Collaborative Policy Learning for Open Knowledge Graph Reasoning

Cong Fu[1*], **Tong Chen**[2*], **Meng Qu**[3], **Woojeong Jin**[4], **Xiang Ren**[4]
[1]Zhejiang University, [2]Carnegie Mellon University, [3]University of Montreal
[4]University of Southern California
fc731097343@gmail.com, tongc2@andrew.cmu.edu, meng.qu@umontreal.ca
{woojeong.jin, xiangren}@usc.edu

Abstract

In recent years, there has been a surge of interests in interpretable graph reasoning methods. However, these models often suffer from limited performance when working on sparse and incomplete graphs, due to the lack of evidential paths that can reach target entities. Here we study *open knowledge graph reasoning*—a task that aims to reason for missing facts over a graph augmented by a background text corpus. A key challenge of the task is to filter out "*irrelevant*" facts extracted from corpus, in order to maintain an effective search space during path inference. We propose a novel reinforcement learning framework to train two collaborative agents jointly, i.e., a *multi-hop graph reasoner* and a *fact extractor*. The fact extraction agent generates fact triples from corpora to enrich the graph on the fly; while the reasoning agent provides feedback to the fact extractor and guides it towards promoting facts that are helpful for the interpretable reasoning. Experiments on two public datasets demonstrate the effectiveness of the proposed approach. Source code and datasets used in this paper can be downloaded at https://github.com/shanzhenren/CPL.

1 Introduction

Knowledge graph completion or reasoning—i.e., the task of inferring the missing facts (entity relationships) for a given graph—is an important problem in natural language processing and has a wide range of applications (Bordes et al., 2011; Socher et al., 2013; Trouillon et al., 2016). Recent neural graph reasoning methods, such as MINERVA (Das et al., 2017), DeepPath (Xiong et al., 2017) and Multi-Hop (Lin et al., 2018), have achieved impressive results on the task, offering both good *prediction accuracy* (compared to embedding-based methods (Trouillon et al., 2016; Dettmers et al., 2018)) and *interpretability* of the model predictions. These reasoning methods frame the link inference task as a path finding problem over the graph (see Fig. 1 for example).

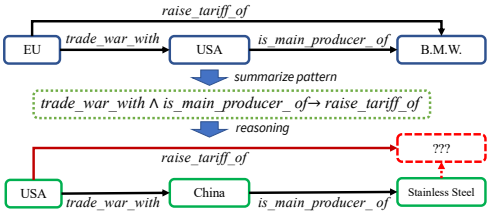

Figure 1: **Illustration of the Knowledge Graph Reasoning Task.** Given an entity (e.g., *Miami*) and a query relation (e.g., *located in*), we learn to infer reasoning paths over the existing graph structure to help predict the answer entity (i.e., *USA*).

However, current neural graph reasoning methods encounter two main challenges as follows: (1) their performance are often sensitive to the sparsity and completeness of the graph—missing edges (i.e., potential false positives) make it harder to find evidential paths reaching target entities. (2) existing models assume the graph is static, and cannot adapt to dynamically enriched graphs where emerging new facts are constantly added.

In this paper, we study the new task of Open Knowledge Graph Reasoning (OKGR), where the new facts extracted from the text corpora will be used to augment the graph dynamically while performing reasoning (as illustrated in Figure 2). All the recent joint graph and text embedding methods focus on learning better knowledge graph embeddings for reasoning (Xu et al., 2014; Han et al., 2018), but we consider adding more facts to the graph from the text to improve the reasoning performance and further provide interpretability. A straightforward solution for the OKGR problem is to directly add extracted facts (by a pre-trained relation extraction model) to the graph. However, most facts so extracted may be noisy or irrelevant

[*] Work done while the authors interned at USC.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2672–2681, Hong Kong, China, November 3–7, 2019.

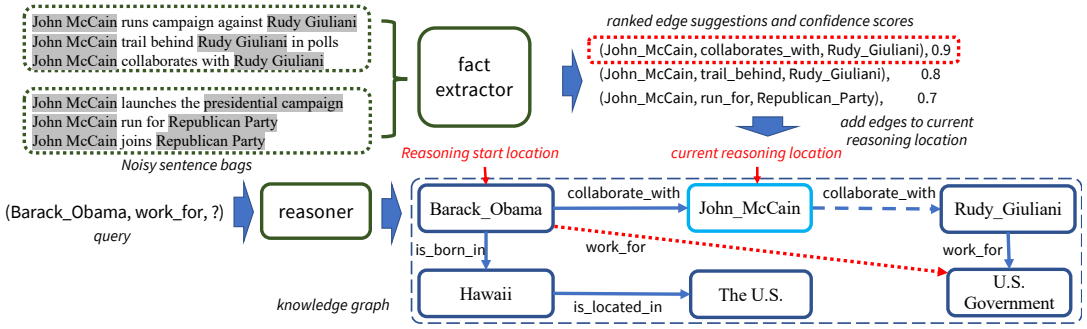

Figure 2: **Overview of our CPL framework for the OKGR problem.** To augment the reasoning with the information from a background corpus, CPL extracts relevant facts (e.g., the triple linked by the blue dotted arrow) to augment the KG dynamically. CPL involves two agents: one learns fact extraction policy to suggest relevant facts; the other learns to reason on dynamically augmented graphs to make predictions (e.g., the red dotted arrow).

to the path inference process. Moreover, adding a large number of edges to the graph will create an ineffective search space and cause scalability issues to the path finding models. Therefore, it is desirable to design a method that can filter out irrelevant facts for augmenting the reasoning model.

To address the above challenges for OKGR, we propose a novel *collaborative policy learning* (CPL) framework to jointly train two RL agents in a mutually enhancing manner. In CPL, besides training a *reasoning* agent for path finding, we further introduce a *fact extraction* agent, which learns the policy to select relevant facts extracted from the corpus, based on the context of the reasoning process and the corpus (see Fig. 2). At inference time, the fact extraction agent dynamically augments the graph with only the most informative edges, and thus enables the reasoning agent to identify positive paths effectively and efficiently.

Specifically, during policy learning, the reasoning agent will be rewarded when reaching the targets, while this positive feedback will also be transferred back to the fact extracting agent if its edge suggestions are adopted by the reasoning agent, i.e. making up correct reasoning paths. This ensures that the fact extraction policy can be learned in a way that edges which are beneficial to path inference will be preferred. By doing so, the fact extraction agent can learn to augment knowledge graphs dynamically to facilitate the reasoning agent, while the reasoning agent performs effective path-inference and provides reward signals to the fact extraction agent. Please refer to Sec. 3 and Fig. 3 for more implementation details.

The major contributions of our work are as follows: (1) We study knowledge graph reasoning in an "open-world" setting, where new facts extracted from background corpora can be used to facilitate path finding; (2) We propose a novel collaborative policy learning framework which models the interactions between fact extraction and graph reasoning; (3) Extensive experiments and analysis are conducted to demonstrate the effectiveness and strengths of our proposed method.

2 Background and Problem

This section introduces basic concepts and notations related to the knowledge graph reasoning task and provides a formal problem definition.

A *knowledge graph* (KG) can be represented by a set of triples (facts) $G = \{(e_s, r, e_o)|e_s, e_o \in E, r \in R\}$, where E is the set of entities and R is the set of relations. e_s, r, and e_o are the subject entity, relation, and object entity respectively.

The task of *knowledge graph reasoning* (KGR) is defined as follows. Given the KG G, a query triple (e_s, r_q, e_q) where e_q is unknown, KGR is to infer e_q through finding a path starting from e_s to e_q on G, namely $\{(e_s, r_1, e_1), ..., (e_n, r_n, e_q)\}$. Usually, KGR methods produce multiple candidate answers by ranking the found paths, while traditional KG completion methods rank all possible answer triples by exhaustively enumerating.

A *background corpus* is a set of sentences labeled with respective entity pairs, namely $C = \{(s_i : (e_k, e_j))|s_i \in S, e_k, e_j \in E\}$, where S is the set of sentences, and the corpus shares the same entity set with G. In our problem setting, we assume the entities have already been extracted; thus, extracting facts from the corpus is equivalent to the relation extraction task. We process the corpus by labeling the sentences with subject and object entity pairs through Distant Supervision (Mintz et al., 2009). There may be many sen-

tences labeled with the same entity pair. Following the formulation of previous work (Lin et al., 2016), we organize the sentences into *sentence bags*, i.e., A sentence bag contains the sentences which are labeled with the same entity pair.

Problem. Formally, *Open Knowledge Graph Reasoning* (**OKGR**) aims to perform KGR based on both G and C, where G is dynamically enriched by the facts extracted from C. This paper focuses on OKGR, i.e., empowering KGR with the corpus information and enriching the graph with relevant facts dynamically. Thus, the evaluation of the relation extraction performance are out of the scope of this paper. We leave this as future work.

3 Proposed Framework

Overview. To resolve the challenges in OKGR, we propose a novel collaborative policy learning (CPL) framework (see Fig. 2), which jointly train two RL agents, i.e., a path reasoning agent and a fact extraction agent. Given a query (e_s, r_q, e_q), the reasoning agent tries to infer e_q via finding a reasoning path on the (augmented) G, while the fact extraction agent aims to select the most informative facts from C to enrich G dynamically. With such an extractor, the framework can effectively overcome the edge sparsity problem while remaining reasonably efficient (compared to the naive solution that adds all possible facts to G). We train the extraction agent by rewarding it according to the the reasoning agent's performance. Hence the fact extractor can learn how to extract the most informative facts to benefit the reasoning.

3.1 Graph Reasoning Agent

The goal of the reasoner is learning to reason via paths finding on KGs. Specifically, given e_s, r_q, the reasoner aims at inferring a path from e_s to some entity e_o regarding r_q, and specifying how likely the relationship r_q holds between e_s and e_o. The inference path acts as the evidence of the prediction, and thus offers interpretation (see Fig. 1). At each time step, the reasoner tries to select an edge based on the observed information. The Markov Decision Process (MDP) of the reasoner is defined as follows:

State. In path-based reasoning, each succeeding edge is closely related to the preceding edge on the path and the query in semantics. Similar to MINERVA(Das et al., 2017), we want the state to encode all observed information, i.e., we define $s_R^t = (e_s, r_q, h^t) \in \mathcal{S}_R$, where h^t encodes the path history, and (e_s, r_q) is the context shared among all states. Specifically, we use a LSTM module to encode the history, $h^t = \text{LSTM}(h^{t-1}, [r^t, e^t])$ (see Fig. 3). e^t is the current reasoning location and r^t is the previous relation connecting e^t.

Action. At time t, the reasoner will select an edge among e^t's out-edges. The reasoner's action space is a union of the edges in the current KG and edges extracted from the corpus. See Sec. 3.3 for details.

Transition. The transition function $f : \mathcal{S}_R \times \mathcal{A}_R \rightarrow \mathcal{S}_R$ is defined as $f(s_R^t, a_R^t) = (e^s, r^q, h^{t+1})$ naturally.

Reward. The reasoner is expected to learn effective reasoning path patterns. We let it explore for a fixed number of steps, which is a hyper-parameter. Only when it reaches the correct target entity, it receives a terminal reward 1, and 0 otherwise. All intermediate states always receive reward 0.

3.2 Fact Extraction Agent

The fact extractor learns to suggest the most relevant facts w.r.t. the current inference step of the reasoner. Suppose the reasoner arrives at entity e^t on the graph at time t, the fact extractor will extract facts in the form of $(e^t, r', e') \notin G$ from the corpus and add them to the graph temporarily. Consequently, the reasoner is offered more choices to expand the reasoning path.

State. When the reasoner is at e^t, the fact extractor tries to extract information from the corpus (sentences) and suggests promising out-edges of entity e^t. Let b_{e^t} denote the sentence bags labeled with $(e^t, e'), e' \in E$. We define the state of the fact extractor to encode the current observed information, i.e., $s_E^t = (b_{e^t}, e^t) \in \mathcal{S}_E$, where $\mathcal{S}_E$ is the whole state space, containing all possible combinations of entities and corresponding sentence bags.

Action. The goal of the fact extractor is to select a reasoning-relevant fact contained in the corpus semantically. At step t, the reasoner will move to a new entity from e^t, and hence only the out-edges of e^t should be considered, e.g., (e^t, r', e') (see Fig. 3). Therefore, for each fact (e^t, r', e') which can be extracted from the sentence bag b_{e^t}, we can derive an action $a_E^t = (r', e')$, and the action space at step t can be denoted as $A_E^t = \{(r', e')\}_{(e^t, r', e') \in b_{e^t}} \subset \mathcal{A}_E$. $\mathcal{A}_E$ is the whole action space containing all facts in the corpus.

Transition. The transition function $f : \mathcal{S}_E \times \mathcal{A}_E \rightarrow \mathcal{S}_E$ is defined as $f(s_E^t, a_E^t) = (r', e')$.

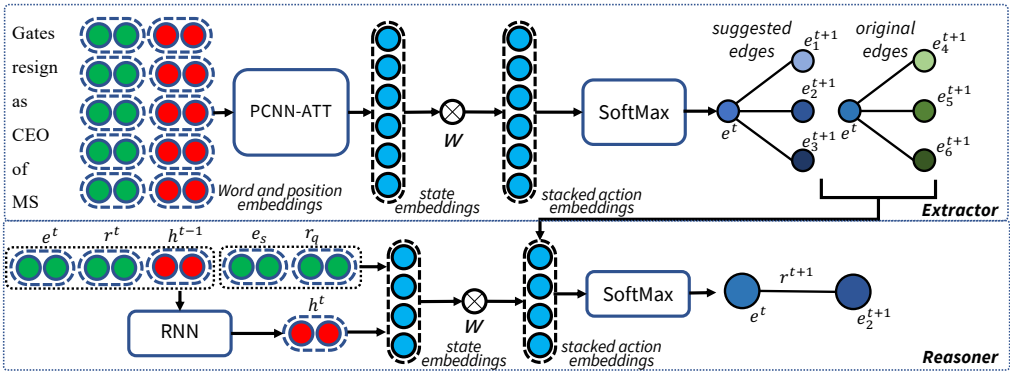

Figure 3: **Detailed Model Design of Collaborative Policy Learning (CPL).** Take PCNN-ATT as an example of the sentence encoder. The figure shows how it works at a certain inference time step t, the reasoner is at entity e^t and will select one edge from the *joint action space*, which consists of new edges extracted by the extractor and the edges in the original graph.

Reward. The fact extractor receives a step-wise delayed reward from the reasoner according to how it improves the reasoners performance. The extractor will be positively rewarded when its suggestion benefits the reasoning process. Please see Sec. 3.3 for details.

3.3 Collaborative Policy Learning

In this section, we will introduce the detailed training process and the collaboration mechanism between the two agents. At the high level, we adopt an *alternative training procedure* to update the two agents jointly: training one of the agents for a few iterations while freezing the other; and vice versa. The policies of both agents are updated via REINFORCE algorithm (Williams, 1992) (details are in later of this section). Specifically, we introduce the details on agent collaboration as follows.

Augmented Action Space for Reasoning. At time t, the fact extractor helps the reasoner via expanding its action space with new edges extracted from the corpus (see Fig. 3). Due to the sparsity and incompleteness of the KG, there may be missing edges preventing the reasoner from inferring the correct reasoning path (Fig. 2). Therefore, we add high-confidence edges extracted by the extractor to the action space of the reasoner. Formally, at time t, the reasoner is at location e^t (Fig. 3) and tries to select an edge out of all out-edges of e^t. Let A_K^t denote the edge set in the current KG, $A_K^t = \{(r, e)|(e^t, r, e) \in G\}$. Let A_C^t denote the edge set suggested by the extractor, $A_C^t = \{(r', e')|(e^t, r', e') \in C\}$. The action space at time t of the reasoner is defined as $A_R^t = A_K^t \cup A_C^t$, $A_R^t \subset \mathcal{A}_R$, where $\mathcal{A}_R$ denotes the whole action space of the reasoner, i.e., all possible edges in the KG and the corpus. The reasoner learns a policy to select the best edges out of the joint action space for reasoning.

Reasoning Feedback for Fact Extraction. The reasoner helps the extractor to learn the extracting policy through providing feedbacks regarding how much the extractor contributes to the reasoning. Therefore we define that the fact extractor receives a step-wise delayed reward from the reasoner. Specifically, when the reasoner finishes exploration (at time T) and arrives at the correct target, we consider the path is effective and positive for reasoning. If the fact extractor contributes to this positive path, it can be rewarded positively, i.e., if an edge on the positive path is suggested by the extractor at time $t, 0 \leq t \leq T$, the extractor will be rewarded 1 at time t, and 0 otherwise. Extracted edges triggering positive rewards will be kept in the graph, while the others will be removed when both agents move to the next state.

Policy Update. The MDPs of both agents are explicitly defined above now, and we can use the typical REINFORCE algorithm (Williams, 1992) to train the two agents. Specifically, their goals are maximizing the reward expectation, defined as

$$J(\theta) = \mathbb{E}_{\pi_\theta(a|s)}[R(s, a)], \tag{1}$$

where $R(s, a)$ is the reward of selecting a given s, and $\pi_\theta(a|s)$ is the policy learned by the agents and will be defined formally in Section. 4.

Given a training sequence sampled from π_θ: $\{(s^1, a^1, r^1), ..., (s^T, a^T, r^T)\}, r^t = R(s^t, a^t)$, at time step t, the parameters are updated according to the REINFORCE algorithm:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a^t|s^t) G^t$$
$$G^t = \sum_{k=t}^{T} \gamma^{k-t} R(s^k, a^k), \quad (2)$$

where G^t is the discounted accumulated reward.

According to Eq. (2), we can see that REINFORCE will update the parameters only when G^t is non-zero. In other words, the value of γ determines how the parameters are updated and to what extent the the internal states will be influenced by the future. If $\gamma > 0$, for positive training sequences, it is easy to verify that the G^t of all states will be non-zero. Thus, the internal states will be positively rewarded, and the model parameters will be updated by the gradients of the internal states. For different task, we should carefully select the value of γ. For the extractor, we set $\gamma = 0$ for the extractor to avoid policy updating on zero-rewarded state-action experiences. This is because zero-rewarded experiences are mostly negative examples. Specifically, if a state of the extractor is zero-rewarded, we can infer that either the suggested edge is not selected by the reasoner, or the selected edge does not contribute to reaching the target. We can not allow the model to be updated on such experiences, so we set $\gamma = 0$ to avoid the influence of future. In contrast, we set $\gamma = 1$ for the reasoner because all the intermediate selected edges are meaningful as long as it leads to the target finally.

4 Model Implementation

In this section, we introduce the policy network architectures (cf. Fig. 3) of the two agents and provide details on model training and inference.

4.1 Policy Network Architectures

Reasoning Agent. We construct the state embedding by concatenating all related embeddings, $s_R^t = [e_s, r_q, h^t]$, where $h^t = \text{LSTM}(h^{t-1}, [r^t, e^t])$. We construct the action embedding by concatenating the relation-entity embedding pair, i.e., $a_R^t = [r, e], (e^t, r, e) \in G \cup C$. We stack all action embeddings in $\mathbf{A}_R^t$. The policy network is defined as:

$$\pi_\theta(a_R^t|s_R^t) = \sigma(\mathbf{A}_R^t W_2(Relu(W_1[e_s, r_q, h^t]))),$$

σ is softmax, and W_1, W_2 are learnable weights.

Fact extraction Agent. We use a PCNN-ATT as the sentence encoder in our experiments to construct the distributed representations for the sentences. Let b_{e^t} denote all the sentence bags labeled by $(e^t, e'), e' \in E$. At time t, we input b_{e^t} into the PCNN-ATT to obtain the sentence-bag-level embeddings E_b^t, which is regarded as the latent state embeddings. As mentioned in Sec. 2, the object entity has been labeled beforehand. We need to select the best relation first, then select the best entity under this relation. Thus, we stack the relation embeddings in A_E^t as $\mathbf{A}_E^t \in \mathbb{R}^{|A_E^t| \times d}$, where d is the dimension of relation embedding. The policy network is defined formally as:

$$\pi_\theta(a_E^t|s_E^t) = \sigma(\mathbf{A}_E^t W E_b^t),$$

where W is a learnable weight.

The extractor will predict the scores for each sentence bag regarding the relation. We will select the sentence bag with the highest score, more formally, the corresponding relation-entity pair in that sentence bag will be chosen as the next action. We train the agents as introduced in Sec. 3.3.

4.2 Model Training and Inference

Training. We use *model pre-training* and *adaptive sampling* to increase training efficiency. In particular, we first train the reasoner on the original KG to get a better initialization. Similarly, we train the extractor on the corpus labeled by distant supervision. Next, we use adaptive sampling to adaptively increase the selecting-priority of corpus-extracted-edges when generating training experiences for the two agents. Adaptive sampling is designed to encourage the reasoner to explore more on new edges and facilitate the collaboration during the joint training. Replay memories (Mnih et al., 2013) are also used to increase training efficiency. We develop several model variants such as removing adaptive sampling or replay memory, or freezing the extractor all the time to conduct ablation studies. Please see Supplementary Material for more details.

Inference. At inference (reasoning) time, we use the trained model to predict missing facts via path finding. The process is similar to the training experience generation step in the training stage, i.e. using the reasoner for path-inference while the extractor suggests edges from the corpus constantly. The only differences are that we do not request rewards, and we use beam search to generate multiple reasoning paths over the graph, and rank them by the scores from the reasoner.

Dataset	#triples(C)	#triple(G)	#entities(C)	#entities(G)	#rel(C)	#rel(G)	S(train)	S(test)	CT/CE	CR/KR
FB60K-NYT10	172,448	268,280	63,696	69,514	57	1,327	570k	172k	2.71	0.04
UMLS-PubMed	910,320	2,030,841	6,575	59,226	271	443	4,73M	910k	138.45	0.61

Table 1: **The dataset information.** #triples(C) & #triples(G) denote the number of triples in the corpus and the KG respectively, and so on. S(train) denotes the number of sentences in the training corpus, while S(test) denotes the number of sentences in the testing corpus. CT/CE denotes triple-entity ratio. Lower triple-entity ratio indicates less triples per entity in average can be extracted from the corpus. CR/KR denotes corpus-relation-quantity/KG-relation-quantity ratio. Lower CR/KR indicates less information overlap between the corpus and the KG.

5 Experiment

5.1 Datasets and Compared Methods

Dataset	Relations
UP	*'gene_associated_with_disease'* *'disease_has_associated_gene'* *'gene_mapped_to_disease'* *'disease_mapped_to_gene'* *'may_be_treated_by'* *'may_treat'* *'may_be_prevented_by'* *'may_prevent'*
FN	*'people/person/nationality'* *'location/location/contains'* *'people/person/place_lived'* *'people/person/place_of_birth'* *'people/deceased_person/place_of_death'* *'people/person/ethnicity'* *'people/ethnicity/people'* *'business/person/company'* *'people/person/religion'* *'location/neighborhood/neighborhood_of'* *'business/company/founders'* *'people/person/children'* *'location/administrative_division/country'* *'location/country/administrative_divisions'* *'business/company/place_founded'* *'location/us_county/county_seat'*

Table 2: **The concerned relations in two datasets.** UP means the UMLS-PubMed dataset, while FB means the FB60K-NYT10 dataset.

Datasets. We construct two datasets for evaluation: *FB60K-NYT10*[1] and *UMLS-PubMed*[2]. FB60K-NYT10 dataset includes the FB-60K KG and the NYT10 corpus; The UMLS dataset contains the UMLS KG and the PubMed corpus. Statistics of both datasets are summarized in Table 1. We study the datasets and find that the relation distributions of the two datasets are very imbalanced. There are not enough reasoning paths for some relation types. Moreover, some relations are meaningless and of no reasoning value. Thus, we select a few meaningful and valuable relations (suggested by domain experts, in Table 2) with enough reasoning paths and construct two subgraphs accordingly. To show the impact of graph size, we sub-sample the KG into different subgraph. Specifically, for the two datasets, we first partition the whole KG into three parts according to the proportion 8:1:1 (The training set, validation set, and testing set). Next we create sub-train-sets with different ratio via random sampling.

Analysis of corpus-KG alignment. We analyze the information overlap (i.e., alignment) between the corpus and the KG in Table 1. The CT/CE (the ratio of triple quantity against entity quantity) of PubMed is far higher than NYT10. Higher CT/CE indicates adding corpus-edges to the KG increases the average degree more significantly, leading to more reduction in sparsity. The low CR/KR ratio of FB60K-NYT10 indicates the overlap between FB60K and NYT10 is lower than that between UMLS and PubMed. We can conclude that the alignment level of FB60K-NYT10 is lower than UMLS-PubMed. Intuitively, FB60K-NYT10 is a more difficult dataset than UMLS-PubMed.

Compared Algorithms. We compare our algorithm with (1) SOTA methods for KG embedding ; (2) methods for joint text and graph embedding; and (3) neural graph reasoning methods.

For triple-ranking-based KG embedding methods, we evaluate **DistMult** (Yang et al., 2014), **ComplEx** (Trouillon et al., 2016), and **ConvE** (Dettmers et al., 2018). For joint text and graph embedding methods, we evaluate **RC-Net** (Xu et al., 2014) and **Joint-NRE** (Han et al., 2018). We also construct a baseline, **TransE+LINE**, by constructing a word-entity co-occurrence network as RC-Net does. We use LINE (Tang et al., 2015) and TransE (Bordes et al., 2011) to jointly learn the entity and relation embeddings to preserve the structure information within the co-occurrence network and the KG. For neural graph reasoning method, we use **MINERVA** (Das et al., 2017), a reinforcement learning based path reasoning method[4].

[1] https://github.com/thunlp/OpenNRE

[2] http://umlsks.nlm.nih.gov/ https://www.ncbi.nlm.nih.gov/pubmed/

[4] There are other SOTA path-based knowledge reasoning methods such as Multi-Hop (Lin et al., 2018) and DeepPath (Xiong et al., 2017) However, DeepPath needs extra path-level supervisions which we do not have and Multi-Hop suf-

Model / Dataset	20%		40%		70%		100%	
	Hits@5	Hits@10	Hits@5	Hits@10	Hits@5	Hits@10	Hits@5	Hits@10
TransE (Bordes et al., 2011)	7.12	11.17	26.86	38.08	31.32	43.58	32.28	45.52
DisMult (Yang et al., 2014)	14.66	21.16	26.90	38.35	31.65	44.98	32.80	47.50
ComplEx (Trouillon et al., 2016)	18.58	18.18	23.77	34.15	30.04	43.60	31.84	46.57
ConvE (Dettmers et al., 2018)	20.51	30.11	**28.01**	**42.04**	31.01	45.81	30.35	45.35
RotatE (Sun et al., 2019)	4.03	6.50	8.65	13.21	14.90	21.67	20.75	27.82
RC-Net (Xu et al., 2014)	7.94	10.77	7.56	11.43	8.31	11.81	9.26	12.00
TransE+LINE	**23.63**	**31.85**	24.86	38.58	25.43	34.88	22.31	33.65
JointNRE (Han et al., 2018)	21.05	31.37	27.96	40.10	30.87	44.47	-	-
MINERVA (Das et al., 2017)	11.55	19.87	24.65	35.71	35.8	46.26	57.63	63.83
Two-Step	8.37	13.5	22.75	32.79	33.14	43.35	55.59	63.49
CPL (our method)	15.32	24.22	26.96	38.03	**37.23**	**47.60**	**58.10**	**65.16**

Table 3: **Performance comparison on the KG reasoning on the UMLS-PubMed dataset.**[3] We test on different graph sizes (i.e., 20-100% of the original graph) using Hits@K (in %). CPL is the best performing graph reasoning method, and gradually outperforms the others when the graphs are denser.

To validate the effectiveness of fact extraction policy in CPL, we design a two-step baseline (i.e., **Two-Step**). It first uses PCNN-ATT to extract relational triples from the corpora, and augments KG with the triples whose prediction confidences are greater than a threshold. PCNN-ATT (Lin et al., 2016) is a fact extraction model, which completes the fact extraction part. We tune the threshold on the dev-set. Then, a MINERVA model is trained on the augmented KG for reasoning.

CPL is our full-fledged model as introduced in Sec. 3. For all the methods, we upload the source codes and list hyper-parameters we used in the supplemental materials.

5.2 Evaluation and Experimental Setup

Following previous work on KG completion (Bordes et al., 2011), we use Hits@K and mean reciprocal rank (MRR) to evaluate the effectiveness of the KGR and OKGR. Given a query $(e_s, r, ?)$ (for each triple in the test set, we pretend not to know the object entity), we rank the correct entity e_q among a list of candidates entities. Suppose $rank_i$ is the rank of the correct answer entity for the i^{th} query. We define $Hit@K = \sum_i \mathbf{1}(rank_i < K)/N$ and $MRR = \frac{1}{N}\sum_i \frac{1}{rank_i}$.

In our experiments, we use a held-out validation set for all compared methods to search for the best hyper-parameters and the best model for testing (via logging checkpoints). For all methods, we train models using three fixed random seeds (55, 83, 5583), and report the metrics in average. More details on model training can be found in the Supplementary Material.

fers from out-of-memory problems on large-scale datasets.

5.3 Performance Comparison

Performances of the KG reasoning of all the algorithms are given in Table 3, 4 and Figure 6. We can draw conclusions as follows:

1. Triple ranking vs. path inference. CPL and MINERVA perform worse than triple-ranking methods when the size of KGs is small, while outperforms them significantly when adding more triples to the KGs (Figure 6). This is because the general and evidential paths for reasoning on sparse KGs are not enough, and path-based models cannot capture the underlying patterns.

2. CPL vs. joint embedding methods. CPL is inferior to RC-net, TransE+Line, and JointNRE on small KG partitions because they are not path-based models and the connections on small KGs are too sparse. CPL outperforms them significantly on larger datasets. The reasons are two-fold : 1) the graphs are denser to provide enough reasoning paths for training; 2) other algorithms do not filter noisy text information in joint-training.

3. CPL vs. other graph reasoning methods. CPL outperforms MINERVA significantly because CPL makes use of relevant text information for prediction. MINERVA is better than CPL on full FB60K-NYT10 because the alignment between FB60K and NYT10 is very limited (Sec. 5.1). The graph is dense at 100%, and the benefits from the corpus information are indiscernible.

5.4 Performance Analysis

1. Ablation Study on Model Components. In CPL, we apply multiple learning techniques to improve the performance, including collaboration, replay memory, and adaptive sampling as introduced in Sec. 4. To show the effects of differ-

Model / Dataset	20%			50%			100%		
	Hits@5	Hits@10	MRR	Hits@5	Hits@10	MRR	Hits@5	Hits@10	MRR
TransE (Bordes et al., 2011)	15.12	18.83	12.57	19.38	23.2	13.36	38.53	43.38	29.90
DisMult (Yang et al., 2014)	1.42	2.55	1.05	15.23	19.05	12.36	32.11	35.88	24.95
ComplEx (Trouillon et al., 2016)	4.22	5.97	3.44	19.10	23.08	12.99	32.91	34.62	24.67
ConvE (Dettmers et al., 2018)	**20.6**	**26.9**	11.96	24.39	30.59	18.51	33.02	39.78	24.45
RotatE (Sun et al., 2019)	9.25	11.83	8.04	25.96	31.63	23.34	**58.32**	**60.66**	**51.85**
RC-Net (Xu et al., 2014)	13.48	15.37	**13.26**	14.87	16.54	14.63	14.69	16.34	14.41
TransE+Line	12.17	15.16	4.88	21.7	25.75	8.81	26.76	31.65	10.97
JointNRE (Han et al., 2018)	16.93	20.74	11.39	26.96	31.54	21.24	42.02	47.33	32.68
MINERVA (Das et al., 2017)	11.64	14.16	8.93	25.16	31.54	22.24	43.80	44.70	34.62
Two-Step	12.14	16.5	9.27	21.66	31.50	19.82	39.22	44.64	34.18
CPL (our method)	15.19	18.00	10.87	**26.81**	**31.7**	**23.80**	43.25	49.50	33.52

Table 4: **Performance comparison on the KG reasoning on the FB60K-NYT10 dataset.** We can observe similar performance trends as those on the UMLS-PubMed dataset.

ent components, we remove them one by one and train the respective model variants. In addition, to shown the effect of collaboration, we train a model variation with the parameters of the extractor frozen. The result is shown in Fig. 5.

From the result, we find that 1) replay memory is only effective when adaptive sampling is also enabled. This is because adaptive sampling solves the sparse positive sample problem to some extent. There are enough positive experiences for replay. 2) Collaboration improves performance significantly. CPL with a trainable extractor performs better than with a frozen extractor, which means the suggestions of the extractor can be improved by the reasoner's feedbacks. 3) The improvement of CPL over MINERVA reduces as we increase the KG size. This is because with more data for training, the graph becomes denser, and hence the contribution from texts will be diluted.

2. Effectiveness of Fact Selection. As mentioned above, Two-Step is the naive solution to OKGR. The best performing Two-Step model adds tens times more edges into the KG than CPL, whereas the Two-Step model's performance is inferior to CPL and MINERVA on all the datasets (Table 3, 4). The reasons are 1) most of the extracted edges used in the Two-Step model are noisy; 2) adding so many edges significantly enlarges the exploration space for reasoning.

2. We perform a case study on the FB60K-NYT10 dataset to show the effectiveness of dynamically fact-filtering. We check the reasoning performance of the MINERVA and CPL periodically during the training. The results show that the extractor's contribution increases along with the training progress and the adaptive sampling can generate sufficient positive training experiences at the very beginning.

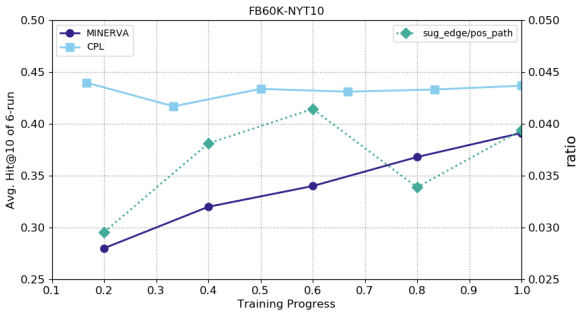

Figure 4: **KG reasoning performance change w.r.t. time.** sug_edge/pos_path means the ratio of positive edges suggested by the extractor w.r.t. the positive paths found by the reasoner.

The result is shown in Figure 4. We find a few interesting points as follows: **1)** the sug_edge/pos_path ratio curve in Figure 4 suggests that the extractor's contribution increases along with the training progress; **2)** CPL has a high initial performance because the adaptive sampling generates sufficient positive training experiences quickly. **3)** The valley shape in the performance curve is because the agent has not learned a stable exploring policy when the adaptive sampling stops, and the adaptive sampling somehow twisted the true pattern distribution in the dataset. But with a good start, the agent can explore on its own to approach the true distribution.

5.5 Case Study of Reasoning Paths

We randomly sample some reasoning paths from the inference results of CPL as examples. Due to the space limit, please refer to the supplemental materials for these examples. These examples show 1) how the reasoner finds the path patterns for the respective relations; 2) how the reasoner finds the inference paths according to the patterns; 3) how the extractor suggests rel-

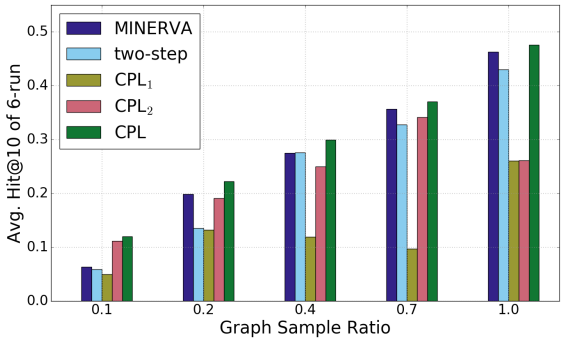

Figure 5: **Ablation Study on UMLS-PubMed dataset.** CPL_1 denotes CPL without adaptive sampling, and the extractor is frozen during training. CPL_2 denotes CPL without adaptive sampling. CPL denotes our proposed final model (with all the components).

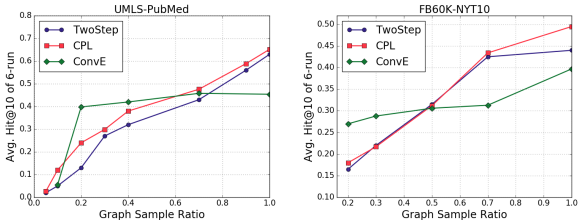

Figure 6: **KG reasoning performance change w.r.t. the size of the graph.** Triple-ranking based methods perform pretty well on smaller partitions, but are soon surpassed by path-based KG reasoning methods with the increase of graph size.

evant edges for each positive paths; 4) how the extractor extracts the relevant facts from related sentences. In summary, these cases show how CPL performs interpretable knowledge graph reasoning (infer the query entity through semantics-related path searching) and how CPL performs interpretable fact-filtering (suggest edges w.r.t the learned reasoning path patterns).

6 Related Work

Knowledge Graph Reasoning. Diverse approaches of embedding-based KG reasoning are presented, including linear models (Bordes et al., 2011), latent factor models (Trouillon et al., 2016), matrix factorization models (Yang et al., 2014) and convolutional neural networks (Dettmers et al., 2018). Performances of these methods are promising, but their predictions are barely interpretable. RL plays a crucial role in interpretable KG reasoning. MINERVA (Das et al., 2017) and DeepPath (Xiong et al., 2017) employ policy networks; (Xiong et al., 2017) uses extra rule-based supervision. Multi-hop (Lin et al., 2018) improves MINERVA via reward shaping and action drop-out.

Joint Embedding of Text and KG. Joint embedding methods aim to unite text corpus and KG. Contrary to our focus, they mainly utilize KGs for better performances of other tasks. (Toutanova et al., 2015) focuses on fact-extraction on the corpus labeled via dependency parsing with the aids of KG and word embeddings, while (Han et al., 2016) conducts the same task with the raw corpus text. As a newer joint model developed from (Han et al., 2016), (Han et al., 2018) deals with fact extraction by employing the mutual attention.

Open-World KG Completion. There are works focusing on similar topics as ours. (Shi and Weninger, 2018) defines an Open World KG Completion problem, in which they complete the KG with unseen entities. (Friedman and Broeck, 2019) introduces the Open-World Probabilistic Databases, an analogy to KGs. Unlike our setting, they try to complete the KG with logical inferences without extra information. (Sun et al., 2018) proposes an open, incomplete KB environment (or KG) with text corpora, but they focus on extracting answers from question-specific subgraphs.

7 Conclusion

In this paper, we focus on a new task named as Open Knowledge Graph Reasoning, which aims at boosting the knowledge graph reasoning with new knowledge extracted from the background corpus. We propose a novel and general framework, namely Collaborative Policy Learning, for this task. CPL trains two collaborative agents, the reasoner and fact extractor, which learns the path-reasoning policy and relevant-fact-extraction policy respectively. CPL can perform efficient interpretable reasoning on the KG and filtering of noisy facts. Experiments on two large real-world datasets demonstrate the strengths of CPL. Our work can cope with different path-finding modules such as MultHop with reward shaping by ConvE or RotatE and thus can improve its performance as the module improves.

8 Acknowledgement

This work has been supported in part by National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift, JP Morgan AI Research Award, and China Scholarship Council. We would like to thank all the collaborators for their constructive feedbacks.

References

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tal Friedman and Guy Van den Broeck. 2019. On constrained open-world probabilistic databases. *arXiv preprint arXiv:1902.10677*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2016. Joint representation learning of text and knowledge for knowledge graph completion. *arXiv preprint arXiv:1611.04125*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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*, pages 926–934.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

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.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

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. *arXiv preprint arXiv:1707.06690*.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1219–1228. ACM.

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*.