EXACTA ⊙: Explainable Column Annotation

Yikun Xian[1*], Handong Zhao[2†], Tak Yeon Lee[2], Sungchul Kim[2], Ryan Rossi[2]
Zuohui Fu[1], Gerard de Melo[1], S. Muthukrishnan[1]
[1]Rutgers University, New Brunswick, NJ [2]Adobe Research, San Jose, CA
yx150@cs.rutgers.edu,{hazhao,talee,sukim,ryrossi}@adobe.com
{zuohui.fu,gerard.demelo}@rutgers.edu,muthu@cs.rutgers.edu

ABSTRACT

Column annotation, the process of annotating tabular columns with labels, plays a fundamental role in digital marketing data governance. It has a direct impact on how customers manage their data and facilitates compliance with regulations, restrictions, and policies applicable to data use. Despite substantial gains in accuracy brought by recent deep learning-driven column annotation methods, their incapability of explaining why columns are matched with particular 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 ascertain the correctness of downstream applications. This work sheds new light on the explainable column annotation problem, the first of its kind column annotation task. To achieve this, we propose a new approach called EXACTA, which conducts multi-hop knowledge graph reasoning using inverse reinforcement learning to find a path from a column to a potential target label while ensuring both annotation performance and explainability. We experiment on four benchmarks, both publicly available and real-world ones, and undertake a comprehensive analysis on the 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 for annotated columns to facilitate human examination.

CCS CONCEPTS

• **Information systems** → *Data analytics*; *Data mining*.

KEYWORDS

Column Annotation, Explainability, Knowledge Graph Reasoning

[1]Data stewards are responsible for interpreting regulations, contractual restrictions, and policies, and applying them directly to data, and thus are central to data governance.

* This work was partially done during the author's internship at Adobe Research.
† Corresponding author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467211

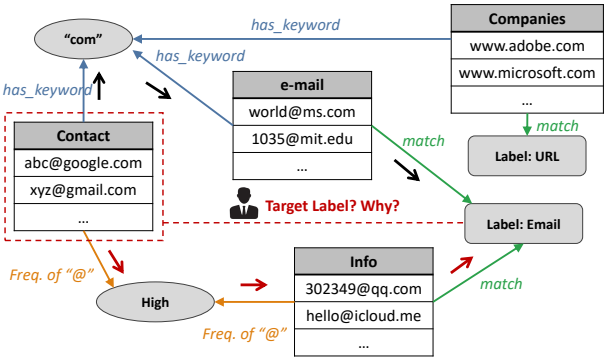


Figure 1: Explainable column annotation via multi-step reasoning over a KG to find the target label "Email" for the source column "Contact" with the accompanying explainable paths marked by red and black arrows.

ACM Reference Format:
Yikun Xian, Handong Zhao, Tak Yeon Lee, Sungchul Kim, Ryan Rossi, Zuohui Fu, Gerard de Melo, S. Muthukrishnan. 2021. EXACTA ⊙: Explainable Column Annotation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467211

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* [6, 15, 22, 28], 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) [1], industry experts such as data stewards and marketers are required to manually verify the correctness of the labeling, 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 an explanation to the experts to justify why particular decisions are made. We refer to this as the *explainable column annotation* task. Existing approaches are mostly accuracy-driven, seeking superior annotation performance via modern deep

neural networks [6, 7, 15, 42]. However, they have critical limitations in industry scenarios due to their lack of explainability and trustworthiness for the experts who conduct manual verification.

To meet both requirements, knowledge graph (KG) reasoning [20] has been widely adopted in link prediction [23, 31, 41], recommendation [39, 40], 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, 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 a candidate label node. The inferred path directly reflects the multi-step decision-making and hence can serve as an explanation for the prediction. As an example, in Fig. 1, the column "*Contact*" is expected to be matched with the label "*Email*", which can be reached by navigating along the evidence path *"Contact"* $\xrightarrow{\textit{freq. of "@"}}$ *"com"* $\xrightarrow{\textit{freq. of "@"}}$ *Column "Info"* $\xrightarrow{match}$ *"Email"*.

Recent work on KG reasoning primarily relies on reinforcement learning (RL) techniques [23, 37, 41], 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 Fig. 1, marked by red and black arrows, respectively. However, the black-arrowed path is intuitively less explainable than the other one due to the keyword "com" being too vague with regard to the label. If we only define a sparse reward on the last step at 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, a common alternative solution is to leverage imitation learning (IL), which requires high-quality trajectories to learn a policy via supervised learning or behavior cloning [16, 26, 44]. In real industry settings, 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, 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 address this, we propose a novel KG reasoning method named 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 harnessed to guide policy learning such that the 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-labelled paths to train the policy, but can still produce faithfully explainable paths for its predictions. We experiment on four diverse benchmarks, both publicly available and real industrial ones, and the results not only demonstrate better column annotation performance compared to various baselines, but, importantly, also show better explainability 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 EXACTA, a novel IRL-based KG reasoning approach, that can automatically learn a noise-tolerant reward function from noisy input paths to guide the policy learning.
- We experiment on four offline real-world benchmarks and conduct an online simulation of explainable column annotation, observing promising results of EXACTA in annotation performance.
- We also systematically evaluate the explainability of our model in terms of perceived explainability, robustness, and faithfulness of the path-based explanations.

2 PRELIMINARIES

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* $\mathcal{Y} \subseteq \mathcal{E}$. 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 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.

(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 rewards, 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 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 [26]. 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) [44], which models the probability of a trajectory by the maximum entropy principle [17], 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.

3 OUR METHOD

In this section, we propose a new method called EXACTA, which casts explainable column annotation as a KG-based IRL problem. As illustrated in Fig. 2, it 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.

3.1 Formulation as IRL Problem

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 $s_1 = (x, x)$. 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 work [23, 41], 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[2], 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 estimated by the function $R_\phi(s_t, a_t)$ with parameters ϕ.

IRL-based Objective Given a set of noisy (less explainable) paths of correctly labeled columns, we aim to learn the reward function R_ϕ to indicate 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 $\Sigma_\omega(s_t) = \text{diag}(\text{FF}_\omega(s_t))$, which is a diagonal matrix with values approximated by a multilayer 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 via p_ω as:

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

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 Eq. 1. Inspired by ME-IRL [44], we also model the probability $p(\tilde{\tau})$ with the reward function $R_\phi(s_t, a_t)$ under the maximum-entropy principle as

$$p_{\phi,\omega}(\tilde{\tau}) = \frac{1}{Z_{\omega,\phi}} p(s_1) \prod_{t=1}^{T} \int_{a_t} e^{R_\phi(s_t,a_t)} p_\omega(\tilde{a}_t|s_t, a_t) \mathrm{d}a_t, \tag{2}$$

Here $Z_{\omega,\phi}$ 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 Eq. 2, we can simplify the model as follows:

$$p_{\phi,\omega}(\tilde{\tau}) = \frac{1}{Z_{\omega,\phi}} p(s_1) \prod_{t=1}^{T} \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} \mathrm{d}a_t, \tag{3}$$

$$f_{\phi,\omega}(s_t, a_t, \tilde{a}_t) = R_\phi(s_t, a_t) - \frac{1}{2}\left(D^2_\omega(\tilde{a}_t) + \log\det(\Sigma_\omega(s_t))\right) \tag{4}$$

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.

[2]We adopt TransE [4] in this work, which can be replaced by other KG embeddings.

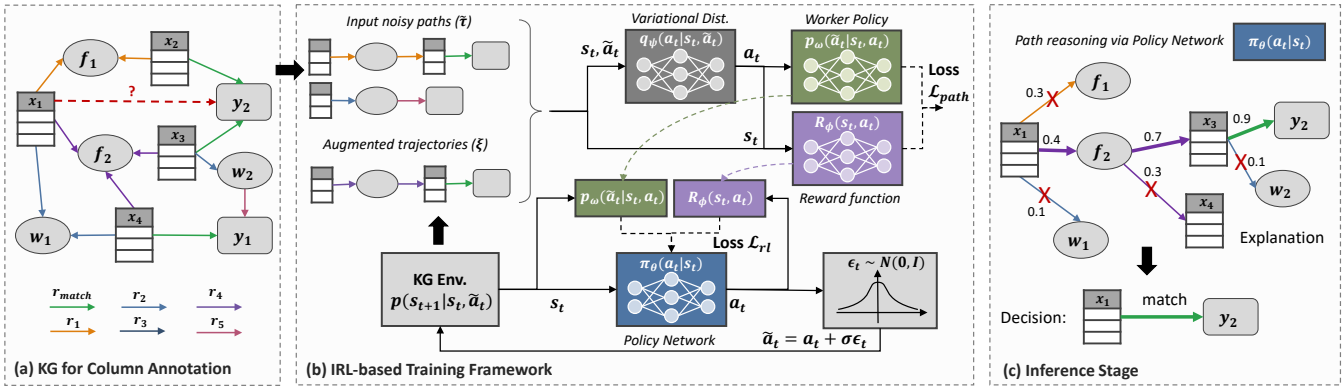


Figure 2: Pipeline of EXACTA. (a) A KG is constructed with columns, labels, and explainable features. (b) Given noisy paths, the reward function and policy network are iteratively learned under the IRL framework with explicit noise modeling via the worker policy. (3) KG reasoning is conducted to generate an explainable path and predicted label for the column.

Eq. 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 $\tilde{\mathcal{D}}$:

$$\max_{\phi,\omega} \frac{1}{|\tilde{\mathcal{D}}|} \sum_{\tilde{\tau}\in\tilde{\mathcal{D}}} \sum_{t=1}^{T} \log \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} \mathrm{d}a_t - \log Z_{\phi,\omega}. \quad (5)$$

3.2 Training Framework

However, directly solving Eq. 5 is intractable in practice due to the integral over the large-scale continuous action space. We adopt the variational approach [18] to change the integral into an expectation by introducing additional variational distributions.

Specifically, for the first logarithm term in Eq. 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

$$\begin{aligned}
&\sum_{t=1}^{T} \log \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} \mathrm{d}a_t = \sum_{t=1}^{T} \log \mathbb{E}_{a_t\sim q_\psi}\left[\frac{e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}}{q_\psi(a_t|s_t,\tilde{a}_t)}\right] \\
&\geq \sum_{(s_t,\tilde{a}_t)\in\tilde{\tau}} \mathbb{E}_{a_t\sim q_\psi}\left[f_{\phi,\omega}(s_t,a_t,\tilde{a}_t) - \log q_\psi(a_t|s_t,\tilde{a}_t)\right] \quad (6) \\
&= \mathcal{L}_{\text{path}}(\phi,\omega,\psi;\tilde{\tau}),
\end{aligned}$$

where Eq. 6 is derived via Jensen's inequality and the expectation can easily be approximated via Monte Carlo methods [34]. Note that $\sum_{t=1}^{T} \log \int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)} \mathrm{d}a_t = \max_\psi \mathcal{L}_{\text{path}}(\phi,\omega,\psi;\tilde{\tau})$. Therefore, we can first estimate ψ by maximizing the objective $\mathcal{L}_{\text{path}}$ and then solve the original problem in Eq. 5, which is equivalent to

$$\max_{\phi,\omega} \frac{1}{|\tilde{\mathcal{D}}|} \sum_{\tilde{\tau}\in\tilde{\mathcal{D}}} \mathcal{L}_{\text{path}}(\phi,\omega,\hat{\psi};\tilde{\tau}) - \log Z_{\phi,\omega}, \quad (7)$$

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

Then, we can approach the partition function in Eq. 7 in a similar way, which can first be expanded as follows.

$$\begin{aligned}
\log Z_{\phi,\omega} &= \log \int_{\tilde{\tau}\in(\mathcal{S}\times\mathcal{A})^T} \left(p(s_1)\prod_{t=1}^{T}\int_{a_t} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}\mathrm{d}a_t\right)\mathrm{d}\tilde{\tau} \\
&= \log \int_{\xi\in(\mathcal{S}\times\mathcal{A}\times\mathcal{A})^T} p(s_1)\prod_{t=1}^{T} e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}\mathrm{d}\xi \quad (8)
\end{aligned}$$

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 [19]. 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 Eq. 8 to eliminate the integral via the variational approach:

$$\log Z_{\phi,\omega} = \log \mathbb{E}_{\xi\sim p_\theta}\left[\prod_{t=1}^{T}\frac{e^{f_{\phi,\omega}(s_t,a_t,\tilde{a}_t)}}{\pi_\theta(a_t|s_t)N(\epsilon_t|0,I)}\right] \quad (9)$$

$$\geq \mathbb{E}_{\xi\sim p_\theta}\left[\sum_{t=1}^{T} f_{\phi,\omega}(s_t,a_t,\tilde{a}_t) - \log\pi_\theta(a_t|s_t) + \frac{1}{2}\|\epsilon_t\|^2\right] \quad (10)$$

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

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 Eq. 10 is a constant based on the quadratic form of random variables and is ignored. It is easy to find that $\log Z_{\phi,\omega} = \max_\theta \mathcal{L}_{\text{rl}}(\phi,\omega,\theta)$, 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 [35] in this work) to learn the optimal policy $\pi_{\hat{\theta}}(a_t|s_t)$ with $\hat{\theta} = \arg\max_\theta \mathcal{L}_{\text{rl}}(\phi, \omega, \theta)$.

The ultimate goal is to maximize the following objective:

$$\mathcal{L}(\phi, \omega; \tilde{\mathcal{D}}) = \frac{1}{|\tilde{\mathcal{D}}|} \sum_{\tilde{\tau} \in \tilde{\mathcal{D}}} \mathcal{L}_{\text{path}}(\phi, \omega, \hat{\psi}; \tilde{\tau}) - \mathcal{L}_{\text{rl}}(\phi, \omega, \hat{\theta}). \quad (11)$$

The IRL-based learning framework is illustrated in Fig. 2(b) and the complete training pipeline is summarized in the Appendix.

3.3 Inference

In the inference stage, as shown in Fig. 2(c), 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 k outgoing edges by:

$$\{(r_t, e_t) \in \mathcal{G} \mid e_t \notin L_{t-1}, \text{rank}(\|\hat{\mathbf{a}}_t - [\mathbf{r}_t; \mathbf{e}_t]\|_2^2) \leq k\}. \quad (12)$$

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.

4 EXPERIMENTS

We extensively evaluate our method in terms of both the annotation performance and the explainability on two publicly available 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? (Section 4.2)
- **RQ2**: Does our method provide good explainability for column annotation despite the noisy input paths? (Section 4.3)
- **RQ3**: What other factors may influence the annotation performance? (Section 4.4)

4.1 Experimental Setup

Datasets We experiment with four real-world datasets from both public sources and industrial platforms. `WWT` is an open-sourced dataset [2] 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[3], 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 [5] for semantic data type detection on column cells. It consists of 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[4], 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 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.

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

Table 1: Statistics of the KGs 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

KG construction and path generation We construct a KG for each dataset by extracting explainable features from columns and labels following a previous study [15]. 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 four KGs are given in Table 1 and the details of KG construction are described in the Appendix. 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 an average of 1.756 paths that are deemed "somewhat explainable" per 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` [30] is an early heuristic approach that detects column types by collecting frequent lexical keywords across all columns. `DSL` [28] is a representative column annotation model that exploits column features and makes its prediction via a random forest.
- Deep neural networks: `ColNet` [6] is a deep neural model that relies on an external knowledge base to identify column types together with entity lookup voting. `Sherlock` [15] is a deep neural model that also utilizes the features extracted from columns. `SATO` [42] 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 [27] features, which are consistent with our KG-based model.
- KG-based approaches: `TransE` [4] is a KG embedding technique via vector translation operations. We also use it to initialize the state and action representation in our model. `RotatE` [36] is an advanced KG embedding modeled in a complex vector space. `ME-IRL` [44] is an IRL method that learns rewards from expert

Table 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				WebTable78				Marketing			
Methods	F1	Hits@1	Hits@3	Hits@5	F1	Hits@1	Hits@3	Hits@5	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	0.4268	0.4344	0.5155	0.5454	0.5362	0.5339	0.6122	0.6294
DSL [28]	0.3649	0.3957	0.5728	0.6464	0.7856	0.8009	0.9988	0.9994	0.5132	0.5262	0.6493	0.7093	0.6226	0.6440	0.6705	0.6831
ColNet [6]	0.4148	0.4332	0.6429	0.7125	0.7592	0.7706	0.9921	0.9994	0.5133	0.5173	0.6589	0.7217	0.7005	0.7202	0.9240	0.9328
Sherlock [15]	0.4437	0.4735	0.6524	0.7293	0.7724	0.7988	1.0000	1.0000	0.5714	0.5852	0.7183	0.7735	0.7400	0.7889	0.9901	0.9952
SATO [42]	0.4387	0.4664	0.6681	0.7296	0.7974	0.8188	1.0000	1.0000	0.6244	0.6279	0.7476	0.7873	0.7692	0.7975	0.9903	0.9960
TransE [4]	0.4347	0.4563	0.6924	0.7768	0.7771	0.8030	1.0000	1.0000	0.5352	0.5233	0.6867	0.7517	0.7567	0.7712	0.9874	0.9937
RotatE [36]	0.4835	0.5150	0.7282	0.7911	0.8111	0.8303	1.0000	1.0000	0.5858	0.5753	0.7233	0.7853	0.7661	0.7953	0.9863	0.9937
ME-IRL [44]	0.3250	0.3372	0.5305	0.5968	0.7250	0.7458	0.9154	0.9525	0.4483	0.4928	0.6193	0.6598	0.6057	0.6209	0.8815	0.9052
KGRL [23]	0.4049	0.4304	0.6623	0.7453	0.7927	0.8102	0.9858	0.9994	0.5025	0.5121	0.6805	0.7422	0.7397	0.7428	0.9808	0.9916
EXACTA (ours)	**0.5080**	**0.5308**	**0.7353**	**0.7966**	**0.8406**	**0.8527**	**1.0000**	**1.0000**	**0.6358**	**0.6347**	**0.7566**	**0.7949**	**0.7973**	**0.8155**	**0.9921**	**0.9976**

trajectories via the max-entropy principle. `KGRL` [23] 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).

4.2 Experiment on Column Annotation

We first evaluate the column annotation performance of our method compared to the selected baselines (**RQ1**). The benchmark results of all methods across the four datasets are reported in Table 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 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 deliver an accurate labeling 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 representations, 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.

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**).

4.3.1 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.[5] 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 scores of 4 algorithms given by 5 experts in Table 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

[5]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.

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

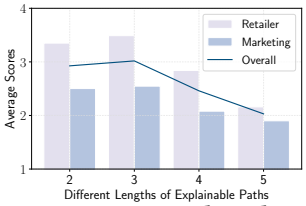
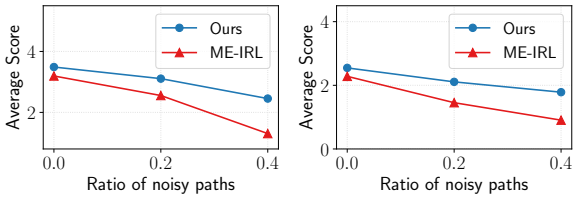


Figure 3: Perceived explainability of our method on various path lengths.



(a) Retailer *(b) Marketing*

Figure 4: Perceived explainability of our method and ME-IRL when trained with varying percentages of noisy paths.

the RL method, which merely has comparable performance with random path sampling. This implies that such RL-based methods completely fail to distinguish the explainability of different paths between the same column–label pair.

4.3.2 Path length on perceived explainability. We further evaluate the influence of the path length on the 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 Fig. 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.

4.3.3 Robustness to Very Noisy Input Paths. Recall that when generating training paths annotated by crowdsourced workers in Section 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 Fig. 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.

4.3.4 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 [8], 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 feature nodes connecting columns). With the annotation metric $m(\cdot)$, e.g., Hits@k, we define *comprehensiveness* h_{com} and *sufficiency* h_{suf} as:

$$h_{\text{com}} = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{x \in \mathcal{X}_{\text{test}}} \frac{m(f(x)) - m(f(x \backslash v_k))}{m(f(x))} \tag{13}$$

$$h_{\text{suf}} = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{x \in \mathcal{X}_{\text{test}}} \frac{m(f(x)) - m(f(v_k))}{m(f(x))} \tag{14}$$

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 [32], Anchor [33]) 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 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 outgoing edges of the column that do not belong to the extracted features.

The results are reported in Table 4. A 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 extracted top-k features only, it is reasonable that SATO's 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.

Table 4: Evaluation of faithfulness of the explainability

Methods	Retailer		Marketing	
	$h_{\text{com}} \uparrow$	$h_{\text{suf}} \downarrow$	$h_{\text{com}} \uparrow$	$h_{\text{suf}} \downarrow$
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**

4.4 Ablation Study

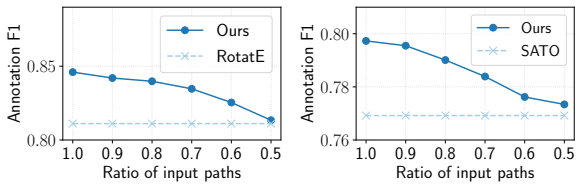
We further seek a better understanding of what other factors may influence the performance of our model (**RQ3**).

4.4.1 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 two industrial 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 Fig. 5. 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 labeling of high-quality explainable paths.

4.4.2 Influence of maximum path length. We further evaluate how different maximum path length (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 two industrial datasets are reported in Fig. 6. We observe that the best performance is consistently achieved when $T = 3$, which achieves results comparable to the case of $T = 2$, but performs substantially better than when operating with longer horizons. The underlying reason for these findings is that when reasoning for longer numbers of steps, one is more likely to arrive at spurious nodes instead of a potentially correct target.

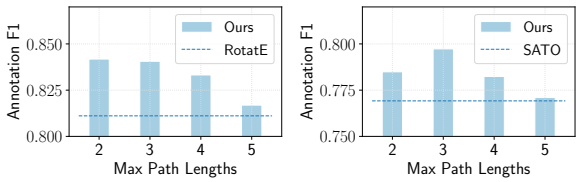
4.5 Online Simulation

Finally, we internally evaluate our model 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. Thus, 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



(a) Retailer *(b) Marketing*

Figure 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) Retailer *(b) Marketing*

Figure 6: Column annotation performance (F1) of our method when varying the maximum step size in reasoning compared to the best baseline on two industry datasets.

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 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 [9, 29]) are not considered in this paper 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 [10, 24] adopt regular expression and dictionary methods to match columns with predefined patterns or keywords. Ramnandan *et al.* [30] 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. Limaye *et al.* [22] annotate tables with types using probabilistic graphical models. Pham *et al.* [28] 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.* [6] 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.* [7] propose another hybrid deep neural network by exploiting table locality features and inter-column semantic features. Zhang *et al.* [42] integrate deep learning and topic

modeling to annotate columns with semantic types. Hulsebos *et al.* [15] 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 models due to the representational power of deep neural networks. However, the issue of explainability remains under-explored, despite being crucial in 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 [23, 31, 41], recommendation [38–40, 43], conversational systems [11–13], content denoising [21]. For instance, Ai *et al.* [3] 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 recommendations, the problem of column annotation is 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 comprehensively evaluate the path explainability.

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 approach 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.

REFERENCES

[1] [n.d.]. General Data Protection Regulation - Wikipedia. https://en.wikipedia.org/wiki/General_Data_Protection_Regulation. (Accessed on 08/17/2020).
[2] [n.d.]. WWT. https://www.cse.iitb.ac.in/~sunita/wwt/. (Accessed on 08/17/2020).
[3] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.
[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.
[5] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *VLDB* (2008).
[6] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. Colnet: Embedding the semantics of web tables for column type prediction. In *AAAI*, Vol. 33. 29–36.
[7] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. Learning semantic annotations for tabular data. In *IJCAI*.
[8] Jay DeYoung, Sarthak Jain, Nazneen Rajani, E. Lehman, Caiming Xiong, R. Socher, and Byron C. Wallace. 2020. ERASER: A Benchmark to Evaluate Rationalized NLP Models. *ArXiv* abs/1911.03429 (2020).
[9] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. 2017. Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *ISWC*. Springer, 260–277.
[10] Open Knowledge Foundation. 2019. messytables: Tools for parsing messy tabular data. https://github.com/okfn/messytables.
[11] Zuohui Fu, Yikun Xian, Yongfeng Zhang, and Yi Zhang. 2020. Tutorial on Conversational Recommendation Systems. In *RecSys*.
[12] Zuohui Fu, Yikun Xian, Yongfeng Zhang, and Yi Zhang. 2021. WSDM 2021 Tutorial on Conversational Recommendation Systems. In *WSDM*.
[13] Zuohui Fu, Yikun Xian, Yaxin Zhu, Yongfeng Zhang, and Gerard de Melo. 2020. COOKIE: A Dataset for Conversational Recommendation over Knowledge Graphs in E-commerce. *arXiv preprint arXiv:2008.09237* (2020).
[14] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. Openke: An open toolkit for knowledge embedding. In *EMNLP*. 139–144.
[15] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *KDD*.
[16] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *CSUR* (2017).
[17] Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical review* 106, 4 (1957), 620.
[18] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning* 37, 2 (1999), 183–233.
[19] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[20] Freddy Lecue. 2020. On the role of knowledge graphs in explainable AI. *Semantic Web* 11, 1 (2020), 41–51.
[21] Zhenyu Liao, Yikun Xian, Jiangfeng Li, Chenxi Zhang, and Shengjie Zhao. 2020. Time-sync comments denoising via graph convolutional and contextual encoding. *Pattern Recognition Letters* 135 (2020), 256–263.
[22] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. *VLDB* (2010).
[23] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *EMNLP* (2018).
[24] Microsoft. 2019. Power BI | Interactive Data Visualization BI Tools. https://powerbi.microsoft.com.
[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
[26] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *ICML*, Vol. 1. 2.
[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.
[28] Minh Pham, Suresh Alse, Craig A Knoblock, and Pedro Szekely. 2016. Semantic labeling: a domain-independent approach. In *ISWC*. Springer, 446–462.
[29] Gianluca Quercini and Chantal Reynaud. 2013. Entity discovery and annotation in tables. In *EDBT*. ACM, 693–704.
[30] S Krishnamurthy Ramnandan, Amol Mittal, Craig A Knoblock, and Pedro Szekely. 2015. Assigning semantic labels to data sources. In *ESWC*. Springer, 403–417.
[31] Saed Rezayi, Handong Zhao, Sungchul Kim, Ryan A. Rossi, Nedim Lipka, and Sheng Li. 2021. Edge: Enriching Knowledge Graph Embeddings with External Text. In *NAACL*.
[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should i trust you?" Explaining the predictions of any classifier. In *KDD*.
[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI*.
[34] Reuven Y Rubinstein and Dirk P Kroese. 2016. *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons.
[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv:1707.06347* (2017).
[36] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *ICLR* (2019).
[37] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[38] Yikun Xian, Zuohui Fu, Qiaoying Huang, Shan Muthukrishnan, and Yongfeng Zhang. 2020. Neural-Symbolic Reasoning over Knowledge Graph for Multi-Stage Explainable Recommendation. *arXiv preprint arXiv:2007.13207* (2020).
[39] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. *SIGIR* (2019).
[40] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard de Melo, S. Muthukrishnan, and Yongfeng Zhang. 2020. CAFE: Coarse-to-Fine Neural Symbolic Reasoning for Explainable Recommendation. In *CIKM*. ACM, 1645–1654.
[41] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *EMNLP* (2017).
[42] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2019. Sato: Contextual Semantic Type Detection in Tables. *arXiv preprint arXiv:1911.06311* (2019).
[43] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. 2021. Faithfully Explainable Recommendation via Neural Logic Reasoning. *NAACL*.
[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning.. In *Aaai*, Vol. 8. 1433–1438.

APPENDIX

A IMPLEMENTATION DETAILS

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

A.1 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 a previous study [15], 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 the "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 the "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 [25] 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.

A.2 Training Pipeline

The complete training pipeline of EXACTA is summarized in Alg. 1.

A.3 Neural Network Architectures and Hyperparameters

For the TransE[4] 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 [14], 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

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

  where $W_1 \in \mathbb{R}^{200\times256}$ and $W_2, W_3 \in \mathbb{R}^{256\times200}$.
- The reward function is defined as

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

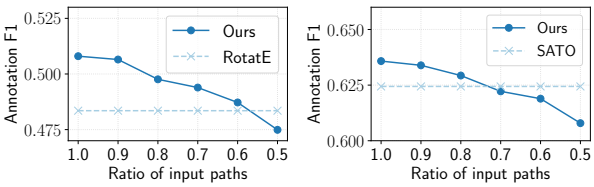
  where $W_4 \in \mathbb{R}^{400\times256}$ and $W_5 \in \mathbb{R}^{256\times1}$.
- The state-dependent covariance matrix in the worker policy is:

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

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

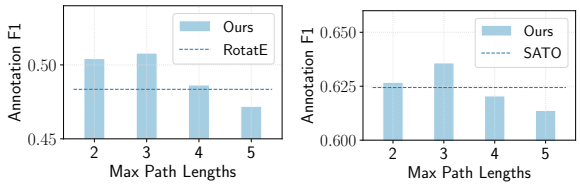
Algorithm 1 Training Pipeline

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 $\sum_{\tilde{\tau}\in\tilde{\mathcal{D}}} \nabla_\psi \mathcal{L}_{\text{path}}(\phi, \omega, \psi; \tilde{\tau})$.
  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 $\mathcal{L}_{\text{rl}}(\omega, \phi, \theta)$ 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



(a) WWT *(b) WebTable78*

Figure 7: Column annotation performance (F1) of our method trained with various portions of input paths compared to the best baseline on two open datasets.



(a) WWT *(b) WebTable78*

Figure 8: Column annotation performance (F1) of our method with various output path lengths compared to the best baseline on two open datasets.

- The variational distribution is defined as

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

  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 [35] 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 $N_{\text{stat}} = 20$, $N_{\text{char}} = 20$ and vocabulary size $|V_{\text{cell}}| = 15,000$.

B MORE EXPERIMENTAL RESULTS

B.1 Ablation Study

We also conduct the ablation study on the two open-source datasets `WWT` and `WebTable78`.

For the influence of the quantity of input paths, the results on the two datasets are illustrated in Fig. 7. For the influence of the maximum number of steps, the results are plotted in Fig. 8. We can see that both results are consistent with those on the two industrial datasets.

B.2 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 Fig. 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.

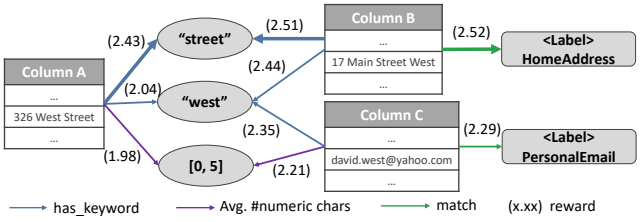


Figure 9: Visualization of learned rewards on a subgraph of the `Retailer` dataset. The explainable path with the highest cumulative rewards is highlighted in bold.