\sum_{t=1}^{T} \log P_\Theta(r_t, e_t|u, h_t). \tag{3.7}$$

However, the objective in Equation 3.7 only forces the reasoning modules to fit the given path, but cannot identify which path may finally lead to potential items of interest. Therefore, we impose an additional pairwise ranking loss $\ell_{\text{rank}}(\Theta; L)$ to jointly train the parameters Θ:

$$\ell_{\text{rank}}(\Theta; L) = -\mathbb{E}_{i^- \sim D_u^-} \left[\sigma\left(\left\langle \mathbf{i}^+, \hat{\mathbf{e}}_T \right\rangle - \left\langle \mathbf{i}^-, \hat{\mathbf{e}}_T \right\rangle\right)\right], \tag{3.8}$$

where D_u^- denotes the set of negative items of user u, i.e., $D_u^- = \{i|i \in \mathcal{I}, (u, r_{ui}, i) \notin \mathcal{G}\}$, $\hat{\mathbf{e}}_T = \phi_{r_T}(\mathbf{u}, \mathbf{h}_T; \Theta_{r_T})$, and $\sigma(\cdot)$ is the sigmoid function.

By aggregating Equation 3.7 and Equation 3.8 over all users in KG $\mathcal{G}_{\text{u}}$, the overall goal is to minimize the following objective:

$$\ell_{\text{all}}(\Theta) = \sum_{u} \sum_{L \sim \mathcal{L}_u} \ell_{\text{path}}(\Theta; L) + \lambda \ell_{\text{rank}}(\Theta; L), \tag{3.9}$$

Algorithm 2 Profile-guided Path Reasoning (PPR) Algorithm

1: **Input:** user u, user profile $\mathcal{T}_u$.
2: **Output:** K user-centric paths.
3: **procedure** MAIN()
4: Construct layout tree T_u based on user profile $\mathcal{T}_u$.
5: $x \leftarrow \textsc{Root}(T_u), \quad \hat{\mathbf{x}} \leftarrow \mathbf{u}, \quad \mathcal{L}_x \leftarrow \{\{u\}\}$.
6: Initialize queue $Q \leftarrow \textsc{Children}(x)$.
7: **while** $Q \neq \emptyset$ **do**
8: $x \leftarrow Q.pop(), \quad p \leftarrow \textsc{Parent}(x)$.
9: $\hat{\mathbf{x}} \leftarrow \phi_{r_x}(\mathbf{u}, \hat{\mathbf{p}}; \Theta_{r_x})$.
10: Initialize $\mathcal{L}_x \leftarrow \{\}$.
11: **for** $L \in \mathcal{L}_p$ **do**
12: $E_x \leftarrow \{e' \mid \forall (e_{|L|}, r_x, e') \in \mathcal{G}, \tau(e') = \tau_t(r_x), \text{rank}(\langle \hat{\mathbf{x}}, \mathbf{e}' \rangle) \leq n_x\}$.
13: $\mathcal{L}_x \leftarrow \mathcal{L}_x \cup (L \cup \{e'\})$, for $e' \in E_x$.
14: Update $Q \leftarrow Q \cup \textsc{Children}(x)$.
15: **return** $\bigcup_{x \in \textsc{Leaves}(T_u)} \mathcal{L}_x$.

where $\mathcal{L}_u = \{L_{u \rightsquigarrow i+} \mid (u, r_{ui}, i^+) \in \mathcal{G}_\text{u}, \text{pattern}(L_{u \rightsquigarrow i+}) \in \Pi\}$, and λ is the weighting factor to balance between the two losses.

3.3.2 Fine-Stage: Path Reasoning for Recommendation

Given the composed user profile $\mathcal{T}_u = \{(\pi_1, w_1), \ldots, (\pi_M, w_M)\}$ of user u, the goal of f_path is to output K reasoning paths along with items such that the number of paths with pattern π_j is proportional to w_j. Considering that finding each path individually is inefficient due to repeated node visitation and calculation [136], we propose a *Profile-guided Path-Reasoning* algorithm (PPR) that is capable of finding a batch of paths simultaneously via selective neural symbolic reasoning modules according to the composed user profile. As illustrated in Figure 3.2(d), it first constructs a layout tree T_u from the user profile $\mathcal{T}_u$ to specify the execution order of neural symbolic reasoning modules. Then, the reasoning modules are executed level by level to produce the next-hop embeddings that are employed to find the closest entities in the KG (Figure 3.2(e)).

The details of the algorithm are given in Algorithm 2. Specifically, the layout tree T_u (line 4) is first constructed by merging patterns in $\mathcal{T}_u$, so that each node $x \in T_u$ is associated

with a relation r_x (a dummy relation is used for the root node), and each root-to-leaf tree path corresponds to a pattern in $\mathcal{T}_u$. Next, an integer n_x is assigned to each node x, which specifies the number of entities to be generated at the current position. If node x is the root, one sets $n_x = 1$. If x is a leaf, n_x is initialized with w_j, i.e., the weight of pattern π_j that ends with relation r_x. Otherwise, n_x is updated by $n_x = \min_{c \in \text{children}(x)}(n_c)$, and subsequently, the value at each child c of node x will be refreshed as $n'_c = \lfloor n_c / n_x \rfloor$.

In fact, T_u specifies the layout of a tree-structured neural network composed of reasoning modules ϕ_{r_x} at each node x with relation r_x. The execution process of the network is described in Algorithm 2 (lines 5-15) to derive K reasoning paths simultaneously. It starts at the root node of T_u and follows level-order traversal to generate paths. At each node $x \in T_u$, ϕ_{r_x} takes as input the user embedding $\mathbf{u}$ and the embedding from its parent node and outputs an embedding vector denoted by $\hat{\mathbf{x}}$. Meanwhile, a set of new paths $\mathcal{L}_x$ up to node x is generated based on $\hat{\mathbf{x}}$ as well as the paths from its parent node $\mathcal{L}_p$. Specifically, for each path $L \in \mathcal{L}_p$, we find at most n_x new entities such that each of them is connected to the last entity in L in the KG, and its embedding is most similar to $\hat{\mathbf{x}}$. Eventually, we obtain the final results by aggregating all the paths at the leaf nodes and rank them based on the dot-product score in Equation 3.6.

3.3.3 Model Analysis

For each user, the time complexity of PPR in Algorithm 2 is $O(MH(Q + KdD))$, where Q is the running time for executing each neural symbolic reasoning module, d is the dimensionality of entity embeddings, D is the maximum node degree in the KG. Intuitively, there are at most $O(MH)$ nodes in T_u, and for each node, it costs $O(MHQ)$ time for the inference (forward pass) of the neural reasoning module, and $O(KdD)$ time to find nearest entities in Algorithm 2. Unlike existing methods [136, 1] that find each individual path separately, our PPR algorithm can derive all K paths simultaneously in the tree level order. If some resulting paths share the same entities, their corresponding embeddings will be

	CDs & Vinyl	Clothing	Cell Phones	Beauty
#Users	75,258	39,387	27,879	22,363
#Items	64,443	23,033	10,429	12,101
#Interactions	1.10M	278.86K	194.32K	198.58K
#Entities	581,105	425,534	163,255	224,080
#Relations	16	16	16	16
#Triples	387.43M	36.37M	37.01M	37.73M

Table 3.1: Statistics of four real-world Amazon KG datasets: *CDs & Vinyl*, *Clothing*, *Cell Phones*, and *Beauty*.

computed only once and hence redundant computations are avoided. The efficiency of the algorithm is also empirically evaluated in subsection 3.4.4.

3.4 Experiments

In this section, we extensively evaluate our proposed approach, providing a series of quantitative as well as qualitative analyses on several real-world datasets.

3.4.1 Experimental Setup

Dataset. We experiment on four domain-specific e-commerce datasets from Amazon [43], namely *CDs and Vinyl*, *Clothing*, *Cell Phones*, and *Beauty*. They provide both rich meta-information of products and diverse user behavior records such as purchase history, ratings, product reviews, and preferred styles. Each dataset is considered as an individual benchmark that constitutes a user-centric KG with various types of relations (including inverse relations), which implies that results are not necessarily comparable across different domains. Table 3.1 summarizes the statistical information of the four datasets. We adopt the same training (70%) and test splits (30%) as previous work [1, 136], which are publicly available[1].

Baselines and Metrics We consider three categories of recommendation approaches as baselines in the following experiments.

[1] https://github.com/orcax/PGPR

- MF-based models: **BPR** [105] is a Bayesian personalized method that optimizes a pairwise ranking between different user–item pairs for top-N recommendation. **BPR-HFT** [85] is a review-based recommendation method based on Hidden Factors and Topics (HFT) to learn latent representations of users and items with the topic distributions incorporated. **DeepCoNN** [153] makes recommendations through a Deep Cooperative Neural Network based on reviews, which is capable of encoding both users and products for rating prediction.

- KG embedding models: **TransRec** [42] is a sequential recommendation method relying on translated embeddings to align user and item latent representations in a shared space. **CKE** [145], or Collaborative Knowledge base Embedding, is a neural recommendation method based on jointly integrating matrix factorization and heterogeneous graph data to infer recommendations. **RippleNet** [125] incorporates a KG into recommendation by propagating user preferences on entities. **JRL** [146] exploits multimodal information, including images, text, and ratings as inputs of users and items into a neural network to learn their representations for recommendation. **KGAT** [129] is the the state-of-the-art KG-based model using graph-based attention techniques.

- Path reasoning models: **HeteroEmbed** [1] is the state-of-the-art Rec-First approach based on TransE [5] embeddings for recommendations, followed by a post-hoc graph search to find paths. **PGPR** [136] is the state-of-the-art path-guided model, which conducts path reasoning using reinforcement learning.

For all models, we adopted the same metrics as previous work [136] to evaluate the top-10 recommendations of each user in the test set, including Normalized Discounted Cumulative Gain (**NDCG**), **Recall**, Hit Rate (**HR**), and Precision (**Prec.**).

Implementation Details The implementations of RippleNet[2] and KGAT[3]) are from pub-

[2]https://github.com/hwwang55/RippleNet
[3]https://github.com/xiangwang1223/knowledge_graph_attention_network

lic code. We tune the parameters to achieve the best performance of these models. The results of other baselines are taken from [136]. In our model, the entity embedding dimensionality is 100. In each neural relation module ϕ_r with respect to some relation r, the parameters are $W_{r,1} \in \mathbb{R}^{200\times256}$, $W_{r,2} \in \mathbb{R}^{256\times256}$, and $W_{r,3} \in \mathbb{R}^{256\times100}$. We use Xavier initialization for the parameters and train them with Adam optimization [59] with a learning rate of 10^{-4}, batch size of 128, and a number of training epochs of 20. The history h_t is set to e_{t-1}. The weighting factor λ for the ranking loss is set to 10. The number of output paths K is 15. For fair comparison with previous work [1, 136], we also restrict the maximum path length H to 3, which leads to 15 candidate user-centric patterns in Π. The influence of these hyperparameters will be studied in subsection 3.4.6.

3.4.2 Overall Performance

We first show the top-10 recommendation performance of our proposed method CAFE compared to all baselines. We evaluate each setting 5 times and report the average scores in Table 3.2.

Overall, we observe that our method outperforms three kinds of state-of-the-art methods (KGAT, HeteroEmbed, PGPR) by a large margin across all settings. For example, on the Clothing dataset, our model achieves 6.340% in Recall, which is higher than 5.172% by KGAT, 5.466% by HeteroEmbed, and 4.834% of PGPR. Similar trends can also be observed on other benchmarks. Additionally, our model shows better ranking performance than the baselines in terms of NDCG. This is mainly attributed to the ranking loss in Equation 3.8, which encourages the model to identify the path based on whether it can lead to good items. The influence of the ranking loss will be studied in subsubsection 3.4.6.

Note that KG embedding based approaches such as RippleNet and KGAT are less competitive on these datasets. One possible reason is that unlike KGs such as Freebase, where the reasoning rules are objective and explicit (e.g., *HasNationality* = *BornIn* $\wedge$ *CityIn*), the patterns of user behavior towards items are more diverse and uncertain in e-commence set-

Dataset	**CDs & Vinyl**				**Clothing**			
Measures (%)	NDCG	Recall	HR	Precision	NDCG	Recall	HR	Precision
BPR	2.009	2.679	8.554	1.085	0.601	1.046	1.767	0.185
BPR-HFT	2.661	3.570	9.926	1.268	1.067	1.819	2.872	0.297
DeepCoNN	4.218	6.001	13.857	1.681	1.310	2.332	3.286	0.229
TransRec	3.372	5.283	11.956	1.837	1.245	2.078	3.116	0.312
CKE	4.620	6.483	14.541	1.779	1.502	2.509	4.275	0.388
RippleNet	4.871	7.145	15.727	1.852	2.195	3.892	6.032	0.603
JRL	5.378	7.545	16.774	2.085	1.735	2.989	4.634	0.442
KGAT	5.411	7.764	17.173	2.120	3.021	5.172	7.394	0.747
HeteroEmbed	5.563	7.949	17.556	2.192	3.091	5.466	7.972	0.763
PGPR	5.590	7.569	16.886	2.157	2.858	4.834	7.020	0.728
CAFE (Ours)	**6.868**	**9.376**	**19.692**	**2.562**	**3.689**	**6.340**	**9.275**	**0.975**
Improvement (%)	+22.86	+17.95	+12.17	+16.88	+19.34	+15.99	+16.34	+24.52
Dataset	**Cell Phones**				**Beauty**			
Measures (%)	NDCG	Recall	HR	Precision	NDCG	Recall	HR	Precision
BPR	1.998	3.258	5.273	0.595	2.753	4.241	8.241	1.143
BPR-HFT	3.151	5.307	8.125	0.860	2.934	4.459	8.268	1.132
DeepCoNN	3.636	6.353	9.913	0.999	3.359	5.429	9.807	1.200
TransRec	3.361	6.279	8.725	0.962	3.218	4.853	9.867	1.285
CKE	3.995	7.005	10.809	1.070	3.717	5.938	11.043	1.371
RippleNet	4.837	7.716	11.454	1.101	5.162	8.127	14.681	1.699
JRL	4.364	7.510	10.940	1.096	4.396	6.949	12.776	1.546
KGAT	5.111	8.978	12.589	1.296	6.108	10.022	16.740	1.893
HeteroEmbed	5.370	9.498	13.455	1.325	6.399	10.411	17.498	1.986
PGPR	5.042	8.416	11.904	1.274	5.449	8.324	14.401	1.707
CAFE (Ours)	**6.313**	**11.086**	**15.531**	**1.692**	**7.061**	**10.948**	**18.099**	**2.270**
Improvement (%)	+17.56	+16.72	+15.43	+24.60	+10.34	+5.16	+3.43	+14.07

Table 3.2: Overall recommendation performance of our method compared to other approaches on four benchmarks. The results are computed based on top-10 recommendations in the test set and are given as percentages (%). The best results are highlighted in bold font and the best baseline results are underlined.

tings (e.g., many factors can contribute to a user purchase behavior), making it harder to mine useful information. Our coarse-to-fine method can first learn a sketch of user behavior (i.e., user profile), which filters out noisy information that may be irrelevant to conduct path reasoning. That is why our model is able to achieve better recommendation performance. The effectiveness of user profiles is studied in the next section.

	CDs & Vinyl				Clothing			
	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
PGPR	5.590	7.569	16.886	2.157	2.858	4.834	7.020	0.728
Rand	5.308	7.217	16.158	2.003	2.654	4.727	6.875	0.680
Prior	5.924	8.259	17.825	2.327	3.157	5.031	7.376	0.773
Ours	**6.868**	**9.376**	**19.692**	**2.562**	**3.689**	**6.340**	**9.275**	**0.975**
	Cell Phones				**Beauty**			
	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
PGPR	5.042	8.416	11.904	1.274	5.449	8.324	14.401	1.707
Rand	4.545	7.229	10.192	1.087	5.293	8.256	14.564	1.718
Prior	5.255	9.842	13.097	1.359	6.180	9.393	16.258	2.024
Ours	**6.313**	**11.086**	**15.531**	**1.692**	**7.061**	**10.948**	**18.099**	**2.270**

Table 3.3: Results of recommendation performance using different user profile variants.

3.4.3 Effectiveness of User Profile (Coarse-Stage)

In this experiment, we evaluate the effectiveness of the approach to compose user profiles as described in subsection 3.3.1. Specifically, we consider the following ways to compose different $\mathcal{T}_u$ for user u while keeping the same path reasoning algorithm in subsection 3.3.2.

- *Rand* stands for randomly sampling a subset of patterns from Π to compose $\mathcal{T}_u$. This straightforward method can represent the path reasoning methods without considering user profiles.
- *Prior* samples the patterns from Π proportional to their frequencies and discards low frequency patterns. This is equivalent to assigning each user the same profile based on global information.
- CAFE is the approach we propose, which estimates the weights by solving the optimization problem in Equation 3.4.

Additionally, we also compare to the SOTA path reasoning approach *PGPR* that also fails to model user profiles.

The results on all datasets are reported in Table 3.3. We observe that our model CAFE with composed user profile exhibits better recommendation performance than other base-

	CDs & Vinyl		**Clothing**	
Time (s)	Rec. (1k users)	Find (10k paths)	Rec. (1k users)	Find (10k paths)
PGPR	287.158 ± 5.213	26.725 ± 0.572	236.118 ± 4.840	21.889 ± 0.437
Hetero.	53.984 ± 1.201	21.674 ± 0.498	55.482 ± 1.703	18.492 ± 0.399
Indiv.	71.769 ± 1.366	25.229 ± 0.482	61.519 ± 1.966	20.128 ± 0.377
Ours	$\mathbf{27.184 \pm 1.026}$	$\mathbf{17.851 \pm 0.364}$	$\mathbf{22.850 \pm 1.378}$	$\mathbf{15.233 \pm 0.309}$
	Cell Phones		**Beauty**	
Time (s)	Rec. (1k users)	Find (10k paths)	Rec. (1k users)	Find (10k paths)
PGPR	279.780 ± 5.135	25.382 ± 0.563	292.447 ± 6.139	26.396 ± 0.591
Hetero.	48.125 ± 1.148	20.037 ± 0.496	51.392 ± 1.369	21.492 ± 0.467
Indiv.	62.259 ± 1.171	23.735 ± 0.502	68.158 ± 1.209	24.938 ± 0.473
Ours	$\mathbf{23.387 \pm 1.124}$	$\mathbf{15.591 \pm 0.406}$	$\mathbf{25.220 \pm 1.141}$	$\mathbf{16.813 \pm 0.458}$

Table 3.4: Time costs of recommendations per 1k users and path finding per 10k paths.

lines. This shows that the path reasoning guided by the user profile can find user-centric paths of higher quality, which are more likely to arrive at an item node of interest to the user. In addition, we also note that the profile-driven methods CAFE and *Prior* outperform the ones without profiles (*PGPR*, *Rand*). This suggests that user profiles can benefit the path reasoning process.

3.4.4 Efficiency of Path Reasoning (Fine-Stage)

We further study the efficiency of our path reasoning algorithm in subsection 3.3.2 compared to other path-finding baselines. Specifically, we consider the SOTA Path-Guided method *PGPR* and the SOTA Rec-First method *HeteroEmbed*. We also include a variant of our algorithm in Algorithm 2 named *Indiv.*, which simply finds each individual path one by one. These algorithms are evaluated on the empirical running time of 1) making recommendations (including both items and paths) for 1k users and 2) the path-finding process (only paths) for generating 10k paths. All experiments are conducted on the same hardware with Intel i7-6850K CPU, 32G memory and one Nvidia 1080Ti GPU. The results are reported in Table 3.4.

We observe that our method costs the least time for both tasks among all tested algo-

	CDs & Vinyl				Clothing			
#Patterns	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
100%	6.868	9.376	19.692	2.562	3.689	6.340	9.275	0.975
70%	6.713	9.152	19.270	2.488	3.586	6.175	9.020	0.946
Decrease	2.31%	2.45%	2.19%	2.98%	2.87%	2.68%	2.83%	3.05%
	Cell Phones				**Beauty**			
#Patterns	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
100%	6.313	11.086	15.531	1.692	7.061	10.948	18.099	2.270
70%	6.143	10.764	15.098	1.639	6.974	10.803	17.835	2.225
Decrease	2.77%	2.99%	2.87%	3.23%	1.25%	1.34%	1.48%	2.02%

Table 3.5: Experimental results for unseen patterns

rithms. In particular, our method is about $10\times$ faster than *PGPR* in making recommendations on all benchmarks, both of which aim to find paths with unknown destination. One reason is that *PGPR* is required to find a lot of candidate paths, which are then ranked to obtain top 10 paths for recommendation. On the contrary, our method seeks out useful paths based on the user profile, and hence it saves much more time in path-reasoning based recommendation. In addition, for both tasks, our method costs less time than *Indiv.*, which means that the batch path finding algorithm in Algorithm 2 is more efficient than finding paths individually. Our algorithm thus avoids redundant computation of embeddings and nearest nodes searches.

3.4.5 Robustness to Unseen Patterns

Recall that the candidate set Π cannot exhaustively cover all possible user-centric patterns in a very large KG, since only high frequency patterns will be collected by the algorithm [68]. Therefore, in this experiment, we investigate if unseen patterns that do not exist in Π will influence the performance. To conduct the experiment, in the coarse stage of user profile composition, we randomly preserve 70% of the candidate patterns in Π for each user to compose the user profile. The remaining 30% of patterns are unseen to the model for each user. All other settings remain the default ones.

The results on the four datasets are reported in Table 3.5. It is interesting to see that the decrease in performance is at around 1.5–3%, which is marginal compared to the regular setting. This shows that our model is robust to unseen patterns for user profile composition and can still provide high-quality recommendations.

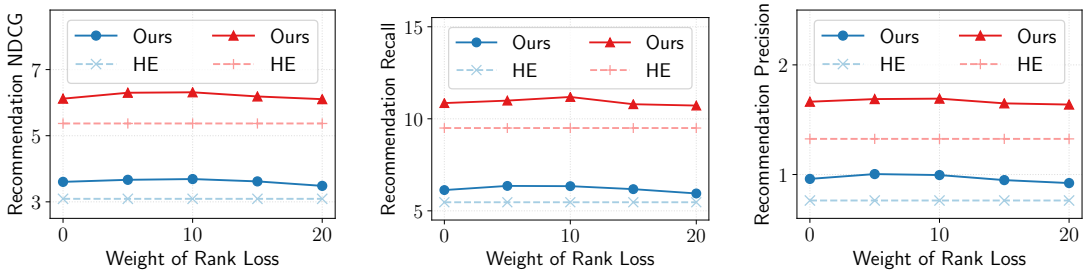
3.4.6 Ablation Study

We study how different settings of hyperparameters influence the recommendation quality of our model. We consider the ranking weight and the number of sampling paths on the Cell Phones dataset only due to space constraints.

Influence of ranking loss

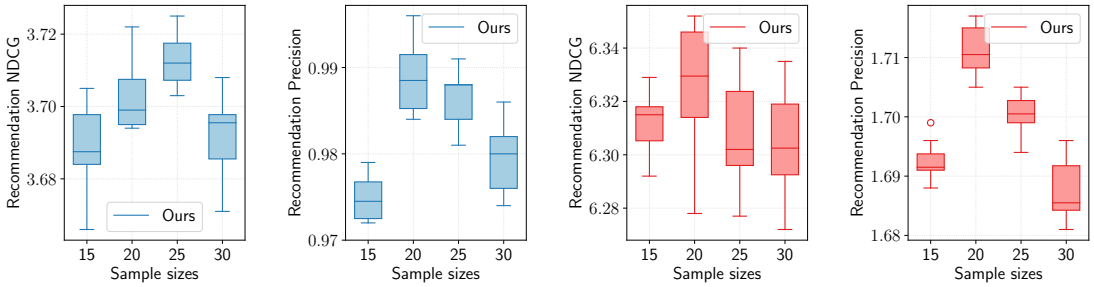
We first show the influence of the ranking loss in Equation 3.9 under different values of the weighting factor $\lambda \in \{0, 5, 10, 15, 20\}$, where $\lambda = 0$ means no ranking loss is imposed for training. The results are plotted in Figure 3.3, including our model (red curves) and the best baseline HeteroEmbed (blue curves).

We observe two interesting trends. First, our model consistently outperforms HeteroEmbed under all settings of λ in terms of NDCG, Recall, and Precision. Even without the ranking loss, our model can still guarantee a high quality of recommendation. On the other hand, a proper choice of λ (e.g., $\lambda = 10$) not only benefits the direct ranking effect (NDCG), but also boosts the model's ability to find more relevant items (recall, hit rate, and precision). Second, a larger weight of the ranking loss may not always entail a better performance, since there is a trade-off between the ranking (Equation 3.8) and path regularization (Equation 3.7). This is reasonable because if the ranking loss plays a dominant role, which implies that the model pays less attention to fitting paths, as a consequence, it may fail to find the correct paths that reach promising items.



(a) NDCG (b) Recall (c) Precision

Figure 3.3: Results of varying ranking weights on Clothing (blue) and Cell Phones (red) datasets. (HE: [1])



(a) NDCG (b) Precision (c) NDCG (d) Precision

Figure 3.4: Results of different number of output reasoning paths on Clothing (blue) and Cell Phones (red) datasets.

Influence of sampling sizes of output paths

Furthermore, we study how the performance varies with different sampling sizes of output reasoning paths $K \in \{15, 20, 25, 30\}$ (see subsection 3.3.2).

In Figure 3.4, we illustrate with box plots the recommendation performance of all users in terms of various metrics. We observe similar trends across all metrics in that there exists an optimal choice of K under each setting, e.g., $K = 20$ for NDCG on the Cell Phones dataset. The variances are within acceptable ranges, which means that the path reasoning procedure of our model leads to satisfying results for most of the users. One possible reason is that some items suitable for recommendation are in fact ranked relatively low. Smaller sampling sizes lead to smaller search spaces that preclude the discovery of such low-ranked items.

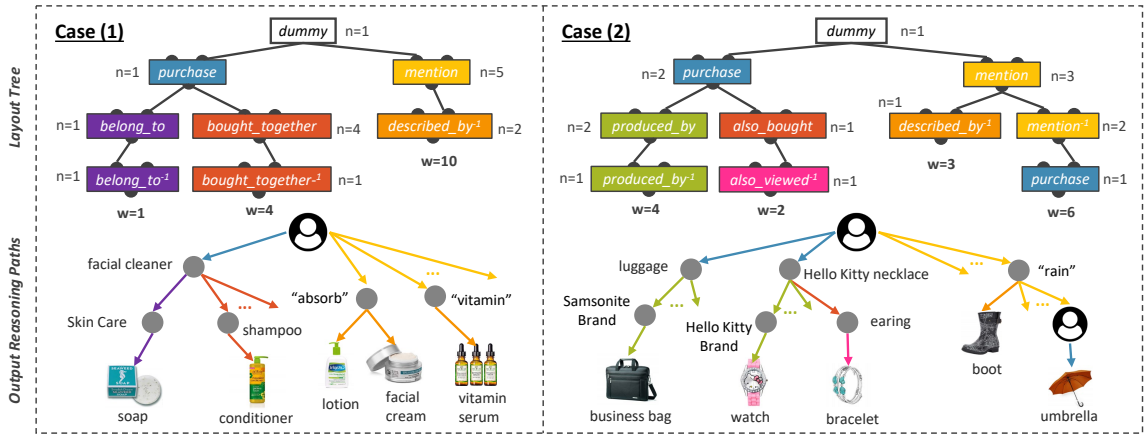


Figure 3.5: Two real cases discovered by our model, each containing a layout tree merged from user profile and a subset of reasoning paths. The end nodes in the resulting paths are the predicted items for recommendation.

3.4.7 Case Study

We showcase two recommendation examples with path-based explanations produced by our model CAFE. As shown in Figure 3.5, each case consists of a layout tree merged from the user profile along with a subset of generated reasoning paths. In Case 1, the pattern containing the "mention" relation takes the dominant role ($w = 10$). For example, the user mentions the keywords "absorb", "vitamin". The recommended items "lotion" and "facial cream" match "absorb", and "vitamin serum" is also consistent with "vitamin". Case 2 shows a user profile with more diverse patterns. For example, the user purchased a "necklace" by "Hello Kitty". It is reasonable for our method to recommend "watch" from the same "Hello Kitty" brand. Similar inferences can also be drawn for "business bag". Moreover, the interaction with another user and the "rain" feature leads to "umbrella" being recommended. In these cases, our method is capable of producing relevant recommendations along with the explainable paths via explicit KG reasoning.

3.5 Related Work

There are two main research lines related to our work: KG-based explainable recommendation and multi-behavior recommendation.

KG-based Explainable Recommendation Explainable recommendation [147, 148, 72, 9, 13, 12, 135] refers to a decision-making system that not only provides accurate recommendation results, but also generates explanations to clarify why the items are recommended.

One line of research focuses on the KG embedding approach. Several works integrate KG representation learning into the recommendation model [143, 52, 126, 49, 41]. Typically, they assume that the recommended items and their attributes can be mapped into a latent vector space along with transnational relations between the them. For example, [143] propose the CKE model, which incorporates diverse item types information into Collaborative Filtering. [52] integrate a KG in multimodal formats capturing dynamic user preferences by modeling the sequential interactions over the KG. [126] consider both semantics and knowledge representations of news contents for improved news recommendation. [49] leverage KG to enhance item representations and [41] jointly conduct KG completion and item recommendations. These methods demonstrate the effectiveness of incorporating KG embedding into recommendation. However, they fail to directly leverage the KG structure to generate reasoning paths as explanations for the recommendation [147].

Another line of work explores incorporating KG reasoning into the process of recommendation. The graph structure empowers the system to exploit informative features and also to deliver intuitive path-based explanations. Early works [8] propose to model logic rules to conduct explicit reasoning over a KG for explainability. However, the rules are handcrafted and can hardly generalize to unexplored entity correlations. In contrast, recent approaches adopt deep neural networks to learn a direct mapping among users, items, and other relations in a KG to enable reasoning for explainable recommendation. Some

approaches [83, 125, 129] only use item-side knowledge but neglect the diverse historical activities of users, while others [7, 1] isolate path generation and recommendations, so the resulting path may be irrelevant to the actual decision making process. We argue that both of these two types of methods fail to model user behaviors to conduct an explicit path reasoning process, which makes the recommendation process less intuitive. Recently, [136] and [149] perform explicit KG reasoning for explainable recommendation via reinforcement learning. Although their paths are generated together with the recommended items, the recommendation performance is limited by the large search space of the KG and the weak guidance of sparse rewards. In this work, we follow the setting of KG reasoning for explainable recommendation [136], but aim to provide better guidance from user history behavior, as confirmed in our experiments.

Multi-Behavior Recommendation On modern e-commerce platforms, users can interact with the system in multiple forms [80, 75, 119, 63]. [80] provide several case studies covering the influence of clicking and saving behavior analysis on the final purchase decision. Existing methods for multi-behavior recommendations may be divided into two categories: collective matrix factorization based approaches and approaches based on learning from implicit interactions. [119] propose factorizing multiple user–item interaction matrices as a collective matrix factorization model with shared item-side embeddings across matrices. [151] learn different embedding vectors for different behavior types in an online social network. [63] share the user embeddings in recommendation based social network data based on the CMF method. In contrast, [81] proposed an extension of Bayesian Personalized Ranking [105] as multi-channel BPR, to adapt the sampling rule from different types of behavior in the training of standard BPR. [38] proposed sampling unobserved items as positive items based on item–item similarity, which is calculated using multiple types of feedback. However, none of these methods consider the reasoning framework to provide explainable recommendations, let alone explicitly model diverse user behaviors over KGs on e-commerce platforms.

3.6 Conclusion

In this chapter, we propose a new coarse-to-fine KG reasoning approach called CAFE for explainable recommendation. Unlike traditional KG based recommendations, our method is characterized by first composing a user profile to capture prominent user behaviors in the coarse stage, and then in the fine stage, conducting path reasoning under the guidance of the user profile. Since the recommendation and path reasoning processes are closely coupled with each other, the output paths can be regarded as the explanation to the recommendations. We extensively evaluate our model on several real-world datasets and show that the proposed approach delivers superior results in recommendation performance.

CHAPTER 4
NEURAL LOGIC REASONING

Knowledge graphs (KG) have become increasingly important to endow modern recommender systems with the ability to generate traceable reasoning paths to explain the recommendation process. However, prior research rarely considers the faithfulness of the derived explanations to justify the decision-making process. To the best of our knowledge, this is the first work that models and evaluates faithfully explainable recommendation under the framework of KG reasoning. Specifically, we propose neural logic reasoning for explainable recommendation (LOGER) by drawing on interpretable logical rules to guide the path-reasoning process for explanation generation. We experiment on three large-scale datasets in the e-commerce domain, demonstrating the effectiveness of our method in delivering high-quality recommendations as well as ascertaining the faithfulness of the derived explanation.

4.1 Introduction

Compared with traditional recommender systems (RS), explainable recommendation is not only capable of providing high-quality recommendation results but also offers personalized and intuitive explanations [147]. Incorporating a knowledge graph (KG) into recommender systems has become increasingly popular, since KG reasoning is able to generate explainable paths connecting users to relevant target item entities. At the same time, there is increasing demand for systems to ascertain the faithfulness of the generated explanation, i.e., assess whether it faithfully reflects the reasoning process of the model and is consistent with the historic user behavior.

However, previous work has largely neglected faithfulness in KG-enhanced explainable recommendation [135, 28]. A number of studies [67, 46, 132] argue that faithful

explanations should also be personalized and gain the capability to reflect the personalized user historic behavior. However, to the best of our knowledge, none of the existing explainable recommendation models based on KGs have considered faithfulness in the explainable reasoning process and its evaluation on the generated explainable paths. For instance, PGPR [136, 149] infers explainable paths over the KG without considering personalized user behavior, and its prediction on next potential entities is merely based on the overall knowledge-driven rewards. CAFE [137] builds user module profiles to guide the path inference procedure. However, as illustrated in [120], such neural module networks only implicitly abstract the reasoning process and lack of considering the faithfulness of explanations.

In this chapter, we propose a new KG-enhanced recommendation model called LOGER to produce faithfully explainable recommendation via neural logic reasoning. To fully account for heterogeneous information and rules about users and items from the KG, we leverage an interpretable neural logic model for logical reasoning, enhanced by a general graph encoder that learns KG representations to capture semantic aspects of entities and relations. These two components are iteratively trained via the EM algorithm by marrying the merits of interpretability of logical rules and the expressiveness of KG embeddings. Subsequently, the learned rule weights are leveraged to guide the path reasoning to generate faithful explanations. The derived logical rules are expected to be consistent with historic user behavior and the resulting paths genuinely reflect the decision making process in KG reasoning. We experiment on three large-scale datasets for e-commerce recommendation that cover rich user behavior patterns. The results demonstrate the superior recommendation performance achieved by our model compared to the state-of-the-art baselines, with the guarantee of the faithfulness on the generated path-based explanations. The contributions of this chapter are threefold.

- We highlight the significance of considering faithfulness in explainable recommendation.

- We propose a novel approach that incorporates interpretable logical rules into KG path reasoning for recommendation and explanation generation.
- We experiment on three large-scale datasets showing promising recommendation performance as well as faithful path-based explanation.

4.2 Problem Formulation

A knowledge graph (KG) for recommendation is defined as $\mathcal{G} = \{(e_h, r, e_t) \mid e_h, e_t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ denotes the entity set consisting of sets of users $\mathcal{U}$, items $\mathcal{I}$, and other entities, while $\mathcal{R}$ denotes the relation set. Each triplet (e_h, r, e_t) represents a fact indicating head entity e_h interacts with tail entity e_t via relation r. In recommendation tasks, we are particularly interested in user–item interactions $\{(u, r_{ui}, v) \mid u \in \mathcal{U}, r_{ui} \in \mathcal{R}, v \in \mathcal{I}\}$ with the special relation r_{ui} meaning *purchase* in e-commerce or *like* in movie recommendation.

The problem of KG reasoning for explainable recommendation is formulated as follows. Given an incomplete KG $\mathcal{G}$ with missing user–item interactions, for every user $u \in \mathcal{U}$, the goal is to select a set of items as recommendations $\{v|(u, r_{ui}, v) \notin \mathcal{G}, v \in \mathcal{I}\}$ along with a set of paths as explanations connecting each pair of the user and a predicted item. The key challenge is to not only guarantee the recommendation quality with the rich information in KG, but also generate faithful explanations that reflect the actual decision-making process of the recommendation model and are consistent with historic user behavior.

4.3 Proposed Method

We introduce the novel neural LOGic Explainable Recommender (LOGER) for producing faithfully explainable recommendations with a KG. As illustrated in Figure 4.1, it consists of three components: (i) a KG encoder for learning embeddings of KG entities and relations to capture their semantics, (ii) a neural logic model for conducting interpretable logical reasoning to make recommendations, and (iii) a rule-guided path reasoner for generating

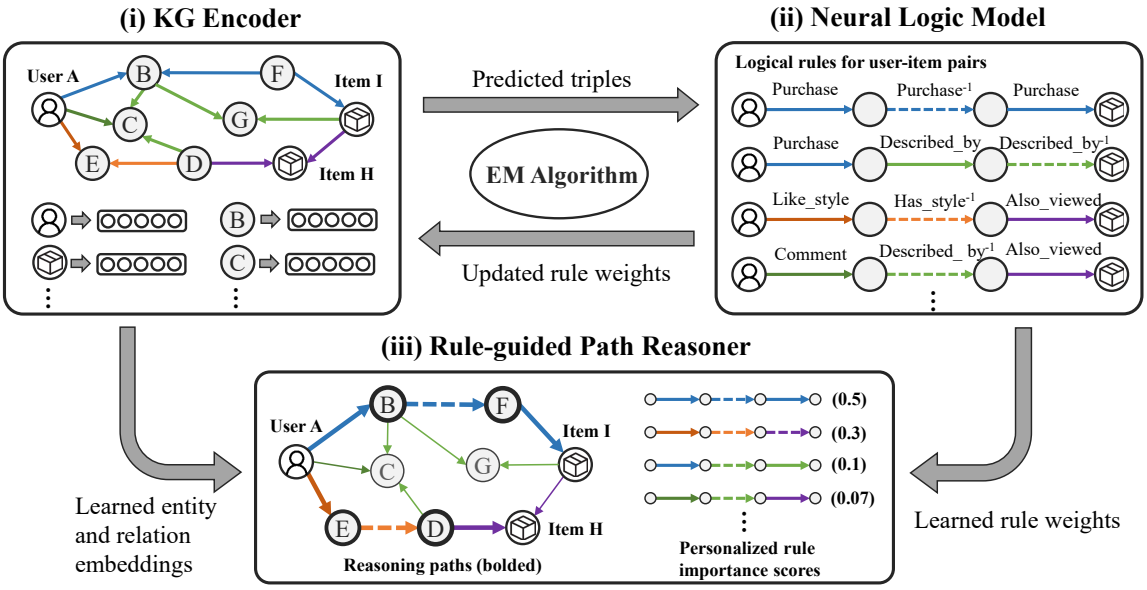


Figure 4.1: Illustration of the proposed method for explainable recommendation including (i) a KG encoder, (ii) a neural logic model, and (iii) a rule-guided path reasoner.

faithfully explainable paths. Both KG encoder and neural logic model are trained iteratively via the EM algorithm [92] so that they mutually benefit to make recommendations via logical reasoning. Additionally, personalized rule importance scores are derived for every user and leveraged to guide the path reasoning for faithful explanation generation.

4.3.1 KG Encoder

Let X_{hrt} be a binary random variable indicating whether a triplet (e_h, r, e_t) is true or not, $X_{\mathcal{G}} = \{X_{hrt} \mid (e_h, r, e_t) \in \mathcal{G}\}$ be a random variable regarding all observed triplets in the KG $\mathcal{G}$, and $X_H = \{X_{hrt} \mid (e_h, r, e_t) \in H\}$ be a random variable of hidden user–item interactions in $H = \{(u, r_{ui}, v) \mid u \in \mathcal{U}, v \in \mathcal{I}, (u, r_{ui}, v) \notin \mathcal{G}\}$. The KG encoder is generally defined as a triplet-wise function $f_\theta : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \mapsto [0, 1]$ parametrized by θ that maps each triplet to a real-valued score. For any triplet $(e_h, r, e_t) \in \mathcal{G} \cup H$, we can interpret its truth probabilistically via the KG encoder f_θ as $q(X_{hrt}|\theta) = \mathsf{Bernoulli}(X_{hrt}|f_\theta(e_h, r, e_t))$. The KG encoder f_θ can be instantiated with any existing KG embedding [56] or graph neural network [133] model.

4.3.2 Neural Logic Model

We focus on composition rules for user–item interactions, i.e., r_{ui} is a composition of relations $r_1, \ldots, r_j$ if $(u, r_1, e_1) \wedge \cdots \wedge (e_{j-1}, r_j, v) \Rightarrow (u, r_{ui}, v), \forall u \in \mathcal{U}, v \in \mathcal{V}, e_1, \ldots, e_{j-1} \in \mathcal{E}$. Given a set of logical rules L mined from the KG, the goal of this component is, for every user $u \in \mathcal{U}$, to emit a set of personalized rule importance scores $y_u = \{y_{u,l}\}_{l \in L}$ to capture the historic user behavior. To achieve this, we build upon Markov Logic Networks [102], an interpretable probabilistic logic reasoning method that models the joint distribution of all triplets via a set of logical rules L, i.e., $p(X_{\mathcal{G}}, X_H|w) = \frac{1}{Z} \exp\left(\sum_{l \in L} w_l n_l\right)$, where $w = \{w_l\}_{l \in L}$ with w_l being the global weight of rule $l \in L$, and n_l denotes the number of true groundings of rule l over observed and hidden triplets. Accordingly, we define the personalized rule importance score to be $y_{u,l} = \frac{w_l n_l(u)}{\sum_{l' \in L} n_{l'}(u)}$, where $n_l(u)$ is the number of groundings of rule l over the observed triplets in $\{(u, r_{ui}, v) \in \mathcal{G}\}$. However, it is intractable to directly maximize the log likelihood of observed triplets to learn the global weights w, i.e., $\max_w \log p(X_{\mathcal{G}}|w)$. Instead, we employ the EM algorithm to iteratively optimize the objective to acquire optimal global weights.

E-Step We introduce a mean-field variational distribution $q(X_H|\theta) \approx \prod_{(e_h, r, e_t) \in H} q(X_{hrt}|\theta)$ over hidden user–item interactions in H. The goal of the E-step is to estimate $q(X_H|\theta)$ by minimizing the KL divergence between $q(X_H|\theta)$ and the posterior distribution $p(X_H|X_{\mathcal{G}}, w)$ with fixed w. For each triplet $(e_h, r, e_t) \in H$, we denote by L_{hrt} the set of rules associated with the triplet and by G_{hrt} the corresponding groundings of all logical rules in L_{hrt}. Following [102], the optimal $q(X_H|\theta)$ can be achieved under the fixed-point condition, i.e., $q(X_{hrt}|\theta) \approx p(X_{hrt}|X_{G_{hrt}}, w)$, for all $(e_h, r, e_t) \in H$. Here, $q(X_{hrt}|\theta)$ is approximated by the KG encoder f_θ, and $p(X_{hrt}|X_{G_{hrt}}, w)$ can be estimated with the global weights w of the rules in L_{hrt} from the last iteration:

$$p(X_{hrt} = 1|X_{G_{hrt}}, w) = \sigma\left(\frac{\sum_{l \in L_{hrt}} w_l}{|L_{hrt}|}\right), \tag{4.1}$$

where $\sigma(\cdot)$ is the sigmoid function. In other words, if a hidden triplet (e_h, r, e_t) is asserted

to be true by the rules (e.g., $p(X_{hrt} = 1 \mid X_{G_{hrt}}, w) > 0.5$), the probability $q(X_{hrt} = 1 \mid \theta)$ given by the KG encoder is also expected to be high. Therefore, to learn the parameter θ, we aim to maximize the log-likelihood function over all observed triplets in $\mathcal{G}$ and the plausibly true hidden triplets in $H^+ = \{(e_h, r, e_t) \mid p(X_{hrt} = 1 | X_{G_{hrt}}, w) \geq \tau\}$, which leads to the objective

$$\ell(\theta) = \sum_{(e_h, r, e_t) \in \mathcal{G} \cup H^+} \log q(X_{hrt} = 1 \mid \theta), \tag{4.2}$$

where τ is a hyperparameter.

M-Step The goal of the M-step is to learn the global rule weights w by maximizing the log-likelihood function $E_{q(X_H)}[\log p(X_{\mathcal{G}}, X_H; w)]$ given a fixed θ from the E-step. Since the log-likelihood term models the joint distribution over all triplets, which is hard to compute for a large KG, we approximate it with the pseudolikelihood [4]: $\ell_{PL}(w) = \sum_{(e_h, r, e_t) \in \mathcal{G} \cup H} E_{q(X_H|\theta)}[\log p(X_{hrt} | X_{G_{hrt}}, w)]$. Then, we can invoke gradient ascent to acquire the optimal w, with the gradient defined as:

$$\begin{aligned} \nabla_{w_l} \ell_{PL}(w_l) = & \sum_{(e_h, r, e_t) \in \mathcal{G}} \frac{1 - p_{hrt}}{|L_{hrt}|} + \\ & \sum_{(e_h, r, e_t) \in H} \frac{q(X_{hrt} = 1 | \theta) - p_{hrt}}{|L_{hrt}|}, \end{aligned} \tag{4.3}$$

where $p_{hrt} = p(X_{hrt} = 1 | X_{G_{hrt}}, w)$. Once the optimal global weights are acquired, we can make a recommendation by calculating the ranking score of a user $u \in \mathcal{U}$ and an item $v \in \mathcal{I}$ as $q(X_{urv} | \theta) + \alpha\, p(X_{urv} = 1 | X_{G_{urv}}, w)$, where $r = r_{ui}$ and $\alpha \in \mathbb{R}$ is a hyperparameter.

4.3.3 Rule-Guided Path Reasoner

We draw on the KG encoder f_θ and the personalized rule importance scores y_u from the last two steps to generate explainable paths for every user u. Specifically, we train an LSTM-based path reasoning network ϕ that takes the start user embedding as input and predicts a sequence of entities and relations to form a path. For every user u, we restrict the reasoner

to generate the paths that follow the rules with the largest scores in y_u. The LSTM-based path reasoner ϕ is based on the graph walker in [89]. It takes as input the embedding of the current entity e_{t-1} and outputs the embeddings of the next relation r_t and the next entity e_t, i.e., $\mathbf{r}_t, \mathbf{e}_t = \phi(\mathbf{e}_{t-1})$. In particular, the next relation embedding $\mathbf{r}_t$ is defined as:

$$\alpha_t = \sigma(W_\alpha \mathbf{e}_{t-1} + b_\alpha),$$
$$\mathbf{r}_t = \sum_{r \in \mathcal{R}} \alpha_{t,r} r,$$

where W_α, b_α are parameters and α_t are the attention weights over all relations in the KG. The next entity embedding $\mathbf{e}_t$ is defined as:

$$\begin{aligned}
\mathbf{z}_t &= \mathbf{e}_{t-1} + \mathbf{r}_t \\
\mathbf{i}_t &= \sigma(W_i[\mathbf{e}_{t-1}; \mathbf{c}_{t-1}] + b_i) \\
\mathbf{c}_t &= (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(W_c[\mathbf{z}_t; \mathbf{e}_{t-1}] + b_c) \\
\mathbf{o}_t &= \sigma(W_o[\mathbf{z}_t, \mathbf{e}_{t-1}, \mathbf{c}_t] + b_o) \\
\mathbf{e}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

Here, $[;]$ denotes concatenation, $\odot$ is elementwise multiplication, i_t, o_t are vectors passing through corresponding gates, and z_t is the context vector.

During training, for every user and its observed user–item triplets, we sample a set of training paths following the rules, with numbers proportional to the rule weights. The goal is to make the path reasoner ϕ generate paths that are close to the training samples, which can be optimized by the hinge loss.

The inference pipeline using the trained path-reasoning network is described in Algorithm 3. Starting with a user u encoded as $\mathbf{e}_0 = \mathbf{u}$, the estimated entity embedding $\mathbf{e}_t$ and relation embedding $\mathbf{r}_t$ at the t-th hop is obtained by the model ϕ. At each hop, for all potential neighbors, we calculate a ranking score based on the dot-product of the neighbor

and estimated $(\mathbf{e}_t, \mathbf{r}_t)$. After ranking these neighbors based on such scores, we can filter a set of candidate neighbors and invoke a Beam Search to identify a set of paths as well as corresponding items for u.

Algorithm 3 Rule-guided path reasoning

Input: KG $\mathcal{G}$, user u, item v, rule set L
Output: a set of paths P
procedure MAIN()
    $P \leftarrow \{\{u\}\}$.
    for $t \leftarrow 1$ to T **do** ▷ T is path length.
        $P_{\text{curr}} \leftarrow \{\}$.
        for path $p \in P$ **do**
            $e_{t-1} \leftarrow$ last node of p.
            $V_{\text{curr}} \leftarrow \{\}$.
            for $(e_{t-1}, r', e') \in \mathcal{G}$ **do**
                $\hat{\mathbf{e}}_t, \hat{\mathbf{r}}_t = \phi(\mathbf{e}_{t-1})$
                $s = \langle \hat{\mathbf{e}}_t, \mathbf{e}' \rangle + \langle \hat{\mathbf{r}}_t, \mathbf{r}' \rangle$.
                $V_{\text{curr}} \leftarrow V_{\text{curr}} \cup \{(r', e', s)\}$.
            $P_{\text{curr}} \leftarrow P_{\text{curr}} \cup \{p \cup \{r', e'\} | \text{rank}(s) \leq \beta, (r', e', s) \in V_{\text{curr}}\}$.
        $P \leftarrow P_{\text{curr}}$.
    $P \leftarrow \{p | p \in P, \texttt{rule}(p) \in L, \texttt{lastnode}(p) = v\}$.
    return P.

4.3.4 Implementation Details

In order to guarantee path connectivity, we add reverse relations into the knowledge graph, i.e., if $(e_h, r, e_t) \in \mathcal{G}$, then $(e_t, r^{-1}, e_h) \in \mathcal{G}$. We restrict the length of candidate rules to be 3. We adopt TransE [5] as the KG encoder f_θ, with the dimensionality of entity and relation embeddings set as 100.

To learn the global rule weights, we first generate the hidden triplet set according to the result of the KG encoder. For each user, the top 50 estimated items with the highest scores predicted by KG encoder are taken as the hidden triplet set H^+. The threshold τ is set to 0.5 and the weighting factor α is set to 0.3 by default. In the path reasoning algorithm, we set the neighboring size β to 10. Other training details can be found in Table 4.1.

Parameter	**Cellphones**	**Grocery**	**Automotive**
# of epochs	4	2	3
KGE batch size	512	512	512
KGE optimizer	Adam	Adam	Adam
KGE learning rate	10^{-4}	10^{-4}	10^{-4}
NLM learning rate	10^{-5}	10^{-5}	10^{-5}
# of sample node	100	100	100

Table 4.1: Training detail for three datasets. KGE = KG encoder. NLM = neural logic model.

Dataset	**Cellphones**	**Grocery**	**Automotive**
#Users	61,254	57,822	95,445
#Items	47,604	40,694	78,557
#Interactions	607,673	709,280	1,122,776
#Entities	169,331	173,369	270,543
#Relations	45	45	73
#Triples	3,117,051	3,742,954	4,580,318

Table 4.2: Overall statistics of the datasets. We identify appreciated aspects of items from a user's historical records on Amazon for the user side and consider the following facts: item category, brand, price, listed features, and predefined styles for item meta-data.

4.4 Experiment

Dataset We experiment on three domain-specific e-commerce datasets from Amazon, namely *Cellphones*, *Grocery*, and *Automotive*. There are two requirements that lead to the selection of these categories in our experiments. First, the constructed KG should contain rich user behavior patterns, e.g., user mentioned features or preferred styles, etc. This is the major difference from most of the existing work [150], which only extends knowledge on the item side. Second, the KGs are assumed to be large-scale. We select several large subsets from [32], where the constructed KG can be regarded as an updated version of those of [2] based on the Amazon review dataset [94]. The remaining three datasets are the ones that satisfy both of the aforementioned requirements. The statistics of our datasets are shown in Table 4.2.

	Cellphones				Grocery				Automotive			
	Precision	Recall	NDCG	Hit	Precision	Recall	NDCG	Hit	Precision	Recall	NDCG	Hit
CKE	0.0360	0.1760	0.1847	0.3067	0.0612	0.2528	0.3070	0.4511	0.0458	0.1871	0.2257	0.3621
RippleNet	0.0419	0.2141	0.2177	0.3715	0.0591	0.2682	0.2858	0.4800	-	-	-	-
PGPR	0.0462	0.2148	0.2366	0.3801	0.0649	0.2710	0.3174	0.4926	0.5893	0.2315	0.2804	0.4409
KGAT	0.0476	0.2274	0.2365	0.3835	0.0702	0.2916	0.3381	0.5020	0.0601	0.2500	0.2859	0.4514
HeteroEmbed	0.0527	0.2543	0.2626	0.4226	0.0785	0.3316	0.3701	0.5572	0.0695	0.2923	0.3314	0.5082
LOGER	**0.0622**	**0.2977**	**0.3227**	**0.4808**	**0.0906**	**0.3754**	**0.4370**	**0.6121**	**0.0743**	**0.3091**	**0.3653**	**0.5346**

Table 4.3: Recommendation performance of all methods on four proposed datasets. The results are computed based on the top-10 recommendation on the test set. The best results are highlighted in bold and the second best results are underlined.

Baselines & Metrics We consider several state-of-art baselines in the following experiments. **CKE** [145] uses semantic representations derived from TransR [78] to enhance the matrix factorization process. **RippleNet** [125] is a hybrid method combining regularization and path formats, and augmenting user representations with a memory-network-like approach. **PGPR** [136] designed a policy-guided graph search algorithm for recommendation over KGs. **HeteroEmbed** [1] aims to learn the embeddings of a heterogeneous graph including users, items, and relations for recommendation. **KGAT** [129] explicitly models higher-order KG connectivity and learns node representations by propagating the embedding of neighbors with corresponding importance discriminated by an attention mechanism. We adopted the same metrics as [1] to evaluate the recommendation performance of all models: **Precision**, **Recall**, Normalized Discounted Cumulative Gain (**NDCG**), and Hit Rate (**HR**).

4.4.1 Recommendation Results

We first evaluate the recommendation quality of our model. The results of all methods across all three datasets are reported in Table 4.3. In general, our method significantly outperforms all state-of-the-art baselines on all metrics. Taking *Cellphones* as an example, our method achieves an improvement of 6.01% in NDCG against the best baseline (underlined), and an improvement of 5.82% in Hits@10. Similar trends can be observed on other benchmarks as well. Note that both our model and HeteroEmbed adopt TransE for

KG representation learning, yet our model achieves better performance, mainly attributed to the iterative learning of graph encoder and neural logic model.

4.4.2 Faithfulness of Explanation

We aim to measure whether the generated explainable paths are consistent with the historic user behavior via a faithfulness metric and a user study.

Measuring Faithfulness Inspired by previous work [84, 116, 120], we define the faithfulness to be the Jensen–Shannon (JS) divergence of rule-related distributions from training and test sets. Specifically, we randomly sample 50 users from the training set. For each user u, we further sample around 1,000 paths between the user and the connected item nodes, and calculate the rule distribution over these paths, denoted by $F(u)$. We compare the proposed LOGER with two baselines, PGPR, and KGAT, each of which is used to generate 20 explainable paths for every selected user in the test phase. Similarly, we can calculate the rule distribution over these 20 paths, denoted by $Q_f(u)$. The JS scores are defined as follows.

$$\mathrm{JS}_f = \mathbb{E}_{u\sim\mathcal{U}}[D_{\mathrm{JS}}(Q_f(u) \,\|\, F(u))]$$
$$\mathrm{JS}_w = \mathbb{E}_{u\sim\mathcal{U}}[D_{\mathrm{JS}}(Q_w(u) \,\|\, F(u))]$$

Here, $Q_w(u)$ is the rule weight distribution derived from the personalized rule importance scores of our method or the path weights of baselines. Smaller values of two JS scores correspond to better faithfulness of the explainable paths. This faithfulness evaluation is motivated in terms of the consistency of the explainable paths with respect to the user historic behavior.

User Study Additionally, we conduct a user study to evaluate the faithfulness of the explainable paths. We display 50 sampled KG paths starting from one user towards purchased items in the training set to represent examples of user historical behaviors. For compari-

	Cellphones & Accessories			Grocery & Gourmet		
	JS_f	JS_w	Avg. Rank	JS_f	JS_w	Avg. Rank
PGPR	0.56	0.49	2.52	0.42	0.38	2.27
KGAT	0.53	0.45	2.14	0.39	0.41	2.08
LOGER	**0.47**	**0.32**	**1.52**	**0.34**	**0.28**	**1.75**

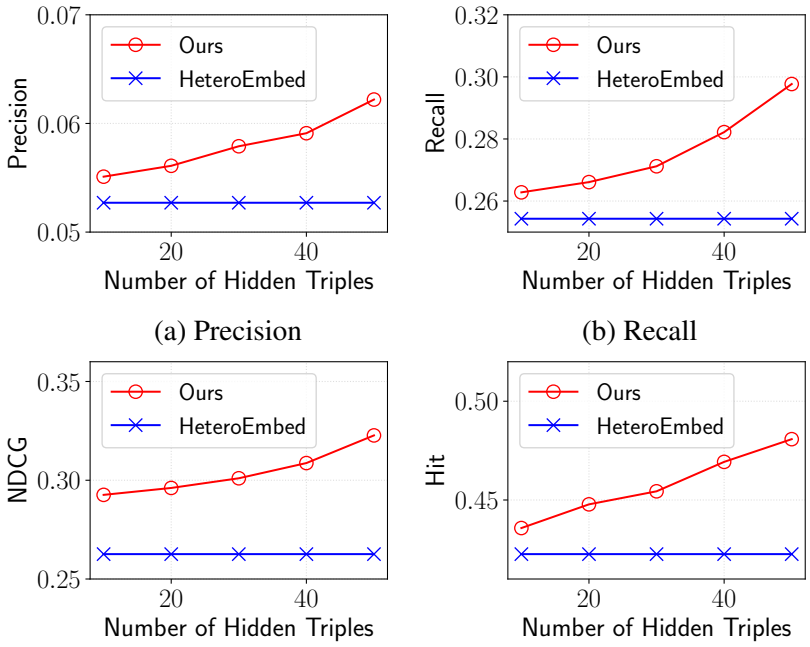
Table 4.4: 20 testers are asked to rank three groups of paths in ascending order. We calculate corresponding averaged rank scores. Bolded number are used to label the best performance.

son, we also present 10 explainable paths generated by three methods for the same user in the test dataset. We ask 20 human subjects to rank these methods based on whether the generated paths are consistent with those from the training set. Then, we calculate the average ranking scores (**Avg. Rank**) by averaging the rank given by each human tester on each method.

Results The results on the *Cellphones* and *Grocery* datasets are reported in Table 4.4. We observe that our method LOGER achieves the lowest JS scores and average ranking score, which reveal the effectiveness of our model in producing more faithful explanations in both quantitative measurements and in the user study.

4.4.3 Ablation Study.

We further study how hidden triplets used in training KG encoder (Equation 4.2) influence the recommendation performance. We experiment on the *Cellphones* data under different sizes of hidden triplet sets H^+. We choose the sizes of $\{10, 20, 30, 40, 50\}$ and keep all other settings unchanged. The results are plotted in Figure 4.2, including our model (red circles) and the best baseline HeteroEmbed (blue crosses). We find that our model consistently outperforms the baseline in all the metrics under different numbers of hidden triplets. Better recommendation performance can be achieved with more hidden triplets included in training the KG encoder, because more candidate items will enhance the capability of our model to discern the logical rules of good quality and hence benefit the recommendation



(a) Precision (b) Recall

(c) NDCG (d) Hit

Figure 4.2: Recommendation performance with varying sizes of estimated hidden triples.

prediction.

4.5 Conclusion

In this chapter, we propose LOGER for faithfully explainable recommendation, which generates explainable paths based on personalized rule importance scores via neural logic reasoning that adequately captures historic user behavior. We experiment on three large-scale datasets for e-commerce recommendation showing superior recommendation quality of LOGER as well as the faithfulness of the generated explanations both quantitatively and qualitatively. We hope to encourage future work that values explainability and in particular the faithfulness of explanations.

CHAPTER 5
INVERSE REINFORCEMENT GRAPH REASONING

Column annotation, the process of annotating tabular columns with labels, plays a fundamental role in digital marketing data governance. It directly impacts how customers manage their data and ensures compliance with regulations, restrictions, and policies applicable to data use. Despite substantial gains in performance brought by recent deep learning-driven column annotation methods, the incapability of explaining why certain columns are matched to the target labels has drawn concern due to the black-box nature of deep neural networks. Such explainability is of particular importance in industrial marketing scenarios, where data stewards[1] need to quickly verify and calibrate the annotation results to guarantee the correctness of downstream applications. This work sheds new light on the explainable column annotation problem, which is the first of its kind column annotation task. To achieve this, we propose a new approach called EXACTA, which conducts knowledge graph reasoning using inverse reinforcement learning to find a multi-hop path from a column to a potential target label while ensuring both annotation performance and explainability. We experiment on four open-source and real industrial benchmarks, and undertake a comprehensive analysis on explainability. The results suggest that our method not only provides competitive annotation performance compared with existing deep learning-based models, but more importantly, produces faithfully explainable paths of annotated columns to facilitate human examination.

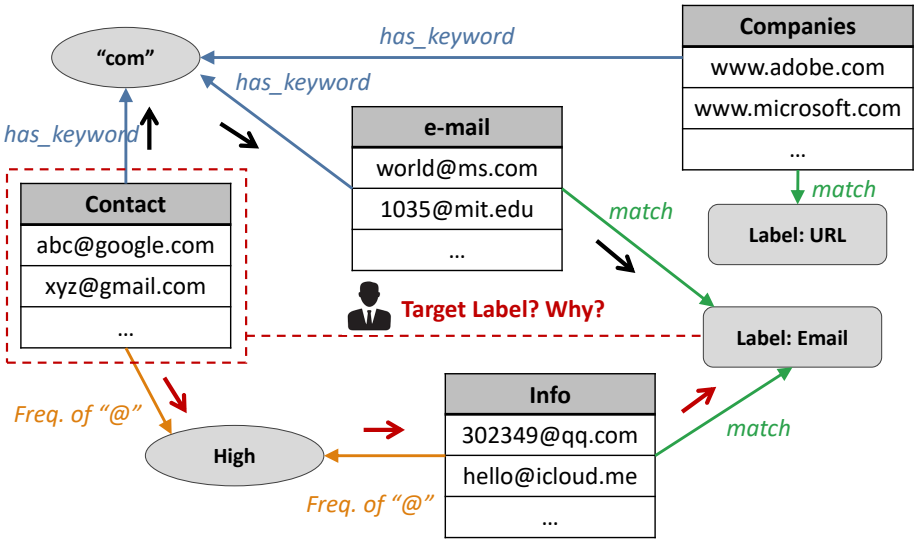


Figure 5.1: Explainable column annotation via multi-step reasoning over a KG to find the target label "Email" for the source column "Contact" with the accompanying explainable paths marked by red and black arrows.

5.1 Introduction

Structured tabular data are commonly acknowledged as a convention for recording relational information with wide application in data management systems and can be consumed to develop business insights and benefit decision-making in industry. One important task in the early stages of a tabular data analysis pipeline is called *column annotation* [76, 99, 10, 53], which aims to match table columns to annotation labels. Take the digital marketing industry as an example: annotated columns usually serve as the input of downstream tasks and a missing or false annotation of a personally identifiable information (PII) column may cause a severe privacy leakage. To comply with the General Data Protection Regulation (GDPR) [35], industry experts such as data stewards and marketers are required to manually verify the correctness of the labeling results, which usually incurs enormous costs in both time and effort. To accelerate the human evaluation process, automated labeling tools are preferred, which ought to obtain good accuracy in column annotation while providing the explanation to the experts to justify why such decision is made. This is the task of

[1]Data stewards are the heart of data governance who are responsible for interpreting regulations, contractual restrictions, and policies and applying them directly to data.

explainable column annotation. Existing approaches are mostly accuracy-driven, seeking superior annotation performance via modern deep neural networks [10, 11, 53, 142]. However, they have critical limitations in real industry scenarios due to the lack of explainability and trustworthiness for the experts who conduct manual verification.

To meet both requirements, knowledge graph (KG) reasoning [69], known as an explainable predictor, has been widely adopted in link prediction [139, 77], medical diagnosis [109], and recommendation [136], etc. The benefits are that it can not only achieve competitive performance with deep neural networks but also generate explainable KG paths that allow tracing back the entire decision-making process. Motivated by this, in this work, we explore approaching the task of explainable column annotation under the framework of KG reasoning. Specifically, a KG is built over all columns, annotations, and their extracted features. Then, a model iteratively traverses this graph from a starting column node towards candidate label nodes. The inferred paths directly reflect the reasoning and decision making and hence can serve as an explanation for the prediction. As an example, illustrated in Figure 5.1, the column "*Contact*" is expected to be matched with the label "*Email*", which can be reached by navigating along an evidence path: *"Contact"* $\xrightarrow{\textit{freq. of "@"}}$ *"com"* $\xrightarrow{\textit{freq. of "@"}}$ *Column "Info"* $\xrightarrow{match}$ *"Email"*.

To achieve this, recent works primarily rely on reinforcement learning (RL) techniques [122, 139, 77], i.e., a policy network is first learned over a KG-based Markov decision process and then used to conduct multi-step reasoning from an unlabeled source node to a potential target node as the prediction. However, the major challenge of these methods lies in the manual definition of the reward function: it is easy to specify relevance between sources and targets, but very hard to quantify the explainability of intermediate nodes. This may result in spurious paths that do not genuinely confer explainability of the column annotation results. For instance, there are two possible paths between the column "*Contact*" and label "*Email*" in Figure 5.1, respectively marked by red and black arrows. But the black-arrowed path is intuitively less explainable than the other one due to the keyword

"com" being too vague to the prediction. If we only define a sparse reward on the last-step of Label "*Email*", the RL agent is very likely to opt for the less explainable path, since both paths result in the same cumulative rewards. To avoid handcrafting rewards, the alternative solution is to leverage imitation learning (IL), which requires high-quality trajectories to learn the policy via supervised learning or behavior cloning [54, 154, 93]. In the real industry, however, it is easier to acquire large-scale noisy path data from less-skilled crowdsourced workers than carefully labeled data from experts. As a trade-off, the quality of these paths may be unsatisfactory, e.g., due to the lack of knowledge of business scenarios, the crowdsourced workers may choose less explainable paths than those preferred by the experts. Thus, the conventional IL methods are not adapted for training the graph traversal model, as they do not explicitly tolerate noise in the inputs and may lead to suboptimal choices of paths.

To this end, we propose a novel KG reasoning method EXACTA for EXplAinable Column anoTAtion based on the framework of inverse reinforcement learning (IRL). Given a set of noisy explainable paths between columns and labels, EXACTA first learns a noise-tolerant reward function from the paths, which is then facilitated to guide policy learning so that policy-based KG walker can generate both accurate predictions and explainable paths via multi-hop reasoning. Our method is specially designed to cope with industry-scale column annotation tasks, as it does not require high-quality expert paths to train the policy, but can still guarantee to produce faithfully explainable paths for the predicted column annotations. We experiment on four open-public and real industrial benchmarks and the results not only demonstrate better column annotation performance compared to various baselines, but more importantly show better explainability provided by our model. The following four aspects highlight our contributions.

- To the best of our knowledge, this is the first work formally studying the problem of explainable column annotation. We articulate the importance of this problem in industrial marketing data management pipelines.

- We propose a novel KG reasoning approach EXACTA based on IRL that can automatically learn a noise-tolerant reward function from noisy input paths to guide the policy learning.
- We experiment on four offline open-public and industrial benchmarks and conduct an online simulation of explainable column annotation, observing promising results of EXACTA in both annotation performance.
- We also systematically evaluate the explainability of our model in terms of perceived explainability, robustness, and faithfulness of the path-based explanations.

5.2 Preliminaries

5.2.1 Problem Formulation

We consider the explainable column annotation problem in the framework of knowledge graph reasoning. Formally, given an entity set $\mathcal{E}$ and a relation set $\mathcal{R}$, a knowledge graph (KG) for column annotation, denoted by $\mathcal{G}$, is defined to be a set of triples, $\mathcal{G} = \{(e, r, e') \mid e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where each triple represents a fact between a head entity e and a tail entity e' via relation r. There are two special subsets of entities in the KG, namely "columns" $\mathcal{X} \subseteq \mathcal{E}$ and "labels" $\mathcal{Y} \subseteq \mathcal{E}$, and the relation connecting them, denoted by $r_{\text{match}} \in \mathcal{R}$, means a column is matched (or annotated) with a label. We assume that each column can only be matched with one correct label via relation r_{match}. The remaining entities in $\mathcal{E} \setminus \{\mathcal{X} \cup \mathcal{Y}\}$ stand for the explainable features extracted from columns and labels such as keywords and statistical values, and the relations in $\mathcal{R} \setminus \{r_{\text{match}}\}$ reflect the *has-a* property of columns and labels with respect to these features. For instance, a triple *("Contact", has_keyword, "com")* expresses that the column *"Contact"* $\in \mathcal{X}$ has the property of being associated with the *keyword* (i.e., relation *has_keyword*$\in \mathcal{R} \setminus \{r_{\text{match}}\}$) with value *"com"* $\in \mathcal{E} \setminus \{\mathcal{X} \cup \mathcal{Y}\}$. The details of the KG construction process are described in the Appendix. By taking advantage of the rich heterogeneous information and relational graph structure in the KG, we are interested in predicting (i) the missing links of relation

r_{match} for unmatched columns, and (ii) an explanation for the matching decision. In this work, we define an explanation for the column–label pair (x, y) to be a KG reasoning path, which is a sequence of entities and relations, denoted by $L = \{e_0, r_1, e_1, \ldots, e_{l-1}, r_l, e_l \mid e_0 = x, e_l = y, l \in \mathbb{R}_+, (e_{t-1}, r_t, e_t) \in \mathcal{G} \forall t \in [|L|]\}$. The KG-enhanced explainable column annotation problem is formulated as follows.

Definition 5.2.1. (*Explainable Column Annotation*) Given a KG $\mathcal{G}$, the goal is, for every unmatched column entity $x \in \mathcal{X}$, to predict a label entity $y \in \mathcal{Y}$ along with a reasoning path L of nodes from x to y that serves as the explanation for the annotation (x, y).

The challenges of this problem are threefold.

- *Faithful Explanation.* The explanation is required to be faithful to the decision-making process, which means the reasoning path should reflect the actual multi-hop inference process of the model, and the visited nodes along the paths should be the genuine causes of the model's annotation outcome.
- *Unknown Target.* Since the annotation results are derived with the path-finding process, the target node is **unknown** prior to the KG reasoning, which makes it hard for the agent to determine if the next step will potentially lead to a "correct" label node.
- *Noisy Paths.* Since the input paths are noisy and not warranted to be the most explainable, a good solution should explicitly model such noise and be robust to the diverse quality of the explainable paths, so that it can find optimal paths in the inference step.

5.2.2 (Inverse) Reinforcement Learning

A finite Markov Decision Process (MDP) in reinforcement learning (RL) is defined as a tuple $(\mathcal{S}, \mathcal{A}, p(s_{t+1}|s_t, a_t), R(s_t, a_t))$ with state $s_t \in \mathcal{S}$, action $a_t \in \mathcal{A}$, transition probability $p(s_{t+1}|s_t, a_t)$ and reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ for each step $t \in [T]$. The goal of RL is to find a policy $\pi(a_t|s_t)$ over the MDP that maximizes the expected cumulative reward, i.e., $\mathbb{E}_{\tau \sim p_\pi(\tau)}[\sum_{(s_t, a_t) \in \tau} R(s_t, a_t)]$, where $\tau = \{s_1, a_1, s_2, \ldots, s_T, a_T, s_{T+1}\}$ denotes a trajectory

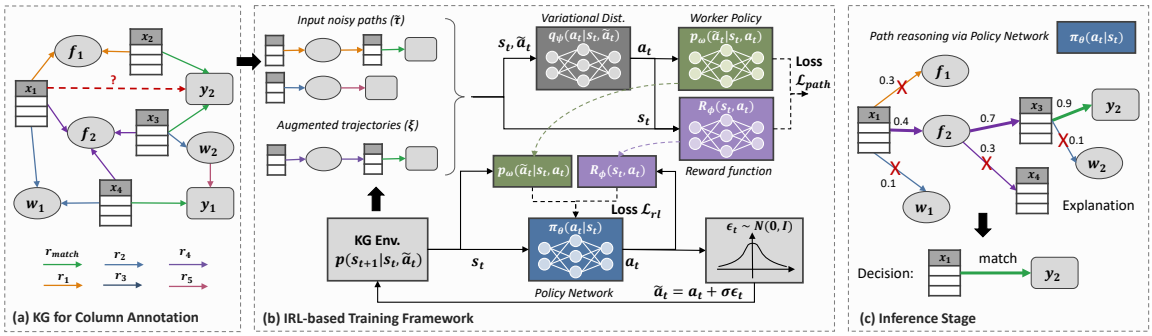


Figure 5.2: Pipeline of the proposed EXACTA. (a) A KG is constructed with columns, labels, and explainable features. (b) Given noisy paths, the reward function and policy network are iteratively learned under the IRL framework with explicit noise modeling via worker policy. (3) KG reasoning is conducted to generate an explainable path and predicted label for the column.

sampled from the distribution $p_\pi(\tau) = p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$. The discounting factor is ignored for simplicity.

One limitation of RL occurs when the reward function is unavailable and hard to define in practice [93], so an alternative solution is to adopt the notion of inverse reinforcement learning (IRL), which aims to learn the reward function from expert trajectories $\mathcal{D} = \{\tau\}$. The common approach to the IRL problem is under the framework of maximum-entropy IRL (ME-IRL) [154], which models the probability of a trajectory by the maximum entropy principle [55], i.e., $p_\phi(\tau) = \frac{1}{Z_\phi} p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)e^{R_\phi(s_t,a_t)}$, where $R_\phi(s_t, a_t)$ is the estimated reward function parametrized by ϕ and Z_ϕ is the partition function defined over all possible trajectories. ME-IRL learns ϕ by maximizing the log-likelihood of $p_\phi(\tau)$ over all training trajectories, which leads to the following optimization problem: $\max_\phi \ \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\sum_{(s_t,a_t)\in\tau} R_\phi(s_t, a_t) - \log Z_\phi$. Once the reward function R_ϕ is derived, we can further learn the policy π over the MDP with the estimated rewards.

5.3 Our Method

In this section, we propose a new method based on ME-IRL called EXACTA, which iteratively learns the reward function from noisy explainable paths and the policy with the

estimated rewards, so that the agent is able to find high-quality explainable paths leading to the correct target labels for the given source columns.

5.3.1 Formulation as IRL Problem

We start by casting explainable column annotation as an IRL problem with noisy paths as input.

KG-based MDP At each step $t \in [T]$, the state s_t is defined to be the joint representation of the starting column $x \in \mathcal{X}$ and the current entity $e_{t-1} \in \mathcal{E}$, i.e., $s_t = (x, e_{t-1})$, with the initial state $s_1 = (x, x)$.[2] The valid action space of the state s_t consists of all outgoing edges of the current entity, $\mathcal{A}_t = \{(r_t, e_t) \mid (e_{t-1}, r_t, e_t) \in \mathcal{G}\}$. We also add a special STOP action that allows the agent to stop at the current entity. Given the state s_t and an action $a_t = (r_t, e_t) \in \mathcal{A}_t$, the transition probability is $p(s_{t+1}|s_t, a_t) = 1$ if $s_{t+1} = (x, e_t)$ and 0 otherwise. A KG path $L = \{x, r_1, e_1, \ldots, e_{T-1}, r_T, y\}$ can be identically converted to a trajectory $\tau = \{s_1 = (x, x), a_1 = (r_1, e_1), s_2 = (x, e_1), \ldots, s_T = (x, e_{T-1}), a_T = (r_T, y), s_{T+1} = (x, y)\}$, so we will use them interchangeably. This definition of MDPs over KGs is commonly adopted by existing works [139, 77], however, its limitation is also obvious, i.e., node out-degrees determine the sizes of the discrete action space, which may vary significantly among different states and become space-inefficient in policy network implementation. Therefore, we reformulate the MDP in continuous space by vectorizing each entity and relation by means of pretrained KG embeddings[3], with state $s_t = [\mathbf{x}; \mathbf{e}_{t-1}] \in \mathcal{S} \subseteq \mathbb{R}^{2d}$ and action $a_t = [\mathbf{r}_t; \mathbf{e}_t] \in \mathcal{A} \subseteq \mathbb{R}^{2d}$, for $t \in [T]$. Here d is the dimensionality of entity and relation embeddings and $[;]$ denotes concatenation. The actual reward is unknown and expected to be estimated by the function $R_\phi(s_t, a_t)$.

IRL-based Objective Given a set of noisy and less explainable paths of correctly labeled

[2] More historical information can be encoded into the state, which can be easily extended under our framework and is not the focus of this work.

[3] We adopt TransE [5] in this work, which can be replaced by other KG embeddings.

columns, we aim to learn the reward function R_ϕ that indicates both the explainability of moving to the next node and the probability of arriving at the correct label eventually. Then, the optimal policy $\pi(a_t|s_t)$ can be learned via RL with the help of the estimated rewards by R_ϕ.

To explicitly model the behavior of crowdsourced workers, we assume each noisy trajectory $\tilde{\tau} = \{s_1, \tilde{a}_1, s_2, \ldots, s_T, \tilde{a}_T, s_{T+1}\}$ is generated following a worker policy with Gaussian noise $p_\omega(\tilde{a}_t|s_t, a_t) = N(\tilde{a}_t|a_t, \Sigma_\omega(s_t))$. That is, each noisy action $\tilde{a}_t \in \mathcal{A}$ is sampled from a multivariate Gaussian distribution with a mean of the optimal action a_t and a state-dependent covariance $\Sigma_\omega(s_t) = \mathrm{diag}(\mathrm{FF}_\omega(s_t))$, which is a diagonal matrix with values approximated by a multi-layer feed-foward neural network FF_ω with parameters ω.

Accordingly, the probability of the noisy trajectory $\tilde{\tau}$ is $p(\tilde{\tau}) = p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, \tilde{a}_t)p(\tilde{a}_t|s_t)$, which can be rewritten by p_ω as:

$$p(\tilde{\tau}) = p(s_1)\prod_{t=1}^{T}\int_{a_t\in\mathcal{A}} \pi(a_t|s_t)p_\omega(\tilde{a}_t|s_t, a_t)da_t, \tag{5.1}$$

where $p(s_1)$ is the constant probability of the initial state, and the transition probability $p(s_{t+1}|s_t, \tilde{a}_t) = 1$ is ignored in Equation 5.1. Inspired by ME-IRL [154], we also model the probability $p(\tilde{\tau})$ with the reward function $R_\phi(s_t, a_t)$ under the maximum-entropy principle:

$$p_{\phi,\omega}(\tilde{\tau}) = \frac{1}{Z_{\omega,\phi}}\, p(s_1)\prod_{t=1}^{T}\int_{a_t} e^{R_\phi(s_t,a_t)} p_\omega(\tilde{a}_t|s_t, a_t)da_t, \tag{5.2}$$

where $Z_{\omega,\phi}$ is the partition function defined over all trajectories with fixed horizon T. By plugging $p_\omega(\tilde{a}_t|s_t, a_t) = N(\tilde{a}_t|a_t, \Sigma_\omega(s_t))$ into Equation 5.2, we can simplify the model as follows:

$$p_{\phi,\omega}(\tilde{\tau}) = \frac{1}{Z_{\omega,\phi}}\, p(s_1)\prod_{t=1}^{T}\int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}da_t, \tag{5.3}$$

$$f_{\phi,\omega}(s_t, a_t, \tilde{a}_t) = R_\phi(s_t, a_t) - \frac{1}{2}\left(D^2_\omega(\tilde{a}_t) + \log\det(\Sigma_\omega(s_t))\right) \tag{5.4}$$

Here, $D_\omega(\tilde{a}_t) = \sqrt{(\tilde{a}_t - a_t)^\top \Sigma_\omega^{-1}(s_t)(\tilde{a}_t - a_t)}$ is the Mahalanobis distance between $\tilde{a}_t$ and $\tilde{a}$. The constant term in the Gaussian distribution is disregarded, as it will not be used in the optimization. Equation 5.3 implies that the higher the quality of an explainable path, the more rewards it is likely to confer to the agent.

Let $\tilde{\mathcal{D}} = \{\tilde{\tau}\}$ be the set of input noisy trajectories. We can learn the reward function R_ϕ and the worker policy p_ω by maximizing the log-likelihood of $p_{\phi,\omega}(\tilde{\tau})$ over all trajectories in $\tilde{\mathcal{D}}$:

$$\max_{\phi,\omega} \frac{1}{|\tilde{\mathcal{D}}|} \sum_{\tilde{\tau}\in\tilde{\mathcal{D}}} \sum_{t=1}^{T} \log \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} da_t - \log Z_{\phi,\omega}. \tag{5.5}$$

5.3.2 Training Framework

However, directly solving Equation 5.5 is intractable in practice due to the integral over the large-scale continuous action space. We adopt the variational approach [57] to change the integral into an expectation by introducing additional variational distributions.

Specifically, for the first logarithm term in Equation 5.5, we introduce a variational distribution $q_\psi(a_t|s_t, \tilde{a}_t)$ approximated by a neural network parametrized by ψ. For each noisy path $\tilde{\tau} \in \tilde{\mathcal{D}}$, we have

$$\begin{aligned}
&\sum_{t=1}^{T} \log \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} da_t = \sum_{t=1}^{T} \log \mathbb{E}_{a_t\sim q_\psi} \left[\frac{e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}}{q_\psi(a_t|s_t,\tilde{a}_t)} \right] \\
&\geq \sum_{(s_t,\tilde{a}_t)\in\tilde{\tau}} \mathbb{E}_{a_t\sim q_\psi} \left[f_{\phi,\omega}(s_t,a_t,\tilde{a}_t) - \log q_\psi(a_t|s_t,\tilde{a}_t) \right] \qquad (5.6) \\
&= \mathcal{L}_{\text{path}}(\phi,\omega,\psi;\tilde{\tau}),
\end{aligned}$$

where Equation 5.6 is derived via Jensen's inequality and the expectation can be easily approximated via Monte Carlo methods [110]. Note that $\sum_{t=1}^{T} \log \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} da_t = \max_\psi \mathcal{L}_{\text{path}}(\phi,\omega,\psi;\tilde{\tau})$. Therefore, we can first estimate ψ by maximizing the objective

$\mathcal{L}_{\text{path}}$ and then solve the original problem in Equation 5.5, which is equivalent to

$$\max_{\phi,\omega} \frac{1}{|\tilde{\mathcal{D}}|} \sum_{\tilde{\tau}\in\tilde{\mathcal{D}}} \mathcal{L}_{\text{path}}(\phi, \omega, \hat{\psi}; \tilde{\tau}) - \log Z_{\phi,\omega}, \tag{5.7}$$

where $\hat{\psi} = \arg\max_{\psi} \mathcal{L}_{\text{path}}(\phi, \omega, \psi; \tilde{\tau})$.

Then, we can approach the partition function in Equation 5.7 in a similar way, which can first be expanded as follows.

$$\begin{aligned} \log Z_{\phi,\omega} &= \log \int_{\tilde{\tau}\in(\mathcal{S}\times\mathcal{A})^T} \left(p(s_1) \prod_{t=1}^{T} \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} da_t \right) d\tilde{\tau} \\ &= \log \int_{\xi\in(\mathcal{S}\times\mathcal{A}\times\mathcal{A})^T} p(s_1) \prod_{t=1}^{T} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} d\xi \end{aligned} \tag{5.8}$$

Here, $\xi = \{s_1, a_1, \tilde{a}_1, \ldots, s_T, a_T, \tilde{a}_T, s_{T+1}\}$ denotes the augmented trajectory with paired actions $\{(a_t, \tilde{a}_t)\}_{t\in[T]}$. To simulate the generation process of augmented trajectory ξ, we assume that the optimal action a_t is first sampled from the policy network $\pi_\theta(a_t|s_t)$ and then the noisy action $\tilde{a}_t$ is sampled from another Gaussian noise distribution $N(\tilde{a}|a_t, \sigma^2 I)$. Note that we cannot adopt the worker policy p_ω here to sample noisy action $\tilde{a}$, as it is based on the assumption of crowdsourced workers generating noisy paths, while here it corresponds to a different MDP setup with paired actions. To simplify the computation of $\tilde{a}_t$ in practice, we can sample Gaussian noise $\epsilon_t \sim N(0, I)$ and then compute $\tilde{a}_t = a_t + \sigma\epsilon_t$, which is known as the reparameterization trick [60]. Accordingly, the probability of ξ is $p_\theta(\xi) = p_0 \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t + \sigma\epsilon_t)\, \pi_\theta(a_t|s_t)\, N(\epsilon_t|0, I)$.

Thus, we can introduce the distribution $\pi_\theta(a_t|s_t)N(\epsilon_t|0, I)$ into Equation 5.8 to elimi-

nate the integral via the variational approach:

$$\log Z_{\phi,\omega} = \log \mathbb{E}_{\xi\sim p_\theta}\left[\prod_{t=1}^{T}\frac{e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}}{\pi_\theta(a_t|s_t)N(\epsilon_t|0,I)}\right] \tag{5.9}$$

$$\geq \mathbb{E}_{\xi\sim p_\theta}\left[\sum_{t=1}^{T} f_{\phi,\omega}(s_t,a_t,\tilde{a}_t) - \log\pi_\theta(a_t|s_t) + \frac{1}{2}\|\epsilon_t\|^2\right] \tag{5.10}$$

$$= \mathbb{E}_{\xi\sim p_\theta}\left[\sum_{t=1}^{T} R_\phi(s_t,a_t) - \log\pi_\theta(a_t|s_t)\right] + g_\omega = \mathcal{L}_{\mathrm{rl}}(\phi,\omega,\theta),$$

where $g_\omega = \mathbb{E}_\xi[\sum_{t=1}^{T}\sigma^2\mathrm{tr}(\Sigma_\omega^{-1}(s_t)) + \log\det(\Sigma_\omega^{-1}(s_t))]$ is a regularization term on parameter ω. $\mathbb{E}_{\xi\sim p_\theta}[\sum_{t=1}^{T}\frac{1}{2}\|\epsilon_t\|^2]$ in Equation 5.10 is a constant based on the quadratic form of random variables and is ignored. It is easy to find that $\log Z_{\phi,\omega} = \max_\theta \mathcal{L}_{\mathrm{rl}}(\phi,\omega,\theta)$, which results in an RL problem for which we can learn the policy π_θ over the MDP with paired action $(a_t,\tilde{a}_t)$ and the estimated reward $R_\phi(s_t,a_t)$. Hence, another benefit of this formulation is that we can adopt any state-of-the-art RL algorithm (e.g., PPO [115] in this work) to learn the optimal policy $\pi_{\hat{\theta}}(a_t|s_t)$ with $\hat{\theta} = \arg\max_\theta \mathcal{L}_{\mathrm{rl}}(\phi,\omega,\theta)$.

The ultimate goal is to maximizing the following objective:

$$\mathcal{L}(\phi,\omega;\tilde{\mathcal{D}}) = \frac{1}{|\tilde{\mathcal{D}}|}\sum_{\tilde{\tau}\in\tilde{\mathcal{D}}}\mathcal{L}_{\mathrm{path}}(\phi,\omega,\hat{\psi};\tilde{\tau}) - \mathcal{L}_{\mathrm{rl}}(\phi,\omega,\hat{\theta}). \tag{5.11}$$

The complete training pipeline is summarized in Algorithm 4.

5.3.3 Inference

In the inference stage, we leverage the learned policy network $\pi_\theta(a_t|s_t)$ to walk over the KG from an unmatched column step-by-step towards a potential label node. Specifically, given a $(t-1)$-hop path $L_{t-1} = \{x, r_1, e_1, \ldots, r_{t-1}, e_{t-1}\}$, we first obtain the mean action vector $\hat{\mathbf{a}}_t = \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ by taking as input the state vector $\mathbf{s}_t = [\mathbf{x};\mathbf{e}_{t-1}]$. Then we select top

Algorithm 4 Training Pipeline

Input: KG $\mathcal{G}$, noisy paths $\{L\}$.
Output: reward function R_ϕ, policy network π_θ
Convert paths $\{L\}$ to trajectories $\tilde{\mathcal{D}}$ with pretrained TransE.
Initialize policy network $\pi_\theta(a_t|s_t)$, reward function $R_\phi(s_t, a_t)$.
Initialize FF_ω and variational distribution $q_\psi(a_t|s_t, \tilde{a}_t)$.
for epoch $n = 1 \ldots, N$ **do**
    Update q_ψ with gradient $\sum_{\tilde{\tau}\in\tilde{\mathcal{D}}} \nabla_\psi \mathcal{L}_{\mathrm{path}}(\phi, \omega, \psi; \tilde{\tau})$.
    for $i = 1, \ldots, m$ **do** ▷ Sample m augmented trajectories
        Initialize $\xi_i = \{s_1 = [\mathbf{x}; \mathbf{x}]\}$, for random $x \in \mathcal{X}$.
        for $t = 1, \ldots, T$ **do**
            Sample $a_t \sim \pi_\theta(a_t|s_t)$, $\epsilon_t \sim N(0, I)$, $\tilde{a}_t = a_t + \sigma\epsilon_t$
            Sample $s_{t+1} \sim p(s_{t+1}|s_t, \tilde{a}_t)$ and add $(a_t, \tilde{a}_t, s_{t+1})$ to ξ_i.
    Compute $\mathcal{L}_{\mathrm{rl}}(\omega, \phi, \theta)$ with $\{\xi_i\}_{i\in[m]}$.
    Update π_θ via PPO with R_ϕ. ▷ Solve RL problem
    Update p_ω with gradient $\nabla_\omega \mathcal{L}(\omega, \phi; \tilde{\mathcal{D}})$.
    Update R_ϕ with gradient $\nabla_\phi \mathcal{L}(\omega, \phi; \tilde{\mathcal{D}})$. ▷ Learn rewards
return R_ϕ, π_θ

k outgoing edges by calculating

$$\{(r_t, e_t) \in \mathcal{G} \mid e_t \notin L_{t-1}, \mathrm{rank}(\|\hat{\mathbf{a}}_t - [\mathbf{r}_t; \mathbf{e}_t]\|_2^2) \leq k\}. \tag{5.12}$$

By extending the path L_{t-1} with these k edges, we obtain k t-hop paths, each of which is further extended until the length reaches the horizon T. At the end, at most $O(k^T)$ paths will have been generated and we rank them according to the cumulative rewards by R_ϕ along each path. The top-ranked reasoning path is expected to end with the correct label and the nodes along the path faithfully explain the decision-making process.

5.3.4 Implementation Details

In this section, we describe the implementations of our method including knowledge graph construction and model training details.

Knowledge Graph Construction

Since we adopt KG reasoning paths as explanations for the column annotation predictions, it is required that each node in the graph must be understandable to a human. Therefore, for KG construction, we extract the following four groups of useful and comprehensible features from columns and labels.

Cell-level statistics Inspired by the previous work [53], we extract 27 global statistical features from each column, including the number of non-empty cell values, the entropy of cell values, fraction of {unique values, numerical characters, alphabetical characters}, {mean, std. dev.} of the number of {numerical characters, alphabetical characters, special characters, words}, {percentage, count, any, all} of the missing values, and {sum, min, max, median, mode, kurtosis, skewness, any, all} of the length of cell values. The values of these features are real-valued numbers, and hence for each feature, we uniformly bucketize its values into N_{stat} bins such that the number of values in each bin is approximately equal. If a feature f with value f_i ($i \in [N_{\text{stat}}]$) is extracted from a column $x \in \mathcal{X}$, we accordingly add a triple (x, r_f, f_i) to the KG, where $r_f \in \mathcal{R}$ is the relation indicating that column x (head entity) has the property of possessing feature f with value f_i (tail entity). For instance, suppose f represents "average number of numerical characters" with value $f_i = [1.0, 3.5]$. The triple (x, r_f, f_i) stands for the fact that the average number of numerical characters in column x lies in the range $[1.0, 3.5]$.

Character-level statistics We also extract statistical features for a set of ASCII-printable characters including digits, letters, and several special characters from each column. Given a character c, we extract 10 features: {any, all, mean, variance, min, max, median, sum, kurtosis, skewness} of the number of occurrences of c in the cells. Again, we bin the values of each feature into N_{char} buckets. If a feature f_c of character c with value $f_{c,i}$ is extracted from column x, we add the corresponding triple $(x, r_{f_c}, f_{c,i})$ to the KG, where r_{f_c} represents the column x has the property of feature f_c with value $f_{c,i}$. For instance,

if f_c represents "average number of character @" ($c =$"@") with value $f_{c,i} = [0.5, 2.2]$, the triple $(x, r_{f_c}, f_{c,i})$ asserts that the average number of occurrences of "@" in the column values for x is within the range of $[0.5, 2.2]$.

Cell keywords The above two kinds of features cover statistical information at different levels of granularity. We further consider word-level features by tokenizing all cell values in a column. After aggregating all unique values, we choose the top $|V_{\text{cell}}|$ frequent values as the keyword vocabulary V_{cell}. The reason not to directly utilize a word embedding such as word2vec [87] for feature extraction is that individual dimensions of the word embedding are not comprehensible. If a column x contains a keyword $w \in V_{\text{cell}}$, we add a triple $(x, r_{\text{has\_keyword}}, w)$ to the KG, meaning that the column entity x connects to a keyword entity w via relation $r_{\text{has\_keyword}} \in \mathcal{R}$.

Header/Label features In some cases, a header can directly reflect the meaning of the column, which can be used to establish a correspondence to a candidate label. Similar to cell keywords, we also tokenize headers and labels to enlargen the keyword set V_{cell}. Supposing a label y or the header of column x contains a keyword w, we denote this fact as $(y, r_{\text{described\_by}}, w)$ or $(x, r_{\text{described\_by}}, w)$. In addition, if a column x is known to be matched to a label y, we directly add a triple (x, r_{match}, y) to the KG.

Neural network architectures and hyperparameters

For the TransE[5] embeddings that are used for initializing the state and action representations, we set both the entity and relation embedding dimensionality to 100. The model is implemented in OpenKE [40], and trained using Adam optimization with a learning rate of 0.0008, batch size of 100, and the number of training epochs set to 100.

For our KG reasoning model, the architectures of the four neural networks are defined below.

- The policy network is defined as

$$\pi_\theta(a_t|s_t) = N(a_t|\mu_\theta(s_t), \mathrm{diag}(\Sigma_\theta(s_t))),$$
$$\mu_\theta(s_t) = W_2\mathrm{ReLU}(W_1(\mathbf{s}_t)), \;\; \Sigma_\theta(s_t) = W_3\mathrm{ReLU}(W_1(\mathbf{s}_t)),$$

  where $W_1 \in \mathbb{R}^{200\times256}$ and $W_2, W_3 \in \mathbb{R}^{256\times200}$.

- The reward function is defined as

$$R_\phi(s_t, a_t) = W_5\mathrm{ReLU}(W_4[\mathbf{s}_t; \mathbf{a}_t]),$$

  where $W_4 \in \mathbb{R}^{400\times256}$ and $W_5 \in \mathbb{R}^{256\times1}$.

- The state-dependent covariance matrix in the worker policy is:

$$\Sigma_\omega(s_t) = W_7\mathrm{ReLU}(W_6\mathbf{s}_t),$$

  where $W_6 \in \mathbb{R}^{200\times256}$ and $W_7 \in \mathbb{R}^{256\times200}$.

- The variational distribution is defined as

$$q_\psi(a_t|s_t, \tilde{a}_t) = N(a_t|\mu_\psi(s_t), \mathrm{diag}(\Sigma_\psi(s_t))),$$
$$\mu_\psi(s_t) = W_9\mathrm{ReLU}(W_8([\mathbf{s}_t); \tilde{\mathbf{a}}_t]),$$
$$\Sigma_\psi(s_t) = W_{10}\mathrm{ReLU}(W_8([\mathbf{s}_t; \tilde{\mathbf{a}}_t])),$$

  where $W_8 \in \mathbb{R}^{400\times256}$ and $W_9, W_{10} \in \mathbb{R}^{256\times200}$.

For ϕ, ω, and ψ, we rely on Adam optimization with a learning rate 10^{-4} and batch size 256. We adopt PPO [115] to train the policy network with parameter θ. The model is trained for 100,000 steps on the `WWT`, `Retailer`, and `Marketing` datasets, and for 200,000 steps on the `WebTable78` dataset.

In the KG construction, we set the bin sizes $N_{\mathrm{stat}} = 20$, $N_{\mathrm{char}} = 20$ and vocabulary

size $|V_{\text{cell}}| = 15,000$.

5.4 Experiments

We extensively evaluate our method in terms of both the annotation performance and the explainability on two open-source datasets and two genuine industry datasets. We aim to answer the following three research questions via experiments.

- **RQ1**: How is the column annotation performance of the proposed EXACTA compared to prior work? (subsection 5.4.2)
- **RQ2**: Does our method provide good explainability for column annotation despite the noisy input paths? (subsection 5.4.3)
- **RQ3**: What other factors may influence the annotation performance? (subsection 5.4.4)

5.4.1 Experimental Setup

Datasets We experiment with four real-world datasets from both open-public domains and industrial platforms. `WWT` is an open-sourced dataset [134] originally used for query search on web tables. It contains over 18,000 columns, about 460,000 rows and 160 unique annotation labels derived from ontology entities. `Retailer` is a real industrial dataset collected from the Adobe Analytics platform[4], which records customer relationship management (CRM) and customer web browsing data. It comprises 16,500 columns and 33 unique manually-annotated labels and each column can harbor thousands of values of diverse data types. `WebTable78` is a subset of WebTable collections [6] for semantic data type detection on column cells. It comprises 5 disjoint datasets including approximately 80,000 columns labeled with 78 classes in total. `Marketing` is another industry dataset collected from the Adobe Marketo Engagement Platform[5], which manages marketing across various channels (e.g., email, text messages, etc). It contains around 24,000 columns with 81 unique labels. Note that `WebTable78` and `Marketing` are more challenging, as they do

[4]https://www.adobe.com/analytics/adobe-analytics.html
[5]https://www.marketo.com/adobe-experience-cloud/

Table 5.1: Statistics of the KG constructed on four datasets.

Dataset	#columns	#labels	#entities	#relations	#triples
WWT	18,670	160	52,755	150	3.503M
Retailer	16,500	33	48,015	150	2.213M
WebTable78	80,000	78	42,515	148	13.409M
Marketing	23,835	81	44,050	148	6.157M

not include column header names. For each dataset, we perform a random split into 60% of columns for training, 20% for validation, and 20% for testing.

KG construction and path generation We construct a KG for each dataset by extracting explainable features from columns and labels following the previous work [53]. These include cell-level statistics (e.g., mean number of numerical characters in column), character-level statistics (e.g., mean number of "@" in each cell), keywords in cells, headers, and label descriptions. The statistics of the 4 KGs are given in Table 5.1. In order to obtain large quantities of paths without quality guarantees, we randomly sample 3 to 5 paths for each training column, and then ask crowdsourced workers to manually assess whether these paths are suitable as explanations for the column–label pairs. A score of "0" is assigned to a path if it does not make any sense for the pair and "1" otherwise. We discard all paths with "0" scores and keep around 1.756 paths that are deemed "somewhat explainable" for each training column–label pair over all 4 datasets. Note that these paths are not the most explainable and serve as the noisy input trajectories for our model.

Baselines We mainly consider the following three categories of baselines. Rule-based methods are earlier techniques for column annotation that are explainable but less effective. Deep neural network based approaches show better performance but the annotation mechanism is opaque to humans due to their black-box nature. KG-based methods rely on the same constructed graph to make predictions with path-based explanations.

- Rule-based methods: `LexicalMatch` [104] is an early heuristic approach that detects column types by collecting frequent lexical keywords across all columns. `DSL` [99] is

a representative column annotation model that exploits column features and makes its prediction via random forest.

- Deep neural networks: `ColNet` [10] is a deep neural model that relies on an external knowledge base to identify column types together with entity lookup voting. `Sherlock` [53] is a deep neural model that also utilizes the features extracted from columns. `SATO` [142] is the state-of-the-art neural model based on Sherlock with additional topic modeling components. For a fair comparison, we train these models with the cell-level statistical features, character-level statistical features, and pretrained word2vec [98] features, which are consistent with our KG-based model.
- KG-based approaches: `TransE` [5] is a KG embedding technique via vector translation operations. We also use it to initialize the state and action representation in our model. `RotatE` [121] is an advanced KG embedding modeled in a complex vector space. `ME-IRL` [154] is an IRL method that learns rewards from expert trajectories via the max-entropy principle. `KGRL` [77] is the state-of-the-art RL-based KG reasoning model to predict missing links.

The implementation of our method is described in the Appendix. For each dataset, we tune each method on the validation set, and repeatedly run it 5 times on the test set and report average performance on four metrics (F1 score, hit rate@1, 3 and 5).

5.4.2 Experiment on Column Annotation

We first evaluate the column annotation performance of our method compared to the selected baselines on four datasets (**RQ1**). The benchmark results of all methods are reported in Table 5.2.

Overall, we observe that our method EXACTA shows superior performance over other baselines on all benchmarks in terms of F1 and Hits@k. For example, on the WWT and Retailer datasets, our model obtains around 5.06% and 3.64% improvement in F1 score, and 3.07% and 2.70% in Hits@1 over the best baseline RotatE. Our model can also handle

Table 5.2: Benchmark results of our method compared to other baseline approaches on four datasets for the column annotation task. The best results are highlighted in bold and the second best results are underlined.

	WWT				**Retailer**			
Methods	F1	Hits@1	Hits@3	Hits@5	F1	Hits@1	Hits@3	Hits@5
LexicalMatch	0.2479	0.2545	0.3438	0.3638	0.3741	0.4173	0.5252	0.5576
DSL [99]	0.3649	0.3957	0.5728	0.6464	0.7856	0.8009	0.9988	0.9994
ColNet [10]	0.4148	0.4332	0.6429	0.7125	0.7592	0.7706	0.9921	0.9994
Sherlock [53]	0.4437	0.4735	0.6524	0.7293	0.7724	0.7988	1.0000	1.0000
SATO [142]	0.4387	0.4664	0.6681	0.7296	0.7974	0.8188	1.0000	1.0000
TransE [5]	0.4347	0.4563	0.6924	0.7768	0.7771	0.8030	1.0000	1.0000
RotatE [121]	0.4835	0.5150	0.7282	0.7911	0.8111	0.8303	1.0000	1.0000
ME-IRL [154]	0.3250	0.3372	0.5305	0.5968	0.7250	0.7458	0.9154	0.9525
KGRL [77]	0.4049	0.4304	0.6623	0.7453	0.7927	0.8102	0.9858	0.9994
EXACTA (ours)	**0.5080**	**0.5308**	**0.7353**	**0.7966**	**0.8406**	**0.8527**	**1.0000**	**1.0000**
	WebTable78				**Marketing**			
Methods	F1	Hits@1	Hits@3	Hits@5	F1	Hits@1	Hits@3	Hits@5
LexicalMatch	0.4268	0.4344	0.5155	0.5454	0.5362	0.5339	0.6122	0.6294
DSL [99]	0.5132	0.5262	0.6493	0.7093	0.6226	0.6440	0.6705	0.6831
ColNet [10]	0.5133	0.5173	0.6589	0.7217	0.7005	0.7202	0.9240	0.9328
Sherlock [53]	0.5714	0.5852	0.7183	0.7735	0.7400	0.7889	0.9901	0.9952
SATO [142]	0.6244	0.6279	0.7476	0.7873	0.7692	0.7975	0.9903	0.9960
TransE [5]	0.5352	0.5233	0.6867	0.7517	0.7567	0.7712	0.9874	0.9937
RotatE [121]	0.5858	0.5753	0.7233	0.7853	0.7661	0.7953	0.9863	0.9937
ME-IRL [154]	0.4483	0.4928	0.6193	0.6598	0.6057	0.6209	0.8815	0.9052
KGRL [77]	0.5025	0.5121	0.6805	0.7422	0.7397	0.7428	0.9808	0.9916
EXACTA (ours)	**0.6358**	**0.6347**	**0.7566**	**0.7949**	**0.7973**	**0.8155**	**0.9921**	**0.9976**

the very challenging case of column headers being entirely missing, as we see that the results are still promising on `WebTable78` and `Marketing`. This indicates that our KG reasoning method can also guarantee the performance even if header keywords are absent in the KG.

It is of particular interest to see that our model outperforms other KG embedding models and RL-based KG reasoning methods. As we adopt TransE embeddings for initializing state and action representation, the considerable gains achieved by our model indicate that

explicit KG reasoning yields additional information such as useful features to deliver final predictions. At the same time, our model also outperforms KGRL and ME-IRL by a large margin across all benchmarks. The former only learns the policy from handcrafted rewards, while the latter learns rewards without accounting for potential noise in the input paths. Hence, the results imply that the performance gap mainly stems from the superior quality of the reward functions, which cause the model to learn a good policy.

We also notice that the KG-based methods (e.g., RotatE) generally work better than the deep neural networks (e.g., SATO) on the datasets with column headers (`WWT`, `Retailer`). After checking the header names with the label names and description, we find headers to be a strong indicator for annotation, e.g., "email" is the common keyword in some columns and the label in the Retailer dataset. In addition, all models attain much better performance on two industrial datasets (`Retailer`, `Marketing`) than on the others (`WWT`, `WebTable78`). After investigating column content in the datasets, we find this performance gap is caused by data intricacies and ambiguity. For instance, many columns in `WWT` consist of names of people but are annotated with diverse labels such as "actors", "footballer", "presidents", etc. This makes annotating this dataset particularly challenging unless one incorporates additional features to introduce world knowledge regarding the profession of a person. On the contrary, most columns in the `Retailer` dataset are fairly easy to recognize and hence one quickly approaches 100% top 5 accuracy. For example, an "ID" column consists entirely of integer values, which can be discerned using the statistical features.

5.4.3 Experiment on Explainability

In this experiment, we comprehensively evaluate the explainability of KG reasoning paths emitted by our model in terms of perceived explainability, robustness, and faithfulness (**RQ2**).

Methods	**Retailer**	**Marketing**
Random	2.29±1.60	2.08±1.23
KGRL	2.41±1.56	2.03±1.22
ME-IRL	3.20±1.21	2.28±0.93
Ours	3.49±1.15	2.55±0.86

Table 5.3: Average perceived explainability (with std. dev.) of 4 methods on 2 datasets.

Expert Evaluation of Perceived Explainability

We first study the perceived explainability of our model compared with other KG reasoning baselines (ME-IRL, KGRL) and random path sampling.

We experiment on two industrial datasets (`Marketing` and `Retailer`) and collaborate with 5 human experts who are specialized in the digital marketing platform and familiar with the domain.[6] For each dataset, we randomly select 50 column–label pairs that are correctly predicted by our model. For every such pair, we run 4 algorithms that each generate a path of fixed length 3, resulting in 4 paths for the pair. Upon completion of the procedure, we collect 400 explainable paths in total from these algorithms on two datasets. Human experts are requested to rate each path along a 5-point Likert scale, where "5" means the path is completely explainable for the prediction, while "1" means the path does make any sense. We consider the path relevance to be the perceived explainability score given by the participants.

We report the average score of 4 algorithms given by 5 experts in Table 5.3. Overall, our method obtains a substantially better perceived explainability compared to all the baselines. Specifically, our model outperforms the other IRL-based method ME-IRL, which establishes that our method is able to find more explainable intermediate nodes despite being trained on unreliable input paths, while ME-IRL is affected by noise in the input trajectories. Meanwhile, both of the IRL-based methods achieve better explainability than the RL method, which merely has comparable performance with random path sampling.

[6]Note that we cannot experiment on the other two datasets due to a lack of experts who can evaluate explanations with specialization in those domains.

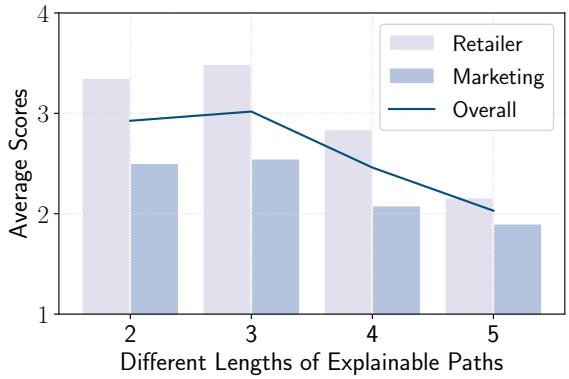


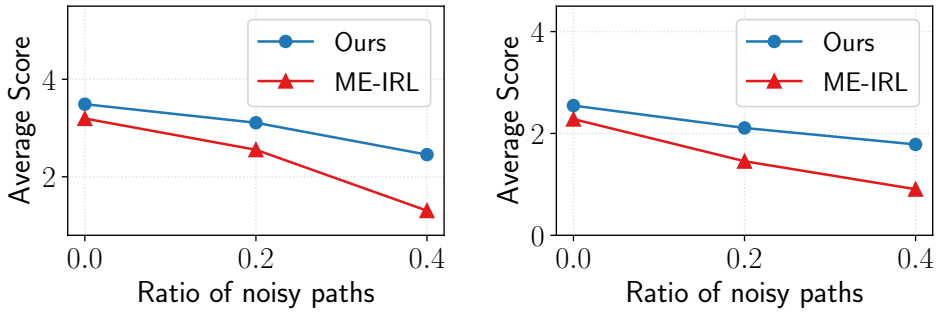
Figure 5.3: Perceived explainability of our method on various path lengths.

This implies that such RL-based methods completely fail to distinguish the explainability of different paths between the same column–label pair.

Path length on perceived explainability

We further evaluate the influence of the path length on perceived explainability and attempt to answer what the most suitable length is for explanatory purposes. For each of two datasets, we adopt the same 50 column–label pairs as in the last experiment but generate further paths of length 2, 4, 5 by our model. We again ask 5 human experts to score the perceived explainability of the paths.

As shown in Figure 5.3, we find that the highest score across both datasets is achieved when the path length is 3, which is slightly higher than the length of 2, but significantly preferred over longer paths. We further check the paths generated in this experiment and find that shorter paths contain mostly more comprehensible features such as keywords, whereas longer paths tend to consist of less understandable statistical features. Intuitively, if the path length is overly restricted, people may not recognize the reasoning process as legible and logical. However, when the path becomes too long, it may fail to possess meaningful explainability, due to the presence of various redundant reasoning steps.



(a) Retailer *(b) Marketing*

Figure 5.4: Perceived explainability of our method and ME-IRL when trained with varying percentages of noisy paths.

Robustness to Very Noisy Input Paths

Recall that when generating training paths annotated by crowdsourced workers in subsection 5.4.1, we only keep the "1"-scoring random paths but discard all the "0"-scoring random paths, as they are deemed not at all explainable. In this experiment, we evaluate how these very noisy paths as training data influence the perceived explainability of our model compared to the regular IRL method (ME-IRL). Specifically, we add the "0"-scoring paths to the training set at different ratios of 0%, 20%, and 40% among all paths. Both our model and ME-IRL are retrained with these new training paths, while maintaining all other settings as in the default setup. The perceived explainability is evaluated in the same way as in the previous experiments.

The average scores of the two methods are plotted in Figure 5.4. We observe that our method (blue curve) consistently achieves better perceived explainability than ME-IRL across both datasets. Furthermore, the gap between the methods grows as further noisy paths are included in the training set, which implies our method is more capable of coping with noise during training and is more robust to varying degrees of quality of the input training data.

Faithfulness of Feature Explainability

In addition to the human evaluation, we further quantitatively measure the faithfulness of explainability, i.e., the extent to which explanations provided by a model genuinely inform the predictions. Motivated by recent work in the field of NLP [20, 141], we adopt two metrics known as *comprehensiveness* and *sufficiency* to evaluate the faithfulness of the generated explanation. Specifically, let x be a column associated with a set of extracted explainable features and $f(x)$ be an annotation model that takes the column x as input and emits top k predicted labels along with k explainable features v_k (or features nodes connecting columns). With the annotation metric $m(\cdot)$, e.g., Hits@k, we define *comprehensiveness* h_{com} and *sufficiency* h_{suf} as:

$$h_{\text{com}} = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{x \in \mathcal{X}_{\text{test}}} \frac{m(f(x)) - m(f(x \backslash v_k))}{m(f(x))} \tag{5.13}$$

$$h_{\text{suf}} = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{x \in \mathcal{X}_{\text{test}}} \frac{m(f(x)) - m(f(v_k))}{m(f(x))} \tag{5.14}$$

Here, $x \backslash v_k$ denotes the column excluding the predicted k features. Note that the metrics h_{com} and h_{suf} focus on the difference ratio of the model performance, assessing to what extent the features contribute to the prediction. Therefore, these new metrics eliminate the effect of absolute differences in performance between models. We set $k = 5$ in this experiment. A high score of h_{com} indicates the extracted explanations v_k indeed influence the annotation, while h_{suf} captures the degree to which the extracted explainable features are adequate for the model to make predictions.

We consider two sets of baselines in this experiment, including model-agnostic approaches and KG-based approaches. Methods in the first category (LIME [107], Anchor [108]) can be applied to any deep classifiers (e.g., SATO) to generate explainable features, and hence the two faithfulness metrics can be directly calculated on the test set. KG-based approaches (KGRL, ME-IRL and ours) first generate reasoning paths and then extract the

Table 5.4: Evaluation of faithfulness of the explainability

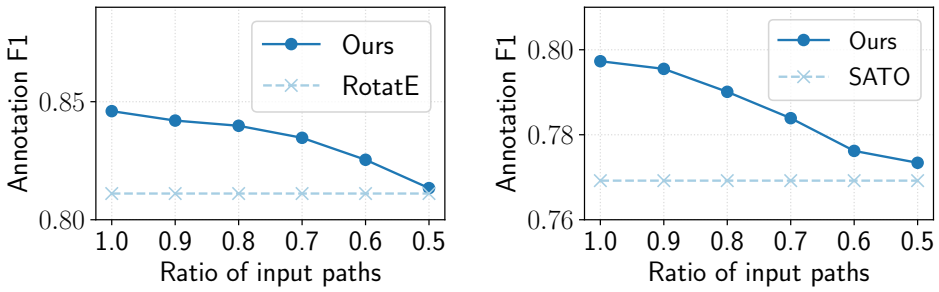
	Retailer		**Marketing**	
Methods	$h_{\text{com}} \uparrow$	$h_{\text{suf}} \downarrow$	$h_{\text{com}} \uparrow$	$h_{\text{suf}} \downarrow$
SATO + LIME	0.024	0.519	0.019	0.601
SATO + Anchor	0.029	0.480	0.022	0.584
KGRL	0.035	0.184	0.029	0.227
ME-IRL	0.041	0.143	0.035	0.209
Ours	**0.070**	**0.116**	**0.058**	**0.165**

feature nodes connecting the start column in KG. The *comprehensiveness* is computed by removing all these features nodes and their edges from the KG, while *sufficiency* is obtained by removing the outgoing edges of the column that do not belong to the extracted features.

The results are reported in Table 5.4. As we discussed before, one possible reason that SATO with LIME is not competitive is that it does not specify which local feature space is applicable. It always yields explanations that may vary considerably for different neighborhoods in the feature space. Compared to the model-agnostic approaches, the graph-based explainable methods are able to provide more faithful feature entities within the reasoning paths. We observe that features extracted by our method indeed make a notable contribution, especially for the *sufficiency* evaluation. With purely extracted top-k features, it is reasonable that SATO performance will drop substantially, whereas our graph-based approach is able to maintain the path inference procedure, which quantitatively proves the faithfulness of the model's explainability.

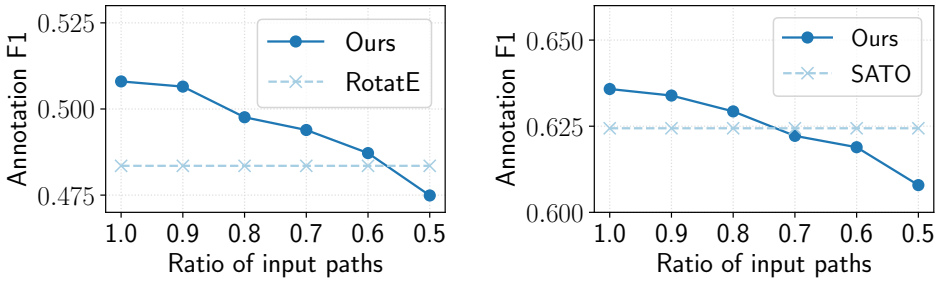
5.4.4 Ablation Study

We further seek a better understanding of what other factors may influence the performance of our model (**RQ3**).



(a) Retailer *(b) Marketing*

Figure 5.5: Column annotation performance (F1) of our method trained with various portions of input paths compared to the best baseline on two industrial datasets.

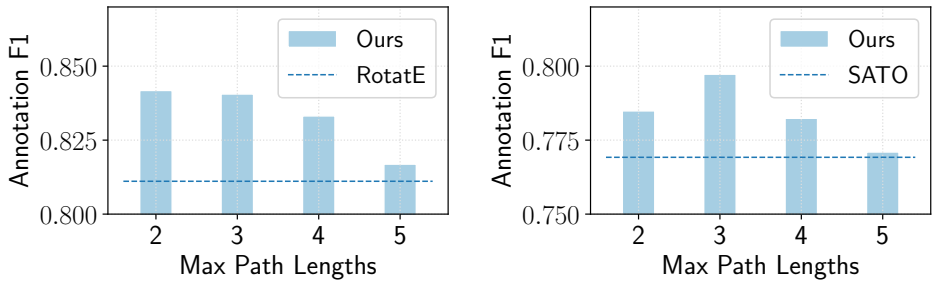


(a) WWT *(b) WebTable78*

Figure 5.6: Column annotation performance (F1) of our method trained with various portions of input paths compared to the best baseline on two open-source datasets.

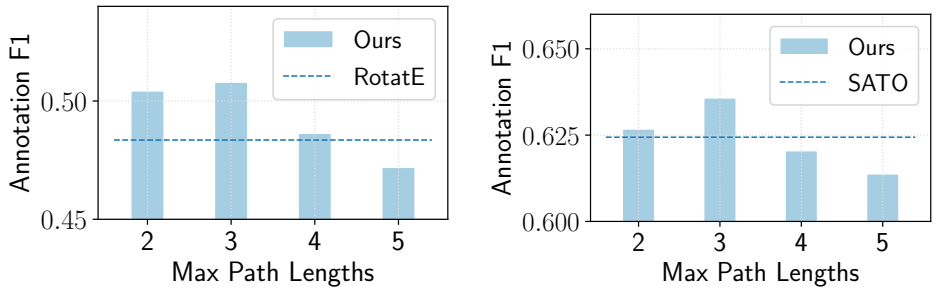
Influence of quantity of input paths

In this experiment, we evaluate if our model requires a large amount of training paths as input and how the performance will change with less training data. Specifically, for each of the four datasets, we only retain paths for 90%, 80%, 70%, 60%, 50% of the input columns while leaving the rest of the columns with no corresponding paths. We retrain our model with these smaller-sized training sets and report F1 scores of our model and the best baseline in Figure 5.5 and Figure 5.6. We find that the performance drop is within an acceptable range, i.e., even if 50% of the columns are left without training paths, our model still achieves comparable results to the best baseline. The benefit is that our model does not require enormous amounts of training paths, which saves much effort in manual explainability labeling of paths.



(a) Retailer *(b) Marketing*

Figure 5.7: Column annotation performance (F1) of our method when varying the maximum step size in reasoning compared to the best baseline on two industry datasets.



(a) WWT *(b) WebTable78*

Figure 5.8: Column annotation performance (F1) of our method with various output path lengths compared to the best baseline on two open-source datasets.

Influence of maximum steps

We further evaluate how different maximum step sizes (i.e., horizons) T influence the annotation performance of our model. In general, larger values of T result in longer paths and hence make it harder to reach the correct destination in KG reasoning. The resulting F1 scores of our method on four datasets are reported in Figure 5.7 and Figure 5.8. We observe that the best performance is consistently achieved when $T = 3$, which is comparable to the case of $T = 2$, but much better than when operating with longer horizons. The reason behind these results is that when reasoning for longer numbers of steps, one is more likely to arrive at spurious nodes instead of a potentially correct target.

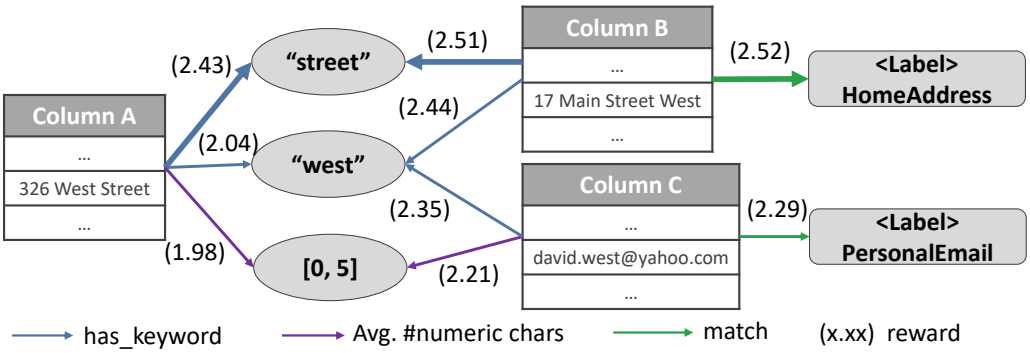


Figure 5.9: Visualization of learned rewards on a subgraph of the `Retailer` dataset. The explainable path with the highest cumulative rewards is highlighted in bold.

5.4.5 Qualitative Analysis on Learned Reward

In order to better investigate why our model is able to reach explainable feature nodes during path reasoning, we visualize the learned rewards to see if important feature nodes in the KG are assigned higher rewards by our model. As illustrated in Figure 5.9, we showcase one example from the test set, where rewards marked in brackets are associated with each edge (action in MDP). We can see that there are in total four paths connecting the starting "Column A" and two potential labels, and the predicted path is highlighted using bold edges, along which the agent is able to collect the highest rewards. Intuitively, the predicted path leads to a correct label and its explainability is better than that of the other 3 paths. This intuition is consistent with the observed reward values, suggesting that our model is able to learn good rewards for KG reasoning.

5.4.6 Online Simulation

Finally, we evaluate our model in the production environment for column annotation in the digital marketing domain. Specifically, we compare with a deployed model that only predicts the annotation for each column without providing explanations, to see if our model can bring performance gains in annotation prediction and genuinely improve the work efficiency of the human experts in evaluating the labels. The experiment is conducted on a newly dumped dataset about retail management including around 1,000 unlabeled columns,

each of which belongs to one of the labels in the `Retailer` dataset. Therefore, each model is trained on the historical `Retailer` dataset and then makes predictions on around 500 columns that are randomly partitioned from the new dataset. To simulate the real column annotation scenario, the derived results by both models are presented to the human experts who verify the correctness of the prediction. Note that our model also generates additional explanations for the human, and for simplicity in presentation, the explanation is only made of important features extracted by our model rather than the whole path. The results show that our model achieves an accuracy of 85.26% on average compared to 82.54% by the deployed model. More importantly, the human experts expend around 12% less time evaluating the results by our model than those by the existing one, which confirms that the additional explanations provisioned by our model are beneficial for the human verification process.

5.5 Related Work

Column Annotation. The task of column annotation involves annotating an entire tabular column with a semantic label. Cell-level or row-level annotation tasks (e.g., knowledge base entity alignment [103, 24]) are not considered in this chapter because they are designed for a different granularity of data. Early works on column annotation usually embraced rule-based methods. Many open-source and commercial systems [66, 26, 86] adopt regular expression and dictionary methods to match columns with predefined patterns or keywords. Ramnandan *et al.* [104] propose a frequency-based method with heuristics to detect data types of table columns. However, these rule-based methods lack extensibility to unseen data and require considerable effort to manually manage rules.

Another line of recent research relies on machine learning techniques to address the problem [58]. Limaye *et al.* [76] annotate tables with types using probabilistic graphical models (PGM) by maximizing a potential function over a set of predefined feature vari-

ables. Pham *et al*. [99] extract statistical features from tables and use random forests to predict the annotations. However, the performance of these methods heavily depends on handcrafted features and they have less generalizability compared to deep learning methods, which have recently been adopted to learn rich features from tabular data. Chen *et al*. [10] embed context information in the model by first looking up cells to retrieve entities in a knowledge base, followed by a prediction step to estimate the classes via a convolutional neural network and majority voting. Chen *et al*. [11] propose another hybrid deep neural network by exploiting table locality features and inter-column semantic features. Zhang *et al*. [142] integrate deep learning and topic modeling to annotate columns with semantic types. Hulsebos *et al*. [53] directly extract richer features, including semantic features such as word embeddings and paragraph embeddings and then invoke a deep neural network to classify the label. These methods achieve superior annotation performance compared to the previous generation of models due to the representational power of deep neural networks. However, the issue of explainability remains under-explored, despite being crucial in a real industry business scenarios.

KG Reasoning for Explainable Prediction. KG reasoning is known for its transparent decision making process via multi-hop reasoning and has been widely explored in missing link prediction [139, 77], medical diagnosis [109], and recommendation [1, 136]. In particular, Ai *et al*. [1] first proposed to leverage the KG path as an explanation of a recommendation, which was shown to be effective in boosting the user shopping experience. Different from the task of recommendation, the problem of column annotation is more challenging due to the complexity and intricacies of tabular data. Moreover, none of these works evaluate the quality of the explainable paths, while we collaborate with domain experts to conduct a comprehensive evaluation of the path explainability.

5.6 Conclusion

We proposed a novel KG reasoning model under the IRL framework, called EXACTA, to approach the explainable column annotation task, which plays an important role in industrial digital marketing data pipelines. Our method automatically learns a noise-tolerant reward function from noisy, potentially less explainable paths to guide the policy learning process such that the agent is able to reason over the KG from a source column node to a potential target label. The derived reasoning paths can naturally be regarded as explanations for the predicted labels. We empirically show that the proposed EXACTA can produce higher-quality column annotations compared with state-of-the-art deep learning-based methods. We also comprehensively evaluate the explainability of our model, which obtains promising results in terms of the perceived explainability, robustness, and faithfulness.

CHAPTER 6
SUMMARY AND FUTURE WORK

In this thesis, we propose four different methods to approach the neural graph reasoning problem for explainable decision-making. We first propose a basic graph reasoning framework based on reinforcement learning called PGPR, which is able to generate ad-hoc path-based explanation via a policy-guided agent to conduct multi-hop reasoning over knowledge graphs. The resulting reasoning paths can be directly regarded as the explanation to the prediction since they explicitly expose the multi-step decision-making procedure. Extensive experiments are conducted to evaluate the performance of the proposed method compared with several state-of-the-art baselines. Note that the PGPR method is a flexible graph reasoning framework and can be easily extended to other graph-related tasks in various domains such as product search and advertising.

Next, we further investigate how to enable the neural reasoning model itself to be transparent and interpretable. Unlike the previous work that represents the graph walker as a single black-box neural network, we propose a second method called CAFE that is characterized by maintaining a set of explainable neural relation modules, each of which represent a concrete relation in knowledge graphs. The graph walker is constructed on the fly with a composition of selected neural relation modules based on user profile, so that the architecture of the graph walker directly reflects the execution process of the decision-making procedure. We evaluate the CAFE model on several real-world datasets and show that it achieves significant performance gain over PGPR.

Moreover, we also study how to leverage logic rules from knowledge graphs to produce faithful explanation during the graph reasoning process. Accordingly, we propose a neural logic model called LOGER, which learns personalized and important logic rules to guide the path-reasoning process for explanation generation under the Expectation-Maximization

framework. We experiment on three large-scale datasets in e-commerce recommendation and the results show the effectiveness of the proposed method in making high-quality decisions accompanied with faithful explainable paths.

Lastly, we consider a special case where explainable paths are available during training and whether the demonstrations are useful in generating high-quality explainable paths. We propose a new graph reasoning approach based on inverse reinforcement learning called EXACTA, which explicitly learns rewards from demonstrated paths to capture both accuracy-driven quality and explainability of each state over knowledge graphs. The learned rewards will be used to jointly learn a policy network for multi-hop graph reasoning. We experiment on four open-sourced and real industrial benchmarks in the digital marketing scenario and show that our method can provide competitive prediction performance in the column annotation task. We also undertake a comprehensive analysis on path-based explainability indicating that our model can produce faithfully explainable paths to facilitate human examination.

Future Work The following related problems/directions would be interesting to researchers in studying explainable graph reasoning.

- **Reasoning over Dynamic Graphs.** Our work primarily focuses on conducting multi-hop reasoning over static knowledge graphs. However, graphs can be also dynamic in two aspects. First, new source and target nodes can be added in the inference stage, which results the inductive graph reasoning problem. The key challenge here is how to estimate the representation of these unseen nodes when conducting graph reasoning. One possible solution is to leverage heterogeneous graph neural networks to model latent embeddings of nodes and edges by incorporating high-order neighboring information. Besides, the dynamics can also occur during the evolution of graphs, where additional temporal information needs to be considered during representation and reasoning. In this case, an additional model may be required to model

temporal dynamics of graphs, which can be later utilized to drive the multi-hop reasoning procedure.

- **Fairness in Graph Reasoning.** Fairness can be another interesting yet important factor to be considered along with explainability in graph reasoning. Many fairness related problems have been studied in human-involved tasks such as recommendation, law judgement, marketing and some potential issues have been discovered in existing machine learning models including gender bias, group unfairness, etc. However, limited effort has been put on the fairness issue in graph reasoning and let alone the derived explanations. Some existing well-formulated fairness frameworks may not be directly applicable to the graph setting since they do not explicitly take into consideration the unique properties in graph structures. Meanwhile, fair explanation can also be critical if the generated explanation is presented to end users who may react distinctly to the explanations when they belong to different groups (e.g., active users, cold-start users in recommender system).

- **Causal Inference for Graph Reasoning.** Our proposed graph reasoning methods can produce explanation by finding the most relevant next-hop nodes that are more likely to lead to a potentially correct target node. Such methods usually consider associative relation among visited graph nodes, but do not model causal relationship, i.e., what if the graph walker moves to another node with lower predicted probability. It would be interesting to study the graph reasoning problem under the casual inference framework to generate counterfactual path-based explanation.

- **Graph Pretraining.** Training representations and graph walkers over large-scale graph may cost lots of time and resources, and it is obviously unreasonable to always retrain the model from scratch when the graph is updated. Recent work that borrows the idea from pretrained language model in the natural language processing field attempts to pretrain the graph representations for later use in downstream tasks. This

may also benefit the graph reasoning approaches that rely on graph representations to make decisions as well as explanations.

- **More Applications.** More human-involved tasks that requires both high accuracy and trustable explainability can be explored under the graph reasoning framework. This includes but not limited to product search, advertising, conversational AI, medical diagnosis, financial related scenarios and so on.

REFERENCES

[1] Q. Ai, V. Azizi, X. Chen, and Y. Zhang, "Learning heterogeneous knowledge base embeddings for explainable recommendation," *Algorithms*, 2018.

[2] Q. Ai, Y. Zhang, K. Bi, and W. B. Croft, "Explainable product search with a dynamic relation embedding model," *TOIS*, 2019.

[3] K. Balog, F. Radlinski, and S. Arakelyan, "Transparent, scrutable and explainable user models for personalized recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 265–274.

[4] J. Besag, "Statistical analysis of non-lattice data," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 24, no. 3, pp. 179–195, 1975.

[5] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 2787–2795.

[6] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *VLDB*, 2008.

[7] Y. Cao, X. Wang, X. He, Z. Hu, and T. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, Eds., ACM, 2019, pp. 151–161.

[8] R. Catherine, K. Mazaitis, M. Eskenazi, and W. Cohen, "Explainable entity-based recommendations with knowledge graphs," in *RecSys*, 2017.

[9] H. Chen, X. Chen, S. Shi, and Y. Zhang, "Generate natural language explanations for recommendation," 2019.

[10] J. Chen, E. Jiménez-Ruiz, I. Horrocks, and C. Sutton, "Colnet: Embedding the semantics of web tables for column type prediction," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, AAAI Press, 2019, pp. 29–36.

[11] ——, "Learning semantic annotations for tabular data," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed., ijcai.org, 2019, pp. 2088–2094.

[12] X. Chen, H. Chen, H. Xu, Y. Zhang, Y. Cao, Z. Qin, and H. Zha, "Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 765–774.

[13] X. Chen, Y. Zhang, and Z. Qin, "Dynamic explainable recommendation based on neural attentive models," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, AAAI Press, 2019, pp. 53–60.

[14] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[15] J. S. Cockerham, *An Integrative Study on the Impact of Undergraduate Coursework Designed for College Student Development*. Northcentral University, 2011.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[17] F. Costa, S. Ouyang, P. Dolog, and A. Lawlor, "Automatic generation of natural language explanations," in *Proceedings of the 23rd international conference on intelligent user interfaces companion*, 2018, pp. 1–2.

[18] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[19] A. Datta, S. Sen, and Y. Zick, "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems," in *2016 IEEE symposium on security and privacy (SP)*, IEEE, 2016, pp. 598–617.

[20] J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace, "ERASER: A benchmark to evaluate rationalized NLP models," in *Proceed-*

ings of the 58th Annual Meeting of the Association for Computational Linguistics, Online: Association for Computational Linguistics, 2020, pp. 4443–4458.

[21] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 731–742.

[22] X. Dong, J. Ni, W. Cheng, Z. Chen, B. Zong, D. Song, Y. Liu, H. Chen, and G. de Melo, "Assymetrical hierarchical networks with attentive interactions for interpretable review-based recommendation," in *AAAI*, AAAI Press, 2020.

[23] X. L. Dong, "Challenges and innovations in building a product knowledge graph," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds., ACM, 2018, p. 2869.

[24] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, and V. Christophides, "Matching web tables with knowledge base entities: From entity lookups to entity embeddings," in *ISWC*, Springer, 2017, pp. 260–277.

[25] G. Farnadi, J. Tang, M. De Cock, and M.-F. Moens, "User profiling through deep multimodal fusion," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 171–179.

[26] O. K. Foundation, *Messytables: Tools for parsing messy tabular data.* https://github.com/okfn/messytables, 2019.

[27] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[28] Z. Fu, Y. Xian, R. Gao, J. Zhao, Q. Huang, Y. Ge, S. Xu, S. Geng, C. Shah, Y. Zhang, and G. de Melo, "Fairness-aware explainable recommendation over knowledge graphs," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, Eds., ACM, 2020, pp. 69–78.

[29] Z. Fu, Y. Xian, S. Geng, Y. Ge, Y. Wang, X. Dong, G. Wang, and G. de Melo, "Absent: Cross-lingual sentence representation mapping with bidirectional gans," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 7756–7763.

[30] Z. Fu, Y. Xian, Y. Zhang, and Y. Zhang, "Tutorial on conversational recommendation systems," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 751–753.

[31] ——, "Wsdm 2021 tutorial on conversational recommendation systems," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 1134–1136.

[32] Z. Fu, Y. Xian, Y. Zhu, Y. Zhang, and G. de Melo, "Cookie: A dataset for conversational recommendation over knowledge graphs in e-commerce," *arXiv preprint arXiv:2008.09237*, 2020.

[33] L. Gao, H. Yang, J. Wu, C. Zhou, W. Lu, and Y. Hu, "Recommendation with multi-source heterogeneous information," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, J. Lang, Ed., ijcai.org, 2018, pp. 3378–3384.

[34] Y. Ge, S. Zhao, H. Zhou, C. Pei, F. Sun, W. Ou, and Y. Zhang, "Understanding echo chambers in e-commerce recommender systems," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, Eds., ACM, 2020, pp. 2261–2270.

[35] *General data protection regulation - wikipedia*, https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, (Accessed on 08/17/2020).

[36] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 855–864.

[37] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, no. 2, 2017.

[38] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, and X. Wang, "Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems," *Knowl. Based Syst.*, vol. 138, pp. 202–207, 2017.

[39] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.

[40] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li, "OpenKE: An open toolkit for knowledge embedding," in *Proceedings of the 2018 Conference on Empirical*

Methods in Natural Language Processing: System Demonstrations, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 139–144.

[41] G. He, J. Li, W. X. Zhao, P. Liu, and J. Wen, "Mining implicit entity preference from user-item interaction data for knowledge graph completion via adversarial learning," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Y. Huang, I. King, T. Liu, and M. van Steen, Eds., ACM / IW3C2, 2020, pp. 740–751.

[42] R. He, W. Kang, and J. J. McAuley, "Translation-based recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, P. Cremonesi, F. Ricci, S. Berkovsky, and A. Tuzhilin, Eds., ACM, 2017, pp. 161–169.

[43] R. He and J. J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao, Eds., ACM, 2016, pp. 507–517.

[44] ——, "VBPR: visual bayesian personalized ranking from implicit feedback," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, D. Schuurmans and M. P. Wellman, Eds., AAAI Press, 2016, pp. 144–150.

[45] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, Eds., ACM, 2017, pp. 173–182.

[46] M. ter Hoeve, A. Schuth, D. Odijk, and M. de Rijke, "Faithfully explaining rankings in a news recommender system," *arXiv preprint arXiv:1805.05447*, 2018.

[47] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. Labra Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. Ngonga Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ArXiv*, vol. 2003.02320, 2020.

[48] M. Hou, L. Wu, E. Chen, Z. Li, V. W. Zheng, and Q. Liu, "Explainable fashion recommendation: A semantic attribute region guided approach," *arXiv preprint arXiv:1905.12862*, 2019.

[49] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Re-*

trieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018, K. Collins-Thompson, Q. Mei, B. D. Davison, Y. Liu, and E. Yilmaz, Eds., ACM, 2018, pp. 505–514.

[50] Q. Huang, Y. Xian, P. Wu, J. Yi, H. Qu, D. Metaxas, *et al.*, "Enhanced mri reconstruction network using neural architecture search," in *International Workshop on Machine Learning in Medical Imaging*, Springer, 2020, pp. 634–643.

[51] Q. Huang, Y. Xian, D. Yang, H. Qu, J. Yi, P. Wu, and D. N. Metaxas, "Dynamic mri reconstruction with end-to-end motion-guided network," *Medical Image Analysis*, vol. 68, p. 101 901, 2021.

[52] X. Huang, Q. Fang, S. Qian, J. Sang, Y. Li, and C. Xu, "Explainable interaction-driven user modeling over knowledge graph for sequential recommendation," *ACM MM*, 2019.

[53] M. Hulsebos, K. Z. Hu, M. A. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. A. Hidalgo, "Sherlock: A deep learning approach to semantic data type detection," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds., ACM, 2019, pp. 1500–1508.

[54] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *CSUR*, 2017.

[55] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[56] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition and applications," *arXiv preprint arXiv:2002.00388*, 2020.

[57] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[58] A. Kimmig, A. Memory, L. Getoor, *et al.*, "A collective, probabilistic approach to schema mapping," in *ICDE*, 2017.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[60] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[61] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: Applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[62] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, 2009.

[63] A. Krohn-Grimberghe, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme, "Multi-relational matrix factorization using bayesian personalized ranking for social network data," in *Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM 2012, Seattle, WA, USA, February 8-12, 2012*, E. Adar, J. Teevan, E. Agichtein, and Y. Maarek, Eds., ACM, 2012, pp. 173–182.

[64] B. Kveton, S. Mahdian, S Muthukrishnan, Z. Wen, and Y. Xian, "Waterfall bandits: Learning to sell ads online," *arXiv preprint arXiv:1904.09404*, 2019.

[65] B. Kveton, S Muthukrishnan, H. T. Vu, and Y. Xian, "Finding subcube heavy hitters in analytics data streams," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1705–1714.

[66] I. D. Lab, *Datalib: Javascript data utilities*, http://vega.github.io/datalib, 2019.

[67] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Faithful and customizable explanations of black box models," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019, pp. 131–138.

[68] N. Lao, T. Mitchell, and W. W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011, pp. 529–539.

[69] F. Lecue, "On the role of knowledge graphs in explainable ai," *Semantic Web*, vol. 11, no. 1, pp. 41–51, 2020.

[70] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, 2000, pp. 556–562.

[71] J. Li, W. X. Zhao, J.-R. Wen, and Y. Song, "Generating long and informative reviews with aspect-aware coarse-to-fine decoding," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, 2019, pp. 1969–1979.

[72] L. Li, Y. Zhang, and L. Chen, "Generate neural template explanations for recommendation," in *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, M. d'Aquin, S. Dietze, C. Hauff, E. Curry, and P. Cudré-Mauroux, Eds., ACM, 2020, pp. 755–764.

[73] S. Li and H. Zhao, "A survey on representation learning for user modeling," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed., ijcai.org, 2020, pp. 4997–5003.

[74] Z. Liao, Y. Xian, J. Li, C. Zhang, and S. Zhao, "Time-sync comments denoising via graph convolutional and contextual encoding," *Pattern Recognition Letters*, vol. 135, pp. 256–263, 2020.

[75] Z. Liao, Y. Xian, X. Yang, Q. Zhao, C. Zhang, and J. Li, "Tscset: A crowdsourced time-sync comment dataset for exploration of user experience improvement," in *IUI*, 2018.

[76] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," *VLDB*, 2010.

[77] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 3243–3253.

[78] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, B. Bonet and S. Koenig, Eds., AAAI Press, 2015, pp. 2181–2187.

[79] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, 2003.

[80] C. Lo, D. Frankowski, and J. Leskovec, "Understanding behaviors that lead to purchasing: A case study of pinterest," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 531–540.

[81] B. Loni, R. Pagano, M. A. Larson, and A. Hanjalic, "Bayesian personalized ranking with multi-channel user feedback," in *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, S. Sen, W. Geyer, J. Freyne, and P. Castells, Eds., ACM, 2016, pp. 361–364.

[82] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4765–4774.

[83] W. Ma, M. Zhang, Y. Cao, W. Jin, C. Wang, Y. Liu, S. Ma, and X. Ren, "Jointly learning explainable rules for recommendation with knowledge graph," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, Eds., ACM, 2019, pp. 1210–1221.

[84] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[85] J. J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, Eds., ACM, 2013, pp. 165–172.

[86] Microsoft, *Power bi — interactive data visualization bi tools*, https://powerbi.microsoft.com, 2019.

[87] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 3111–3119.

[88] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[89] S. Moon, P. Shah, A. Kumar, and R. Subba, "OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, 2019, pp. 845–854.

[90] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*

(ICML-10), June 21-24, 2010, Haifa, Israel, J. Fürnkranz and T. Joachims, Eds., Omnipress, 2010, pp. 807–814.

[91] *Natural Language Generation*, 1998.

[92] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*, Springer, 1998, pp. 355–368.

[93] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, P. Langley, Ed., Morgan Kaufmann, 2000, pp. 663–670.

[94] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[95] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, L. Getoor and T. Scheffer, Eds., Omnipress, 2011, pp. 809–816.

[96] J. Oramas, K. Wang, and T. Tuytelaars, "Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks," *arXiv preprint arXiv:1712.06302*, 2017.

[97] H. Park, H. Jeon, J. Kim, B. Ahn, and U Kang, "Uniwalk: Explainable and accurate recommendation for rating and network data," *arXiv preprint arXiv:1710.07134*, 2017.

[98] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543.

[99] M. Pham, S. Alse, C. A. Knoblock, and P. Szekely, "Semantic labeling: A domain-independent approach," in *ISWC*, Springer, 2016, pp. 446–462.

[100] G. K. Pullum, "The land of the free and the elements of style," *English Today*, vol. 26, no. 2, pp. 34–44, 2010.

[101] Z. Qin, Y. Xian, F. Zhang, and D. Zhang, "Mimu: Mobile wifi usage inference by mining diverse user behaviors," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–22, 2020.

[102] M. Qu and J. Tang, "Probabilistic logic neural networks for reasoning," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 7710–7720.

[103] G. Quercini and C. Reynaud, "Entity discovery and annotation in tables," in *EDBT*, ACM, 2013, pp. 693–704.

[104] S. K. Ramnandan, A. Mittal, C. A. Knoblock, and P. Szekely, "Assigning semantic labels to data sources," in *ESWC*, Springer, 2015, pp. 403–417.

[105] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009.

[106] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, ACM, 1994, pp. 175–186.

[107] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 1135–1144.

[108] ——, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S. A. McIlraith and K. Q. Weinberger, Eds., AAAI Press, 2018, pp. 1527–1535.

[109] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag, "Learning a health knowledge graph from electronic medical records," *Scientific reports*, vol. 7, no. 1, pp. 1–11, 2017.

[110] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016, vol. 10.

[111] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., Curran Associates, Inc., 2007, pp. 1257–1264.

[112] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, Eds., ACM, 2001, pp. 285–295.

[113] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data mining and knowledge discovery*, 2001.

[114] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, Springer, 2018, pp. 593–607.

[115] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[116] S. Serrano and N. A. Smith, "Is attention interpretable?" In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, 2019, pp. 2931–2951.

[117] A. Sharma and D. Cosley, "Do social explanations work? studying and modeling the effects of social explanations in recommender systems," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 1133–1144.

[118] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[119] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *KDD*, 2008.

[120] S. Subramanian, B. Bogin, N. Gupta, T. Wolfson, S. Singh, J. Berant, and M. Gardner, "Obtaining faithful interpretations from compositional neural networks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, 2020, pp. 5594–5608.

[121] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *7th International Conference on Learning*

Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.

[122] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[123] G. Theocharous, P. S. Thomas, and M. Ghavamzadeh, "Personalized ad recommendation systems for life-time value optimization with guarantees," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Q. Yang and M. J. Wooldridge, Eds., AAAI Press, 2015, pp. 1806–1812.

[124] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: Explaining recommendations using tags," in *Proceedings of the 14th international conference on Intelligent user interfaces*, 2009, pp. 47–56.

[125] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, A. Cuzzocrea, J. Allan, N. W. Paton, D. Srivastava, R. Agrawal, A. Z. Broder, M. J. Zaki, K. S. Candan, A. Labrinidis, A. Schuster, and H. Wang, Eds., ACM, 2018, pp. 417–426.

[126] H. Wang, F. Zhang, X. Xie, and M. Guo, "DKN: deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, P. Champin, F. L. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds., ACM, 2018, pp. 1835–1844.

[127] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds., ACM, 2019, pp. 968–977.

[128] N. Wang, H. Wang, Y. Jia, and Y. Yin, "Explainable recommendation via multi-task learning in opinionated text data," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 165–174.

[129] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua, "KGAT: knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchor-*

age, AK, USA, August 4-8, 2019, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds., ACM, 2019, pp. 950–958.

[130] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," *AAAI*, 2019.

[131] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, and X. Xie, "A reinforcement learning framework for explainable recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018, pp. 587–596.

[132] J. Wu and R. Mooney, "Faithful multimodal explanation for visual question answering," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy: Association for Computational Linguistics, 2019, pp. 103–112.

[133] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[134] *Wwt*, https://www.cse.iitb.ac.in/~sunita/wwt/, (Accessed on 08/17/2020).

[135] Y. Xian, Z. Fu, Q. Huang, S. Muthukrishnan, and Y. Zhang, "Neural-symbolic reasoning over knowledge graph for multi-stage explainable recommendation," *AAAI DLGMA Workshop*, 2020.

[136] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, and Y. Zhang, "Reinforcement knowledge graph reasoning for explainable recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, and F. Scholer, Eds., ACM, 2019, pp. 285–294.

[137] Y. Xian, Z. Fu, H. Zhao, Y. Ge, X. Chen, Q. Huang, S. Geng, Z. Qin, G. de Melo, S. Muthukrishnan, and Y. Zhang, "CAFE: coarse-to-fine neural symbolic reasoning for explainable recommendation," in *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, M. d'Aquin, S. Dietze, C. Hauff, E. Curry, and P. Cudré-Mauroux, Eds., ACM, 2020, pp. 1645–1654.

[138] Y. Xian, H. Zhao, T. Y. Lee, S. Kim, R. Rossi, Z. Fu, G. de Melo, and S Muthukrishnan, "Exacta: Explainable column annotation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3775–3785.

[139] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 564–573.

[140] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds., ACM, 2018, pp. 974–983.

[141] M. Yu, S. Chang, Y. Zhang, and T. Jaakkola, "Rethinking cooperative rationalization: Introspective extraction and complement control," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 4094–4103.

[142] D. Zhang, Y. Suhara, J. Li, M. Hulsebos, Ç. Demiralp, and W.-C. Tan, "Sato: Contextual semantic type detection in tables," *arXiv preprint arXiv:1911.06311*, 2019.

[143] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 353–362.

[144] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, 2019.

[145] W. Zhang, Q. Yuan, J. Han, and J. Wang, "Collaborative multi-level embedding learning from reviews for rating prediction," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, S. Kambhampati, Ed., IJCAI/AAAI Press, 2016, pp. 2986–2992.

[146] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-n recommendation with heterogeneous information sources," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, E. Lim, M. Winslett, M. Sanderson, A. W. Fu, J. Sun, J. S. Culpepper, E. Lo, J. C. Ho, D. Donato, R. Agrawal, Y. Zheng, C. Castillo, A. Sun, V. S. Tseng, and C. Li, Eds., ACM, 2017, pp. 1449–1458.

[147] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *Foundations and Trends in Information Retrieval*, 2020.

[148] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, S. Geva, A. Trotman, P. Bruza, C. L. A. Clarke, and K. Järvelin, Eds., ACM, 2014, pp. 83–92.

[149] K. Zhao, X. Wang, Y. Zhang, L. Zhao, Z. Liu, C. Xing, and X. Xie, "Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, Eds., ACM, 2020, pp. 239–248.

[150] W. X. Zhao, G. He, K. Yang, H. Dou, J. Huang, S. Ouyang, and J.-R. Wen, "Kb4rec: A data set for linking knowledge bases with recommender systems," *Data Intelligence*, vol. 1, no. 2, pp. 121–136, 2019.

[151] Z. Zhao, Z. Cheng, L. Hong, and E. H. Chi, "Improving user topic interest profiles by behavior factorization," in *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, A. Gangemi, S. Leonardi, and A. Panconesi, Eds., ACM, 2015, pp. 1406–1416.

[152] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "DRN: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, P. Champin, F. L. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds., ACM, 2018, pp. 167–176.

[153] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, M. de Rijke, M. Shokouhi, A. Tomkins, and M. Zhang, Eds., ACM, 2017, pp. 425–434.

[154] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *Aaai*, vol. 8, 2008, pp. 1433–1438.