DIVINE: A Generative Adversarial Imitation Learning Framework for Knowledge Graph Reasoning

Ruiping Li and **Xiang Cheng**[*]
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing 100876, China
{liruiping,chengxiang}@bupt.edu.cn

Abstract

Knowledge graphs (KGs) often suffer from sparseness and incompleteness. Knowledge graph reasoning provides a feasible way to address such problems. Recent studies on knowledge graph reasoning have shown that reinforcement learning (RL) based methods can provide state-of-the-art performance. However, existing RL-based methods require numerous trials for path-finding and rely heavily on meticulous reward engineering to fit specific dataset, which is inefficient and laborious to apply to fast-evolving KGs. In this paper, we present DIVINE, a novel plug-and-play framework based on generative adversarial imitation learning for enhancing existing RL-based methods. DIVINE guides the path-finding process, and learns reasoning policies and reward functions self-adaptively through imitating the demonstrations automatically sampled from KGs. Experimental results on two benchmark datasets show that our framework improves the performance of existing RL-based methods without extra reward engineering.

1 Introduction

Knowledge graphs (Suchanek et al., 2007; Auer et al., 2007; Bollacker et al., 2008; Carlson et al., 2010; Vrandečić and Krötzsch, 2014), typically composed of massive relational triples, are useful resources for many downstream natural language processing applications such as information extraction and question answering. Although existing KGs have an extraordinarily large scale, they are still highly incomplete (Min et al., 2013), leading extensive research efforts on automated inference of missing information from observed evidence. In this paper, we focus on the problem of multi-hop reasoning in KGs, which learns explicit inference formulas from existing triples to complete missing ones.

[*]Corresponding author

To tackle multi-hop reasoning, various path-based methods (Lao et al., 2011; Gardner et al., 2013, 2014; Guu et al., 2015; Neelakantan et al., 2015; Toutanova et al., 2016; Das et al., 2017) have been proposed, which leverage the elaborately selected relational paths in KGs as the reasoning evidence. However, such evidential paths are obtained by random walks, which inevitably introduces inferior or even noisy paths. To address this problem, the RL-based methods, such as DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018), strive for more reliable evidential paths by policy-conditioned walking and achieve state-of-the-art performance. They formulate the path-finding problem as a Markov decision process where their policy-based agents continuously choose the most promising relation for state transition based on the current state and reasoning policy. Once a relational path found, the reasoning policy is updated by a reward function according to the path quality. Finally, through such a trial-and-error process, the well-trained policy-based agent can be used to find evidential paths for predictions.

However, these RL-based methods still suffer from the following pain points. Firstly, they tend to require numerous trials from scratch to find a reliable evidential path since the action space can be very large due to the complexity of KGs, which leads to poor convergence properties. Secondly, and most importantly, an efficient trial-and-error optimization in RL requires designing a reward function manually to fit the specific dataset. However, such reward engineering depends on meticulous artificial design with domain expertise, which can be significantly challenging in practice (Ng et al., 2000). In particular, these RL-based methods are extremely sensitive to their reward functions, where a little variation may lead to a significant fluctuation of the reasoning performance.

Therefore, for different datasets, the reward functions in the RL-based methods need manual adjustments to achieve a good performance, which will not only be inefficient and laborious but also difficult to adapt to the rapid evolutions of real-world KGs (Shi and Weninger, 2018).

In this paper, we present a novel plug-and-play framework based on generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) for enhancing existing RL-based methods, which is referred to as DIVINE for "Deep Inference via Imitating Non-human Experts". DIVINE trains a reasoner, consisting of a generator and a discriminator, from demonstrations by employing generative adversarial training, where the generator can be any of the policy-based agents in existing RL-based methods and the discriminator can be considered as a self-adaptive reward function. In this way, for different datasets, the reward functions can be automatically tuned to approximate the optimal performance, eliminating extra reward engineering and manual interventions. In particular, to enable the policy-based agent to find more diverse evidential paths for predictions, we propose a path-based GAIL method, which can learn the reasoning policy by imitating the path-level semantic features of the demonstrations. In addition, to acquire demonstrations without extra manual labor, we design an automated sampler for our framework to dynamically sample relational paths from KGs as the demonstrations according to the specific environment of each entity.

In summary, our contributions are threefold:

- We present a plug-and-play framework based on GAIL to enhance existing RL-based reasoning in KGs by learning reasoning policies and reward functions through imitating demonstrations. To the best of our knowledge, we are the first to introduce GAIL into the field of knowledge graph reasoning.
- We propose a path-based GAIL method to encourage the diversity of evidential paths and design an automated sampler for our framework to sample demonstrations without extra manual labor.
- We conduct extensive experiments on two benchmark datasets. The experimental results illustrate that our framework improves the performance of the current state-of-the-art RL-based methods while eliminating extra reward engineering.

2 Related Work

Automated reasoning on KGs has been a long-standing task for natural language processing. In recent years, various embedding-based methods using tensor factorization (Nickel et al., 2011; Bordes et al., 2013; Riedel et al., 2013; Yang et al., 2014; Trouillon et al., 2017) or neural network models (Socher et al., 2013) have been developed, where they learn a projection which maps the triples into a continuous vector space for further tensor operations. Despite the impressive results they achieved, most of them lack the ability to capture chains of multi-hop reasoning patterns contained in paths.

To address the limitation of the embedding-based methods, a series of path-based methods have been proposed, which consider the selected relational paths as reasoning evidence. Lao et al. (2011) propose the Path-Ranking Algorithm (PRA) which uses random walks for path-finding. Gardner et al. (2013, 2014) propose a variation on PRA which computes feature similarity in the vector space. To combine the embedding-based methods, several neural multi-hop models (Neelakantan et al., 2015; Guu et al., 2015; Toutanova et al., 2015, 2016; Das et al., 2017) are proposed which perform a hybrid reasoning. Nevertheless, the evidential paths they used are gathered by random walks, which might be inferior and noisy.

Recently, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018) were proposed to address the problem above by using reinforcement learning, where they are committed to learn a policy which guides the agent to find more superior evidential paths to maximize the expected reward. Specifically, DeepPath parameterizes its policy with a fully-connected neural network and uses manual reward criteria, including global accuracy, efficiency and diversity, to evaluate the path quality. In the training phase, DeepPath applies the linear combination of these criteria as the positive reward while using a hand-craft constant as the negative penalty. As for MINERVA, it parameterizes its policy with a long short-term memory network (LSTM) and considers the path validity as the only reward criterion. In the training phase, MINERVA uses boolean value as a terminal signal to evaluate whether the current path reaches the target entity and manually tunes a moving average of cumulative discounted reward on different datasets for variance reduction.

3 Preliminaries

3.1 Knowledge Graph Reasoning

Given an incomplete knowledge graph $\mathcal{G} = \{(h, r, t)|h \in \mathcal{E}, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and relation set, respectively. There are two main tasks in knowledge graph reasoning, namely link prediction and fact prediction. Link prediction involves inferring the tail entity t given the head entity h and the query relation r_q, while fact prediction seeks to predict whether an unknown fact (h, r_q, t) holds or not. Recently, RL-based reasoning has become a popular approach for knowledge graph reasoning, which achieves state-of-the-art performance. In general, RL-based reasoning methods strive to find relational paths to tune their reasoning policies for predictions and formulate the path-finding problem as a Markov decision process (MDP). In such a process, the policy-based agent decides to take an action $a \in \mathcal{A}$ from the current state (i.e., the current entity and its context information) $s \in \mathcal{S}$ to reach the next one according to its reasoning policy π, where the action space is defined as all the relations in $\mathcal{G}$. In particular, each relational chain in the relational paths can be considered as a reasoning chain.

3.2 Imitation Learning

Imitation learning focuses on learning policies from demonstrations, which has achieved great success in solving reward engineering. The classical approach is to find the optimal reward function by inverse reinforcement learning (IRL) (Russell, 1998; Ng et al., 2000) to explain expert behaviors. However, IRL requires solving RL inside a learning loop, which can be expensive to run in large environments. Therefore, generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) has recently been proposed, which learns the expert policy with generative adversarial network (GAN) (Goodfellow et al., 2014), eliminating any intermediate IRL steps.

In GAIL, a generator G_θ is trained to generate trajectories matching the distribution of expert trajectories (i.e., demonstrations). Each trajectory τ is represented as a state-action sequence $[(s_t, a_t)]_{t=0}^{\infty}$ $(s_t \in \mathcal{S}, a_t \in \mathcal{A})$. In addition, a discriminator D_ω is learned to distinguish between the generated policy π_θ and expert policy π_E. For each training epoch, the discriminator is updated first with the gradient as:

$$\mathbb{E}_{\tau}[\nabla_\omega log(D(s,a))] + \mathbb{E}_{\tau_E}[\nabla_\omega log(1-D(s,a))], \quad (1)$$

where τ_E denotes the expert trajectories generated by π_E and the trajectory expectation can be calculated in the γ-discounted infinite horizon as:

$$\mathbb{E}_{\tau}[D(s,a)] = \sum_{t=0}^{\infty} \mathbb{E}[\gamma^t D(s_t, a_t)], \qquad (2)$$

where the discriminator here can be interpreted as a local reward function to provide feedback for the policy learning process. Then, the generator is updated with the cost function $\log(D(s, a))$ using the trust region policy optimization (TRPO) (Schulman et al., 2015). After sufficient adversarial training, the optimal policy $\hat{\pi}$ can be found by GAIL to rationalize the expert policy π_E.

4 Methodology

4.1 Framework Overview

As shown in Figure 1, our framework DIVINE consists of two modules, namely a generative adversarial reasoner and a demonstration sampler. In particular, the reasoner is composed of a generator and a discriminator. The generator can be any of the policy-based agents in existing RL-based methods and the discriminator can be interpreted as a self-adaptive reward function. For each query relation, the sampler and the generator are adopted respectively to automatically extract demonstrations and generate relational paths from the given KG. The discriminator is then used to evaluate the semantic similarity between the generated paths and demonstrations to update the generator.

After sufficient adversarial training between the generator and the discriminator alternatively, the well-trained policy-based agent (i.e., generator) can be used to find evidential paths matching the distribution of the demonstrations and make predictions by synthesizing these evidential paths.

4.2 Generative Adversarial Reasoner

In our framework, the reasoner is learned from the demonstrations through generative adversarial training. A straightforward approach is to directly apply GAIL to train the reasoner. In particular, the policy-based agent in the reasoner is trained to find evidential paths by imitating the state-action pairs in each expert trajectory (i.e., demonstration). However, such an approach may lead to

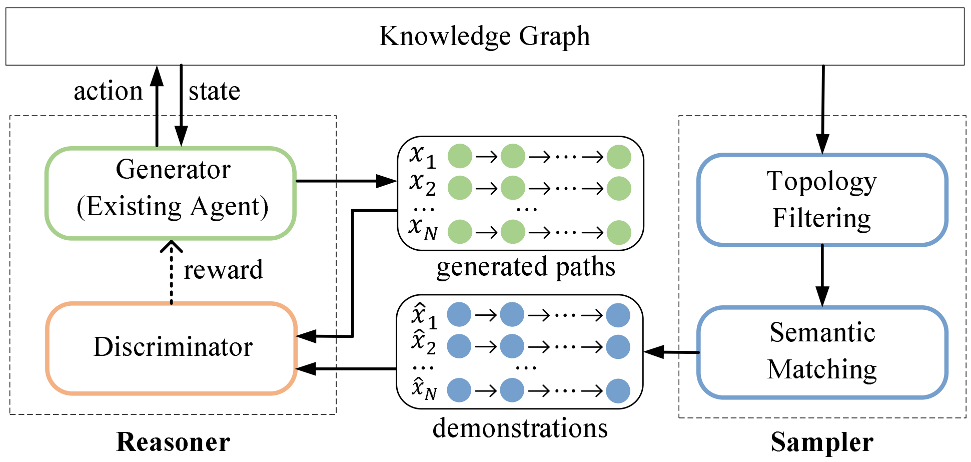

Figure 1: Overview of our framework DIVINE, where the generator can be any of the existing policy-based agent for knowledge graph reasoning and the discriminator can be considered as a self-adaptive reward function.

poor performance. The main reason lies in that the agent will tend to choose the same actions as in the expert trajectories under certain states, while ignoring many valuable evidential paths which are semantically similar to the expert trajectories but contain different reasoning chains.

Therefore, to encourage the agent to find more diverse evidential paths, it is desirable to train the agent by imitating each trajectory instead of each of its state-action pairs. In addition, in the scenario of knowledge graph reasoning, since the reasoning chains consist of only relations, the demonstrations do not necessarily contain the state information. In other words, the demonstrations can be composed of only relational paths.

Based on the above analysis, we propose a path-based GAIL method, where the reasoning policy is learned by imitating the path-level semantic features of the demonstrations which are composed of only relational paths.

In what follows, we first describe the two components of the reasoner, i.e., the generator and the discriminator. Then, we show how to extract the path-level semantic features.

Generator

The generator can be any of the policy-based agent in existing RL-based methods. We strive to enable the generator to find more diverse evidential paths matching the distribution of the demonstrations in the semantic space.

Discriminator

To better semantically distinguish between generated paths and demonstrations, we choose the convolutional neural networks (CNNs) to construct our discriminator D, as CNNs have shown high performance in semantic feature extraction from natural languages (Kim, 2014).

Semantic Feature Extraction

For each positive entity pair, we respectively pack the current generated paths and corresponding demonstrations in the same package form. For each package $\mathcal{P} = \{x_1, x_2, ..., x_N\}$ containing N relational paths, we encode the package to a real-valued matrix as:

$$\boldsymbol{p} = \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus ... \oplus \boldsymbol{x}_N, \tag{3}$$

where $\boldsymbol{x}_n \in \mathbb{R}^k$ is the k-dimensional path embedding and $\oplus$ denotes the concatenation operator for the package representation $\boldsymbol{p} \in \mathbb{R}^{N\times k}$. In particular, given a relational path $x = \{r_1, r_2, ...r_t, ...\}$, the path embedding $\boldsymbol{x}$ is encoded as:

$$\boldsymbol{x} = \sum_{r_t \in x} \boldsymbol{r}_t, \tag{4}$$

where each relation r_t is mapped into a real-valued embedding $\boldsymbol{r}_t \in \mathbb{R}^k$ pre-trained by TransE (Bordes et al., 2013).

After packing, we feed the package representation $\boldsymbol{p}$ into our discriminator D to parameterize its semantic features $D(\boldsymbol{p})$. Specifically, a convolutional layer activated by ReLU nonlinearity is first used to extract local features by sliding a kernel $\boldsymbol{\omega} \in \mathbb{R}^{h\times l}$ for a new feature map:

$$\boldsymbol{c} = ReLU(Conv(\boldsymbol{p}, \boldsymbol{\omega}) + \boldsymbol{b}_c), \tag{5}$$

where $\boldsymbol{b}_c$ denotes the bias term. Then, a fully-connected hidden layer and an output layer are used for further semantic feature extraction:

$$D(\boldsymbol{p}) = \sigma(\boldsymbol{W}_2 ReLU(\boldsymbol{W}_1 \boldsymbol{c})), \tag{6}$$

where the corresponding biases are not shown above for brevity and the output layer is normalized by a sigmoid function while other layers are activated by ReLU nonlinearity.

4.3 Demonstration Sampler

For imitation learning, the first prerequisite is to have high-quality demonstrations. However, due to the large scale and complexity of KGs, manually constructing a large number of reasoning demonstrations requires considerable time and expert efforts. Therefore, we design an automated sampler to sample reliable reasoning demonstrations from KGs without supervision and extra manual labor.

Static Demo Sampling

For each query relation, we use all positive entity pairs to sample demonstration candidates from the given KG. Specifically, for each positive entity pair, we use bi-directional breadth-first search to explore the shortest path between two entities. In particular, since the shorter paths incline to characterize more direct correlations between two entities, we prefer to use them for initialization to ensure the quality of demonstration candidates. As for the longer paths, despite their potential utility values, they are more likely to contain worthless or even noisy inference steps, thus we learn them only in the training phase. In doing so, we can get a demonstration set Ω_E which contains all the candidates we sampled. Finally, to accommodate the fixed input dimension of the discriminator D, we can simply select a subset $P_e \subseteq \Omega_E$ with the top N occurrence frequency, where N is normally much smaller than $|\Omega_E|$.

Dynamic Demo Sampling

Despite the simplicity of the static demo sampling method, the obtained demonstrations by this method are fixed and ignore the specific environment of each entity in the given KG. Therefore, we propose an improving method to dynamically sample demonstrations by taking the topological correlations of entities into consideration.

Given a positive entity pair $\langle e_{head}, e_{tail} \rangle$, we introduce a relational set $\mathcal{R}_h$ which contains all relations directly connected to e_{head}. For each reasoning attempt, $\mathcal{R}_h$ can be considered as the region of interest (ROI) of the agent to start reasoning, where the ROI related paths tend to be more relevant to the current entity pair. Thus, we refine the demonstration set by filtering out the demonstrations which begin from $\mathcal{R}_h$:

$$\Omega'_E = \{x \mid r_1(x) \in \mathcal{R}_h, x \in \Omega_E\} \;\; (\mathcal{R}_h \neq \varnothing), \quad (7)$$

where Ω_E is generated by the static demo sampling method and $r_1(x)$ denotes the first relation in relational path $x = \{r_1, r_2, ...r_t, ...\}$.

In most cases, we can obtain enough demonstrations in Ω'_E to select a subset $P_e \subseteq \Omega'_E$ in the same way as the static demo sampling method. However, due to the sparsity of data in KG, we may get insufficient demonstrations on long-tail entities. To solve this problem, we perform semantic matching to explore more demonstrations from the remaining candidates $\mathcal{C}_E = \Omega_E \setminus \Omega'_E$. Since the reasoning policy is updated based on the semantic similarities between the generated paths and demonstrations, candidates which are semantically similar to the current demonstrations are also instructive for the imitation process.

Inspired by the neighborhood attention for one-shot imitation learning (Duan et al., 2017), we use each demonstration in Ω'_E to query other candidates in correlation to itself. We adopt the dot product to measure the semantic matching similarity between two path embeddings:

$$\alpha_i = \sum_{x_j \in \Omega'_E} \bar{\boldsymbol{x}}_i \cdot \boldsymbol{x}_j \quad i = 1, 2, ..., |\mathcal{C}_E|, \quad (8)$$

where α_i represents the sum of matching scores between the current candidate $\bar{x}_i$ and existing demonstrations in Ω'_E. Finally, we iteratively select the candidate with the highest α to pad the refined demonstration set Ω'_E until accommodating the required input dimension N.

4.4 Training

In the training phase, all the positive entity pairs are used to generate demonstration candidates Ω_E for the imitation learning process. Specifically, for each positive entity pair, the demonstration sampler is required first to choose the corresponding demonstrations, while the generator is conducted to generate some relational paths. Then, the demonstrations are packed into package $\mathcal{P}_e$ and the generated paths are packed into different packages $\{\mathcal{P}_g \mid \mathcal{P}_g \subseteq \Omega_G\}$ according to their validity, i.e., whether the agent can reach the target entity along the current path, where Ω_G is the collection of all generated paths.

For each package pair $\langle \mathcal{P}_g, \mathcal{P}_e \rangle$, we train the discriminator D by minimizing its loss and expect it to be expert in distinguishing between $\mathcal{P}_e$ and $\mathcal{P}_g$. In addition, to make the adversarial training process more stable and effective, we adopt the loss function proposed in WGAN-GP (Gulrajani et al., 2017) to update the discriminator:

$$\begin{aligned} &\mathcal{L}_C = \mathbb{E}[D(\boldsymbol{p}_g)] - \mathbb{E}[D(\boldsymbol{p}_e)]; \\ &\mathcal{L}_P = \lambda \mathbb{E}[(\parallel \nabla_{\tilde{\boldsymbol{p}}} D(\tilde{\boldsymbol{p}}) \parallel_2 - 1)^2]; \\ &\mathcal{L}_D = \mathcal{L}_C + \mathcal{L}_P, \end{aligned} \tag{9}$$

where $\mathcal{L}_C$, $\mathcal{L}_P$ and $\mathcal{L}_D$ respectively denote the original critic loss, gradient penalty and the loss of discriminator, λ is the gradient penalty coefficient and $\tilde{\boldsymbol{p}}$ is sampled uniformly along straight lines between $\boldsymbol{p}_g$ and $\boldsymbol{p}_e$. According to the feedback of the discriminator, we calculate the reward $\mathcal{R}_G$ as:

$$\mathcal{R}_G = \delta_g \max\{\mathbb{E}[D(\boldsymbol{p}_g)] - \mathbb{E}[D(\boldsymbol{p}_n)], 0\}, \tag{10}$$

$$\delta_g = \begin{cases} 1, & \mathcal{P}_g \subseteq \Omega_G^+ \\ 0, & otherwise \end{cases}, \tag{11}$$

where $\boldsymbol{p}_n$ denotes a noise embedding composed of random noise with continuous uniform distribution, δ_g is a characteristic function which characterizes the validity of package $\mathcal{P}_g$, Ω_G^+ is the collection of all valid generated paths. We only give positive rewards for partial valid paths that at least have higher expectations than noise embedding $\boldsymbol{p}_n$, which filters out the paths of inferior quality to improve the convergence efficiency of the training process. Once the reward obtained, we updated the generator G by maximizing the expected cumulative reward with Monte-Carlo Policy Gradient (i.e., REINFORCE) (Williams, 1992).

We use mini-batch stochastic gradient descent (SGD) to optimize the loss function of discriminator, while the generator is updated with the Adam algorithm (Kingma and Ba, 2014).

5 Experiments

5.1 Datasets and Evaluation Metrics

Dataset	# Ent.	# Rel.	# Facts	# Tasks
NELL-995	75,492	200	154,213	12
FB15K-237	14,505	237	310,116	20

Table 1: Statistics of the datasets. # Ent. denotes the number of unique entities and # Rel. denotes the number of relations.

The experiments are conducted on two benchmark datasets: NELL-995 (Xiong et al., 2017) and FB15K-237 (Toutanova et al., 2015). The details of the two datasets are described in Table 1. In particular, NELL-995, which is known to be a simple dataset for reasoning tasks, is generated from the 995th iteration of the NELL system (Carlson et al., 2010) by selecting the triples with Top-200 frequently occurring relations. Compared to NELL-995, FB15K-237 is more challenging and closer to real-world scenarios, where its facts are created from FB15K (Bordes et al., 2013) with redundant relations removed. For each triple (h, r, t), both datasets contain the inverse triple (h, r^{-1}, t) such that the agent can step backward in KGs, which makes it possible to recover from a potentially wrong decision that has been taken before. For each reasoning task with a query relation r_q, all the triples with r_q or r_q^{-1} are removed from the KG and split into train and test samples.

Similar to recent works (Das et al., 2018; Xiong et al., 2017), we use mean average precision (MAP), mean reciprocal rank (MRR) and Hits@k to evaluate the reasoning performance, where Hits@k is the fraction of positive instances ranked in the top k positions.

5.2 Baselines and Implementation Details

In our experiments, we consider two state-of-the-art RL-based methods as baselines: DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018). Deep path feeds the gathered evidential paths to PRA (Lao et al., 2011) for both link prediction and fact prediction tasks, while MINERVA directly applies the well-trained agent to the link prediction task for question answering. For DeepPath, we use the code released by Xiong et al. (2017). For MINERVA, we use the code released by Das et al. (2018). The experiment settings for the baselines are set according to the suggestions in the original papers.

In the implementation of our framework, we set the path number N to 5 for each path package $\mathcal{P}$, while the path dimension k is set to 200 which is the same as the relation dimension in baselines. For the discriminator, we set the convolution kernel size to 3×5, the hidden layer size to 1024, and the output layer size to the path dimension k, while the gradient penalty coefficient λ is set to 5 and L2 regularization is also used to avoid over-fitting.

During testing, we also rank the answer triples

		Link Prediction				Fact Prediction	
Data	**Metric**	**DeepPath**	**Div(DeepPath)**	**MINERVA**	**Div(MINERVA)**	**DeepPath**	**Div(DeepPath)**
NELL-995	Hits@1	0.718	0.749	0.799/0.663†	0.840/0.668†	0.687	0.701
	Hits@3	0.847	0.878	0.936/0.773†	0.936/0.778†	0.839	0.841
	MRR	0.784	0.811	0.852/0.725†	0.874/0.731†	0.764	0.774
	MAP	0.796	0.821	0.885	0.893	0.774/0.493‡	0.785/0.494‡
FB15K-237	Hits@1	0.417	0.520	0.427/0.217†	0.444/0.223†	0.394	0.421
	Hits@3	0.646	0.740	0.582/0.329†	0.590/0.331†	0.565	0.599
	MRR	0.560	0.639	0.546/0.293†	0.556/0.296†	0.522	0.545
	MAP	0.572	0.658	0.553	0.568	0.536/0.311‡	0.558/0.339‡

Table 2: Overall results on NELL-995 and FB15K-237. "†" denotes the results with settings for question answering and "‡" denotes the results of directly ranking all the positive and negative triples given a query relation.

against the negative triples used in DeepPath and MINERVA. In particular, there are approximately 10 corresponding negative triples for each positive ones. Each negative triple is generated by replacing the answer entity t with a faked one t' given a positive triple (h, r, t).

5.3 Results

The main results on the two datasets are shown in Table 2. We use "Div(*)" to denote the RL-based method "*" which adopts our framework DIVINE. For a fair comparison, we follow MINERVA to report the Hits@k and MRR scores (denoted with "†") to evaluate the link prediction performance for question answering, which ranks entities according to the probability that the agent can reach the entity along evidential paths. Moreover, we also follow DeepPath to report the MAP scores (denoted with "‡") on the fact prediction task, which directly ranks all the positive and negative triples for a given query relation.

From the results shown in Table 2, we can observe that our framework produces consistent improvements for the two RL-based methods under varying degrees on both link prediction and fact prediction tasks. On the one hand, for existing RL-based methods, their results on FB15K-237 are generally lower than those on NELL-995 since FB15K-237 is more complex and arguably more difficult to design proper reward functions manually. However, our framework reliefs this problem to some extent by dynamically learning superior reward functions, thus we make greater improvements on challenging FB15K-237. On the other hand, for different datasets, the improvements our framework makes for DeepPath vary a lot while MINERVA not. This is because MINERVA manually adjusts its hyper-parameters accordingly when calculating cumulative discounted reward, while DeepPath keeps the same. Obviously, it validates the necessity for existing RL-based methods to adjust their reward functions accordingly to fit different datasets. Enhanced by our framework, these RL-based methods no longer require additional manual adjustments for different datasets, which reveals great robustness.

Similar to existing RL-based methods, we also report the decomposed results of link prediction and use MAP to evaluate the performance for each query relation on NELL-995 in Table 3. From the results, we can observe that the results on different relations are of high variance and the enhanced RL-based methods achieve better or comparable performance for all query relations.

5.4 Effectiveness of the Path-Based GAIL Method

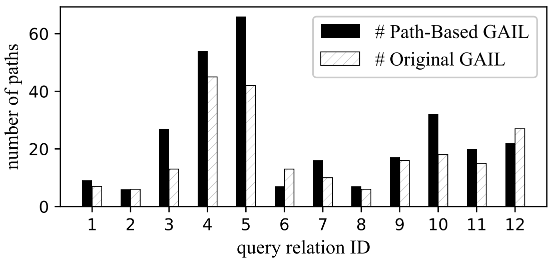

Figure 2: Statistics of path-finding on NELL-995.

		MAP	
	Path Avg.	# Link	# Fact
# Path-Based GAIL	24	0.821	0.494
# Original GAIL	18	0.814	0.468

Table 4: Results of different imitation learning settings.

To investigate the effectiveness of the path-based GAIL method, we train the policy-based agent in DeepPath on NELL-995 by our path-based GAIL method and the original GAIL method, respectively. In particular, in the process of training the a-

(ID) Query Relations	DeepPath	Div(DeepPath)	MINERVA	Div(MINERVA)
(01) AgentBelongsToOrg	0.576	**0.674**	0.860	**0.876**
(02) AthleteHomeStadium	**0.890**	0.877	0.895	**0.916**
(03) AthletePlaysForteam	0.750	**0.779**	**0.824**	0.813
(04) AthletePlaysInLeague	0.960	**0.962**	**0.970**	0.965
(05) AthletePlaysSport	0.957	**0.960**	0.985	**0.986**
(06) OrgHeadquarteredInCity	0.790	**0.801**	**0.946**	0.936
(07) OrgHiredPerson	0.742	**0.758**	0.851	**0.860**
(08) PersonBornInLocation	0.757	**0.780**	0.793	**0.827**
(09) PersonLeadsOrg	0.794	**0.810**	0.851	**0.881**
(10) TeampPlaysInLeague	0.881	**0.896**	**0.970**	0.957
(11) TeampPlaysSport	0.739	**0.816**	0.846	**0.874**
(12) WorksFor	0.711	**0.741**	**0.825**	0.822
Overall	0.796	**0.821**	0.885	**0.893**

Table 3: Results of link prediction decomposed over different query relations on NELL-995.

gent by the original GAIL method, the demonstrations are composed of state-action trajectories. For each state-action pair (s_t, a_t), the state representation $\boldsymbol{s}_t$ is calculated by $(\boldsymbol{e}_t, \boldsymbol{e}_{tail} - \boldsymbol{e}_t)$, where $\boldsymbol{e}_t$ and $\boldsymbol{e}_{tail}$ denote the embeddings of the current entity and the tail entity, respectively. In Figure 2, we show the statistics of the evidential path set P_{new} which are found by the agent and different from the demonstrations. In Table 4, we compare the average path number of P_{new} and the reasoning performance on the two prediction tasks.

As shown in Figure 2 and Table 4, we can observe that our path-based GAIL method obtains more evidential paths for most query relations and achieves better performance on both link and fact predictions, which validates the effectiveness of our path-based GAIL method and the rationality of encouraging the agent to find more diverse evidential paths.

5.5 Ablation Studies

We conduct ablation studies by embedding DeepPath into our framework to quantify the role of components. Specifically, we re-train our framework by ablating certain components:

- W/O Semantic Matching, where no semantic matching is performed on long-tail entities. Instead, we directly extract some paths from the remaining demonstration candidates $\mathcal{C}_E$ according to their occurrence frequency.
- W/O Dynamic Sampling, where no dynamic demo sampling is performed to incorporate the local environment of entities. In other words, we only adopt the static demo sampling method to obtain demonstrations.
- W/O Demo Sampling, where no demonstration is used for imitation learning, which degenerates to DeepPath.

	MAP	
Configuration	NELL-995	FB15K-237
Div(DeepPath)	0.821	0.658
W/O Semantic Matching	0.818	0.651
W/O Dynamic Sampling	0.806	0.640
W/O Demo Sampling	0.796	0.572

Table 5: Ablation on different components.

We use MAP to evaluate the link prediction performance on both NELL-995 and FB15K-237 in Table 5. From the results, we can observe that: (1) Based on imitation learning, our framework can effectively improve the reasoning performance, even if we use the static demo sampling method to obtain demonstrations; (2) High-quality demonstrations are crucial for imitation learning, which indicates both topology filtering and semantic matching play important roles in the demonstration sampler of our framework.

6 Conclusion

In this paper, we proposed a novel plug-and-play framework DIVINE for knowledge graph reasoning based on generative adversarial imitation learning, which enables existing RL-based methods to learn reasoning policies and reward functions self-adaptively to adapt the fast evolutions of real-world KGs. The experimental results show that our framework improves the performance of existing RL-based methods while eliminating extra reward engineering.

Acknowledgments

This work was supported by National Natural Science Foundation of China under Grant 61872045. We acknowledge anonymous reviewers for their valuable comments.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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 26*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1306–1313.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. pages 132–141.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-shot imitation learning. In *Advances in Neural Information Processing Systems 30*, pages 1087–1098.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 397–406.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5767–5777.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, volume 11, pages 809–816.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Stuart Russell. 1998. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1434–1444.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via a complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.