# Multi-Hop Knowledge Graph Reasoning with Reward Shaping

**Xi Victoria Lin** **Richard Socher** **Caiming Xiong**
Salesforce Research
{xilin,rsocher,cmxiong}@salesforce.com

arXiv:1808.10568v1 [cs.AI] 31 Aug 2018

## Abstract

Multi-hop reasoning is an effective approach for query answering (QA) over incomplete knowledge graphs (KGs). The problem can be formulated in a reinforcement learning (RL) setup, where a policy-based agent sequentially extends its inference path until it reaches a target. However, in an incomplete KG environment, the agent receives low-quality rewards corrupted by false negatives in the training data, which harms generalization at test time. Furthermore, since no golden action sequence is used for training, the agent can be misled by spurious search trajectories that incidentally lead to the correct answer. We propose two modeling advances to address both issues: (1) we reduce the impact of false negative supervision by adopting a pretrained one-hop embedding model to estimate the reward of unobserved facts; (2) we counter the sensitivity to spurious paths of on-policy RL by forcing the agent to explore a diverse set of paths using randomly generated edge masks. Our approach significantly improves over existing path-based KGQA models on several benchmark datasets and is comparable or better than embedding-based models.

## 1 Introduction

Large-scale knowledge graphs (KGs) support a variety of downstream NLP applications such as semantic search (Berant et al., 2013) and dialogue generation (He et al., 2017). Whether curated automatically or manually, practical KGs often fail to include many relevant facts. A popular approach for modeling incomplete KGs is knowledge graph embeddings, which map both entities and relations in the KG to a vector space and learn a truth value function for any potential KG triple parameterized by the entity and relation vectors (Yang et al., 2014; Dettmers et al., 2018).

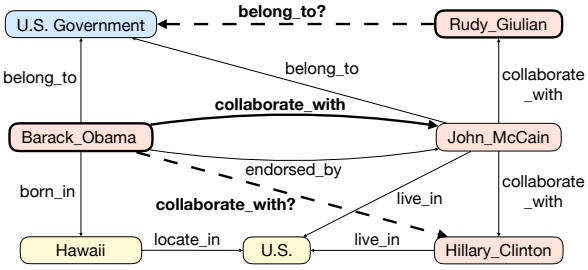


Figure 1: Example of an incomplete knowledge graph which contains missing links (dashed lines) that can possibly be inferred from existing facts (solid lines).

Embedding based approaches ignore the symbolic compositionality of KG relations, which limit their application in more complex reasoning tasks. An alternative solution for KG reasoning is to infer missing facts by synthesizing information from multi-hop paths, e.g. *bornIn(Obama, Hawaii)* $\wedge$ *locatedIn(Hawaii, US)* $\Rightarrow$ *bornIn(Obama, US)*, as shown in Figure 1. Path-based reasoning offers logical insights of the underlying KG and are more directly interpretable. Early work treats it as a link prediction problem and perform maximum-likelihood classification over either discrete path features (Lao et al., 2011, 2012; Gardner et al., 2013) or their hidden representations in a vector space (Guu et al., 2015; Toutanova et al., 2016; McCallum et al., 2017).

More recent work formulates multi-hop reasoning as a sequential decision problem, and leverages reinforcement learning (RL) to perform effective path search (Xiong et al., 2017; Das et al., 2018; Shen et al., 2018; Chen et al., 2018). In particular, MINERVA (Das et al., 2018) uses the REINFORCE algorithm (Williams, 1992) to train an end-to-end model for multi-hop KG query answering: given a query relation and a source entity, the trained agent searches over the KG starting from the source and arrives at the candidate answers without access to any pre-computed paths.

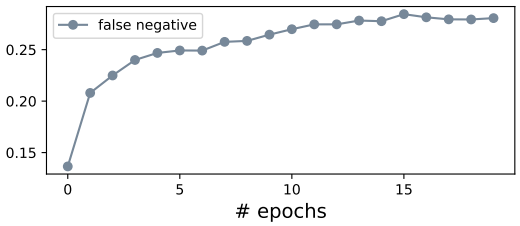


Figure 2: Percentage of false negatives hit (where the model predicted an answer that exists in the full KG but not in the training subset) in the first 20 epochs of walk-based QA training on the UMLS knowledge graph (Kok and Domingos, 2007).

We refer to the RL formulation adopted by MINERVA as "learning to walk towards the answer" or "walk-based query-answering (QA)". Walk-based QA eliminates the need to precompute path features, yet this setup poses several challenges for training. First, because practical KGs are intrinsically incomplete, the agent may arrive at a correct answer nonexistent in the training graph (false negative targets) without receiving any reward (Figure 2). Second, since no ground truth path is available for training, the agent may go through spurious paths that lead to a correct answer only incidentally (false positive paths). Because the commonly used REINFORCE (Williams, 1992) algorithm uses on-policy training which encourages past actions with high reward, it can bias the policy toward spurious paths found early in training (Guu et al., 2017).

We propose two modeling advances for RL approaches in the walk-based QA framework to address the previously mentioned problems. First, instead of using a binary reward based on whether the agent has reached a correct answer or not, we use pre-trained state-of-the-art embedding-based models (Dettmers et al., 2018; Trouillon et al., 2016) to estimate a soft reward for target entities whose correctness cannot be determined. As embedding-based models capture link semantics well, unobserved but correct answers would receive a higher reward score compared to a true negative entity using a well-trained model. Second, we perform action dropout, which randomly blocks some outgoing edges of the agent at each training step so as to enforce effective exploration of a diverse set of paths and dilute the negative impact of the spurious ones. Empirically, our overall model significantly improves over state-of-the-art multi-hop reasoning approaches on four out of five benchmark KG datasets (UMLS, Kinship, FB15k-237, WN18RR). It is also the first path-based model that achieves consistently comparable or better performance than embedding-based models. In addition, we perform a thorough ablation study and result analysis, demonstrating the effect of each modeling innovation.

## 2 Approach

In this section, we first review the walk-based QA framework (§2.2) and the on-policy reinforcement learning approach proposed by Das et al. (2018) (§2.3,§2.4). Then we describe our proposed solutions to the false negative reward and spurious path problems: knowledge-based reward shaping (§2.5) and action dropout (§2.6).

### 2.1 Formal Problem Definition

We formally represent a knowledge graph as $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations. Each directed link in the knowledge graph $l = (e_s, r, e_o) \in \mathcal{G}$ represents a fact (also called a triple).

Given a query $(e_s, r_q, ?)$, where $e_s$ is the source entity and $r_q$ is the relation of interest, the goal is to perform an efficient search over $\mathcal{G}$ and collect the set of possible answers $E_o = \{e_o\}$ s.t. $(e_s, r_q, e_o) \notin \mathcal{G}$ due to incompleteness.

### 2.2 Reinforcement Learning Formulation

The search can be viewed as a Markov Decision Process (MDP) (Sutton and Barto, 1998): starting from $e_s$, the agent sequentially selects an outgoing edge $l$ and traverses to a new entity until it arrives at a target. Specifically, the MDP consists of the following components (Das et al., 2018).

**States** Each state $s_t = (e_t, (e_s, r_q)) \in \mathcal{S}$ is a tuple where $e_t$ is the entity visited at step $t$ and $(e_s, r_q)$ are the source entity and query relation. $e_t$ can be viewed as state-dependent information while $(e_s, r_q)$ are the global context shared by all states.

**Actions** The set of possible actions $A_t \in \mathcal{A}$ of at step $t$ consists of the outgoing edges of $e_t$ in $\mathcal{G}$. Concretely, $A_t = \{(r', e') | (e_t, r', e') \in \mathcal{G}\}$. To give the agent the option of terminating a search, a self-loop edge is added to every $A_t$. Because search is unrolled for a fixed number of steps $T$, the self-loop acts similarly to a "stop" action.

**Transition** A transition function $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is defined by $\delta(s_t, A_t) = \delta(e_t, (e_s, r_q), A_t)$. For walk-based QA, the transition is entirely determined by $\mathcal{G}$.

**Rewards** In the default formulation, the agent receives a terminal reward of 1 if it arrives at a correct target entity at the end of search and 0 otherwise.

$$R_b(s_T) = \mathbb{1}\{(e_s, r_q, e_T) \in \mathcal{G}\}. \tag{1}$$

### 2.3 Policy Network

The search policy is parameterized using state information and global context, plus the search history (Das et al., 2018).

Specifically, every entity and relation in $\mathcal{G}$ is assigned a dense vector embedding $\mathbf{e} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$. The action $a_t = (r_{t+1}, e_{t+1}) \in A_t$ is represented as the concatenation of the relation embedding and the end node embedding $\mathbf{a}_t = [\mathbf{r}; \mathbf{e}'_t]$.

The search history $h_t = (e_s, r_1, e_1, \ldots, r_t, e_t) \in \mathcal{H}$ consists of the sequence of observations and actions taken up to step $t$, and can be encoded using an LSTM:

$$\mathbf{h}_0 = \text{LSTM}(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_s]) \tag{2}$$

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{a}_{t-1}),\ t > 0, \tag{3}$$

where $r_0$ is a special start relation introduced to form a start action with $e_s$.

The action space is encoded by stacking the embeddings of all actions in $A_t$: $\mathbf{A}_t \in \mathbb{R}^{|A_t| \times 2d}$. And the policy network $\pi$ is defined as:

$$\pi_\theta(a_t|s_t) = \sigma(\mathbf{A}_t \times W_2\ \text{ReLU}(W_1[\mathbf{e}_t; \mathbf{h}_t; \mathbf{r}_q])), \tag{4}$$

where $\sigma$ is the softmax operator.

### 2.4 Optimization

The policy network is trained by maximizing the expected reward over all queries in $\mathcal{G}$:

$$J(\theta) = \mathbb{E}_{(e_s, r, e_o) \in \mathcal{G}}[\mathbb{E}_{a_1, \ldots, a_T \sim \pi_\theta}[R(s_T|e_s, r)]]. \tag{5}$$

The optimization is done using the REINFORCE (Williams, 1992) algorithm, which iterates through all $(e_s, r, e_o)$ triples in $\mathcal{G}$[1] and updates $\theta$ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t R(s_T|e_s, r) \log \pi_\theta(a_t|s_t). \tag{6}$$

### 2.5 Knowledge-Based Reward Shaping

According to equation 1, the agent receives a binary reward based on solely the observed answers in $\mathcal{G}$. However, $\mathcal{G}$ is intrinsically incomplete and this approach rewards the false negative search results identically to true negatives. To alleviate this problem, we use existing KG embedding models designed for the purpose of KG completion (Trouillon et al., 2016; Dettmers et al., 2018) to estimate a soft reward for target entities whose correctness is unknown.

Formally, the embedding models map $\mathcal{E}$ and $\mathcal{R}$ to a vector space, and estimate the likelihood of each fact $l = (e_s, r, e_t) \in \mathcal{G}$ using a composition function $f(e_s, r, e_t)$ of the entity and relation embeddings. $f$ is trained by maximizing the likelihood of all facts in $\mathcal{G}$. We propose the following reward shaping strategy (Ng et al., 1999):

$$R(s_T) = R_b(s_T) + (1 - R_b(s_T)) f(e_s, r_q, e_T). \tag{7}$$

Namely, if the destination $e_T$ is a correct answer according to $\mathcal{G}$, the agent receives reward 1. Otherwise the agent receives a fact score estimated by $f(e_s, r_q, e_T)$, which is pre-trained. Here we keep $f$ in its general form and it can be replaced by any state-of-the-art model (Trouillon et al., 2016; Dettmers et al., 2018) or ensemble thereof.

### 2.6 Action Dropout

The REINFORCE training algorithm performs on-policy sampling according to $\pi_\theta(a_t|s_t)$, and updates the $\theta$ stochastically using equation 6. Because the agent does not have access to any oracle path, it is possible for it to arrive at a correct answer $e_o$ via a path which is irrelevant to the query relation. As shown in Figure 1, the path *Obama* $-$*endorsedBy*$\rightarrow$ *McCain* $-$*liveIn*$\rightarrow$ *U.S.* $\leftarrow$*locatedIn*$-$ *Hawaii* cannot be used to infer the fact $bornIn(Obama, Hawaii)$.

Discriminating paths of different qualities is non-trivial, and existing RL approaches for walk-based KGQA largely rely on the terminal reward to bias the search. Since there are often more spurious paths than correct ones, spurious paths are often found first, and following exploration can be increasingly biased towards them (equation 6).

[1] This training strategy treats a query with $n > 1$ answers as $n$ single-answer queries. In particular, given a query $(e_s, r_q, ?)$ with multiple answers $\{e_{t_1}, \ldots e_{t_n}\}$, when training w.r.t. the example $(e_s, r_q, e_{t_i})$, MINERVA removes all $\{e_{t_j} | j \neq i\}$ observed in the training data from the possible set of target entities in the last search step so as to force the agent to walk towards $e_{t_i}$. We adopt the same technique in our training.

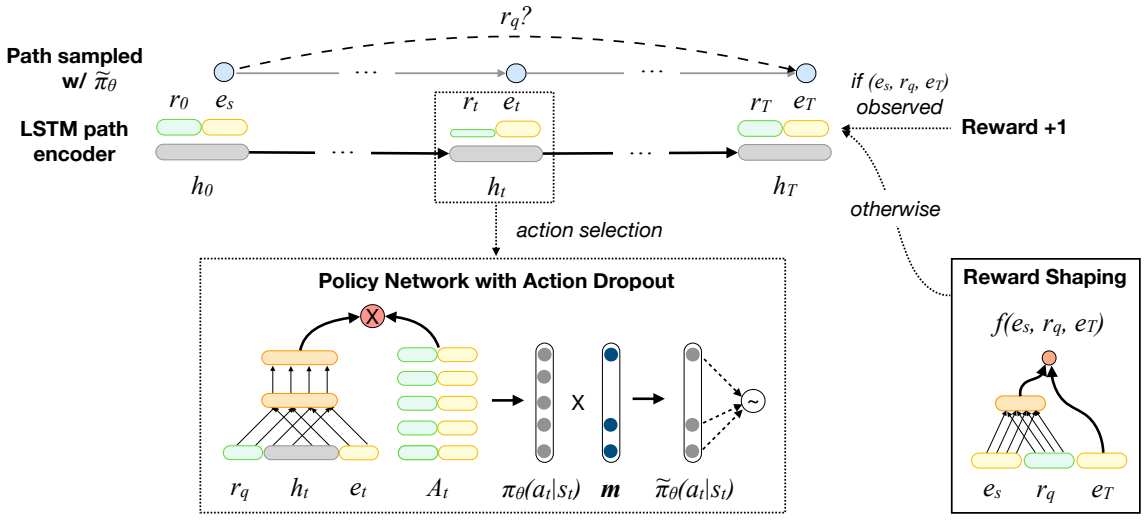


Figure 3: Overall training approach. At each time step $t$, the agent samples an outgoing link according to $\tilde{\pi}_\theta(a_t|s_t)$, which is the stochastic REINFORCE policy $\pi_\theta(a_t|s_t)$ perturbed by a random binary mask $\mathbf{m}$. The agent receives reward 1 if stopped at an observed answer of the query $(e_s, r_q, ?)$; otherwise, it receives reward $f(e_s, r_q, e_T)$ estimated by the reward shaping (RS) network. The RS network is pre-trained and doesn't receive gradient updates.

Entities with larger fan-in (in-degree) and fan-out (out-degree) often exacerbate this problem.

Guu et al. (2017) identified a similar issue in RL-based weakly-supervised semantic parsing, where programs that do not semantically match the user utterance frequently pass the tests. To solve this problem, Guu et al. (2017) proposed randomized beam search combined with a meritocratic update rule to ensure all trajectories that obtain rewards are up-weighted roughly equally.

Here we propose the action dropout technique which achieves similar effect as randomized search and is simpler to implement over graphs. Action dropout randomly masks some outgoing edges for the agent in the sampling step of REINFORCE. The agent then performs sampling[2] according to the adjusted action distribution

$$\tilde{\pi}_\theta(a_t|s_t) = \sigma(\pi_\theta(a_t|s_t) \cdot \mathbf{m} + \epsilon) \quad (8)$$

$$m_i \sim Bernoulli(1-\alpha), i = 1, \ldots |A_t|, \quad (9)$$

where each entry of $\mathbf{m} \in \{0,1\}^{|A_t|}$ is a binary variable sampled from the Bernoulli distribution with parameter $1-\alpha$. A small value $\epsilon$ is used to smooth the distribution in case $\mathbf{m} = \mathbf{0}$, where $\tilde{\pi}_\theta(a_t|s_t)$ becomes uniform.

Our overall approach is illustrated in figure 3.

## 3 Related Work

In this section, we summarize the related work and discuss their connections to our approach.

[2]We only modify the sampling distribution and still use $\pi_\theta(a_t|s_t)$ to compute the gradient update in equation 6.

### 3.1 Knowledge Graph Embeddings

KG embeddings (Bordes et al., 2013; Socher et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Dettmers et al., 2018) are one-hop KG modeling approaches which learn a scoring function $f(e_s, r, e_o)$ to define a fuzzy truth value of a triple in the embedding space. These models can be adapted for query answering by simply return the $e_o$'s with the highest $f(e_s, r, e_o)$ scores. Despite their simplicity, embedding-based models achieved state-of-the-art performance on KGQA (Das et al., 2018). However, such models ignore the symbolic compositionality of KG relations, which limits their usage in more complex reasoning tasks. The reward shaping (RS) strategy we proposed is a step to combine their capability in modeling triple semantics with the symbolic reasoning capability of the path-based approach.

### 3.2 Multi-Hop Reasoning

Multi-hop reasoning focus on learning symbolic inference rules from relational paths in the KG and has been formulated as sequential decision problems in recent works (Xiong et al., 2017; Das et al., 2018; Shen et al., 2018; Chen et al., 2018). In particular, DeepPath (Xiong et al., 2017) first adopted REINFORCE to search for generic representative paths between pairs of entities. DIVA (Chen et al., 2018) also performs generic path search between entities using RL and its variational objective can be interpreted as model-based reward assignment. MINERVA (Das et al., 2018) first introduced RL

to search for answer entities of a particular KG query end-to-end. MINERVA uses entropy regularization to softly encourage the policy to sample diverse paths, and we show that hard action dropout is more effective in this setup. ReinforceWalk (Shen et al., 2018) further proposed to solve the reward sparsity problem in walk-based QA using off-policy learning. ReinforceWalk scores the search targets with a value function which is updated based on the search history accumulated through epochs. In comparison, we leveraged existing embedding-based models for reward shaping, which makes the training more efficient.

### 3.3 Reinforcement Learning

Recently, RL has seen a variety of applications in NLP including machine translation (Ranzato et al., 2015), summarization (Paulus et al., 2017), and semantic parsing (Guu et al., 2017). Compared to the domain of games (Mnih et al., 2013) and many other applications, RL formulations in NLP often have a large action space (e.g., in machine translation, the space of possible actions is the entire vocabulary of a language). This also holds for KGs, as some entities may have thousands of neighbors (e.g. *U.S.*). Since often there is no golden path available for a KG reasoning problem, we cannot use supervised pre-training to give the path search a better start position following the common practice adopted in RL-based natural language generation (Ranzato et al., 2015). On the other hand, the inference paths being studied in a KG are often much shorter (usually containing 2-5 steps) compared to the NL sentences in the sequence generation problems (often containing 20-30 words).

## 4 Experiment Setup

We evaluate our modeling contributions on five KGs from different domains and exhibiting different graph properties (§ 4.1). We compare with two classes of state-of-the-art KG models: multi-hop neural symbolic approaches and KG embeddings (§4.2). In this section, we describe the datasets and our experiment setup in detail.

### 4.1 Dataset

We adopt five benchmark KG datasets for query answering: (1) Alyawarra Kinship, (2) Unified Medical Language Systems (Kok and Domingos, 2007), (3) FB15k-237 (Toutanova et al., 2015), (4) WN18RR (Dettmers et al., 2018), and (5) NELL-995 (Xiong et al., 2017). The statistics of the datasets are shown in Table 1.

| Dataset | #Ent | #Rel | #Fact | #degree | |
|---|---|---|---|---|---|
| | | | | mean | median |
| Kinship | 104 | 25 | 8,544 | 85.15 | 82 |
| UMLS | 135 | 46 | 5,216 | 38.63 | 28 |
| FB15k-237 | 14,505 | 237 | 272,115 | 19.74 | 14 |
| WN18RR | 40,945 | 11 | 86,835 | 2.19 | 2 |
| NELL-995 | 75,492 | 200 | 154,213 | 4.07 | 1 |

Table 1: KGs used in the experiments sorted by increasing sparsity level.

### 4.2 Baselines and Model Variations

We compare with three embedding based models: DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016) and ConvE (Dettmers et al., 2018). We also compare with three multi-hop neural symbolic models: (a) NTP-$\lambda$, an improved version of Neural Theorem Prover (Rocktäschel and Riedel, 2017), (b) Neural Logical Programming (NeuralLP) (Yang et al., 2017) and (c) MINERVA. For our own approach, we include two model variations that use ComplEx and ConvE as the reward shaping modules respectively, denoted as Ours(ComplEx) and Ours(ConvE). We quote the results of NeuralLP, NTP-$\lambda$ and MINERVA as reported in Das et al. (2018), and replicated the embedding based systems.[3]

### 4.3 Implementation Details

**Beam Search Decoding** We perform beam search decoding to obtain a list of unique entity predictions. Because multiple paths may lead to the same target entity, we compute the list of unique entities reached in the final search step and assign each of them the maximum score among all paths that led to it. We then output the top-ranked unique entities. We find this approach to improve over outputting the entities ranked at the beam top directly, as many of them are repetitions.

**KG Setup** Following previous work, we treat every KG link as bidirectional and augment the graph with the reversed $(e_o, r^{-1}, e_s)$ links. We use the same train, dev, and test set splits as Das et al. (2018). We exclude any link from the dev and test set (and its reversed link) from the train set in case there is an overlap. Following Das et al. (2018),

[3] Das et al. (2018) reported MINERVA results with the entity embedding usage as an extra hyperparameter – the quoted performance of MINERVA in Table 2 on UMLS and Kinship were obtained with entity embeddings setting to zero. In contrast, our system always uses trained entity embeddings.

| Model | UMLS | | | Kinship | | | FB15k-237 | | | WN18RR | | | NELL-995 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | @1 | @10 | MRR | @1 | @10 | MRR | @1 | @10 | MRR | @1 | @10 | MRR | @1 | @10 | MRR |
| DistMult (Yang et al., 2014) | 82.1 | 96.7 | 86.8 | 48.7 | 90.4 | 61.4 | 32.4 | 60.0 | 41.7 | **43.1** | 52.4 | **46.2** | 55.2 | 78.3 | 64.1 |
| ComplEx (Trouillon et al., 2016) | 89.0 | 99.2 | 93.4 | **81.8** | **98.1** | **88.4** | 32.8 | 61.6 | 42.5 | 41.8 | 48.0 | 43.7 | 64.3 | 86.0 | 72.6 |
| ConvE (Dettmers et al., 2018) | **93.2** | **99.4** | **95.7** | 79.7 | **98.1** | 87.1 | **34.1** | **62.2** | **43.5** | 40.3 | **54.0** | 44.9 | **67.8** | **88.6** | **76.1** |
| NeuralLP (Yang et al., 2017) | 64.3 | 96.2 | 77.8 | 47.5 | 91.2 | 61.9 | 16.6 | 34.8 | 22.7 | 37.6 | **65.7** | 46.3 | – | – | – |
| NTP-$\lambda$ (Rocktäschel et. al. 2017) | 84.3 | **100** | 91.2 | 75.9 | 87.8 | 79.3 | – | – | – | – | – | – | – | – | – |
| MINERVA (Das et al., 2018) | 72.8 | 96.8 | 82.5 | 60.5 | 92.4 | 72.0 | 21.7 | 45.6 | 29.3 | 41.3 | 51.3 | 44.8 | **66.3** | 83.1 | 72.5 |
| Ours(ComplEx) | 88.7 | 98.5 | 92.9 | **81.1** | **98.2** | **87.8** | **32.9** | 54.4 | 39.3 | **43.7** | 54.2 | **47.2** | 65.5 | 83.6 | 72.2 |
| Ours(ConvE) | **90.2** | 99.2 | **94.0** | 78.9 | **98.2** | 86.5 | 32.7 | **56.4** | **40.7** | 41.8 | 51.7 | 45.0 | 65.6 | **84.4** | **72.7** |

Table 2: Query answering performance compared to state-of-the-art embedding based approaches (top part) and multi-hop reasoning approaches (bottom part). The @1, @10 and MRR metrics were multiplied by 100. We highlight the best approach in each category.

we cut the maximum number of outgoing edges of an entity by a threshold $\eta$ to prevent GPU memory overflow: for each entity we retain its top-$\eta$ neighbors with the highest PageRank scores (Page et al., 1999).

**Hyperparameters** We set the entity and relation embedding size to 200 for all models. We use Xavier initialization (Glorot and Bengio, 2010) for the embeddings and NN layers. For ConvE, we use the same convolution layer and label smoothing hyperparameters as Dettmers et al. (2018). For path-based models, we use a three-layer LSTM as the path encoder and set its hidden dimension to 200. We perform grid search on the reasoning path length $(2, 3)$, the node fan-out threshold $\eta$ (256-512) and the action dropout rate $\alpha$ (0.1-0.9). Following Das et al. (2018), we add an entropy regularization term in the objective and tune the weight parameter $\beta$ within 0-0.1. We use Adam optimization (Kingma and Ba, 2014) and search the learning rate (0.001-0.003) and mini-batch size (128-512).[4] For all models we apply dropout to the entity and relation embeddings and all feed-forward layers, and search the dropout rates within 0-0.5. We use a decoding beam size of 512 for NELL-995 and 128 for the other datasets.

**Evaluation Protocol** We convert each triple $(e_s, r, e_o)$ in the test set into a query and compute ranking-based evaluation metrics. The models take $e_s, r$ as the input and output a list of candidate answers $E_o = [e^1, \ldots, e^L]$ ranked in decreasing order of confidence score. We compute $r_{e_o}$, the rank of $e_o$ among $E_o$, after removing the other correct answers from $E_o$ and use it to compute two types of metrics: (1) Hits@$k$ which is the percentage of examples where $r_{e_o} \leq k$ and (2) mean reciprocal rank (MRR) which is the mean of $1/r_{e_o}$ for all examples in the test set. We use the entire test set for evaluation, with the exception of NELL-995, where test triples with unseen entities are removed following Das et al. (2018).

We will release the Pytorch implementation of all experiments. Please check the authors' web page for updates.

## 5 Results

### 5.1 Model Comparison

Table 2 shows the evaluation results of our proposed approach and the baselines. The top part presents embedding based approaches and the bottom part presents multi-hop reasoning approaches.[5]

We find embedding based models perform strongly on several datasets, achieving overall best evaluation metrics on UMLS, Kinship, FB15K-237 and NELL-995 despite their simplicity. While existing path based approaches can achieve comparable performance on some of the datasets (WN18RR, NELL-995, and UMLS), the performance gap to embedding based approaches on the others (Kinship and FB15k-237) are considerable (9.1 and 14.2 absolute points respectively). A possible reason for this is that embedding based methods map every link in the KG into the same embedding space, which implicitly encodes the connectivity of the whole graph. In contrast, path based models use the discrete representation of a

[4] On some datasets, we found larger batch size to continue improving the performance but had to stop at 512 due to memory constraints.

[5] We report model robustness measurements in § A.1.

| Model | UMLS | Kinship | FB15k237 | WN18RR | NELL995 |
|---|---|---|---|---|---|
| Ours(ConvE) | **73.0** | **75.0** | **38.2** | 43.8 | **78.8** |
| −RS | 67.7 | 66.5 | 35.1 | **45.7** | 78.4 |
| −AD | 61.3 | 65.4 | 31.0 | 39.1 | 76.1 |

Table 3: Comparison of dev set MRR of Ours(ConvE) and models without reward shaping and action dropout.

KG as input, and therefore have to disregard a significant proportion of the combinatorial path space by selection. For some path based approaches, computation cost is a bottleneck. In particular, NeuralLP and NTP-$\lambda$ failed to scale to the larger datasets and their results are omitted from the table, as Das et al. (2018) reported.

Ours is the first multi-hop reasoning approach which is consistently comparable or better than embedding based approaches on all five datasets. The best single model, Ours(ConvE), improves the SOTA performance of path-based models on three datasets (UMLS, Kinship, and FB15k-237) by 4%, 9%, and 39% respectively. On NELL-995, our approach did not significantly improve over existing SOTA. The NELL-995 dataset consists of only 12 relations in the test set and, as we further detail in the analysis (§ 5.3.3), our approach is less effective for those relation types.

The model variations using different reward shaping modules perform similarly. While a better reward shaping module typically results in a better overall model, an exception is WN18RR, where ComplEx performs slightly worse on its own but is more helpful for reward shaping. We left the study of the relation between reward shaping module accuracy and the overall model performance as future work.

## 5.2 Ablation Study

We perform an ablation study where we remove reward shaping (−RS) and action dropout (−AD) from Ours(ConvE) and compare their MRRs to the whole model on the dev sets.[6] As shown in Table 3, on most datasets, removing each component results in a significant performance drop. The exception is WN18RR, where removing the ConvE reward shaping module improves the performance.[7] Removing reward shaping on NELL-995 does not change the results significantly. In general, removing action dropout has a greater impact, suggesting that thorough exploration of the path space is important across datasets.

## 5.3 Analysis

### 5.3.1 Convergence Rate

We are interested in studying the effect of each proposed strategy on the training convergence rate. In particular, we expect reward shaping to accelerate the convergence of RL (to a better performance level) as it propagates prior knowledge about the underlying KG to the agent. On the other hand, a fair concern for action dropout is that it can be slower to train, as the agent is forced to explore a more diverse set of paths. Figure 4 eliminates this concern.

The first row of Figure 4 shows the change in dev set MRR of Ours(ConvE) (green) and the two ablated models w.r.t. # epochs. In general, the proposed approach is able to converge to a higher accuracy level much faster than either of the ablated models and the performance gap often persists until the end of training (on UMLS, Kinship, and FB15k-237). Particularly, on FB15k-237, our approach still shows improvement even after the two ablated models start to overfit, with −AD beginning to overfit sooner. On WN18RR, introducing reward shaping hurt dev set performance from the beginning, as discussed in § 5.2. On NELL-995, Ours(ConvE) performs significantly better in the beginning, but −RS gradually reaches a comparable performance level. It is especially interesting that introducing action dropout improves the model performance on all datasets from early on. A possible explanation for this is by exploring a more diverse set of paths the search agent learns policies that generalize better.

### 5.3.2 Path Diversity

We also compute the total number of unique paths the agent explores during training and visualize its change w.r.t. # training epochs in the second row of Figure 4. We include both the edge relation and intermediate entity when counting a unique path. First we observe that, on all datasets, the agent explores a large number of paths before

[6] According to Table 3 and Table 2, the dev and test set evaluation metrics differ significantly on UMLS, Kinship and FB15k-237. We discuss the cause of this in § A.2.

[7] A possible explanation for this is that as path-based models tend to outperform the embedding based approaches on WN18RR, ConvE may be supplying more noise than useful information about the KG. Yet counter-intuitively, we found that adding the ComplEx reward shaping module helps, despite the fact that CompLEx performs slightly worse than ConvE on this dataset. This indicates that dev set accuracy is not the only factor which determines the effectiveness of reward shaping.

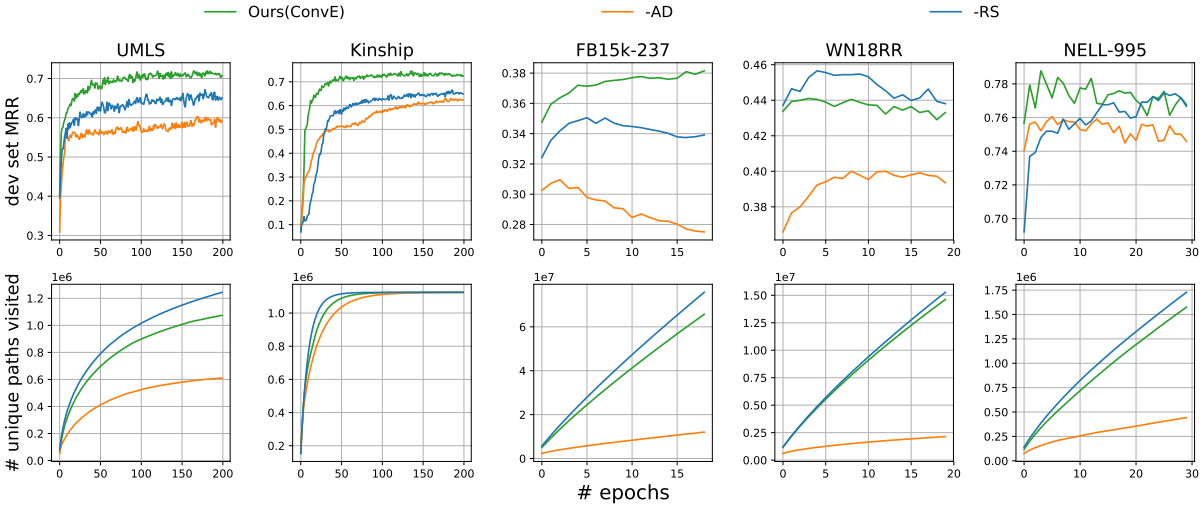


Figure 4: Illustration of convergence rate and path exploration efficiency. The three curves in each subplot represents Ours(ConvE) (green) and the two ablated models: −RS (blue) and −AD (orange). The top row shows the change of dev set MRR and the bottom row shows the growth of # unique paths explored w.r.t. # epochs.

| Dataset | To-many | | | | To-one | | | |
|---|---|---|---|---|---|---|---|---|
| | % | Ours(ConvE) | −RS | −AD | % | Ours(ConvE) | −RS | −AD |
| UMLS | 99.1 | 73.1 | 67.9 (**-7**%) | 61.3 (**-16**%) | 0.9 | 62.5 | 55.5 (**-11**%) | 54.4 (**-13**%) |
| Kinship | 100 | 75 | 66.5 (**-11**%) | 65.4 (**-13**%) | 0 | – | – | – |
| FB15k-237 | 76.6 | 28.3 | 24.5 (**-13**%) | 20.9 (**-26**%) | 23.4 | 72 | 69.8 (**-3**%) | 63.9 (**-11**%) |
| WN18RR | 52.8 | 65 | 65.7 (+1%) | 57.9 (**-11**%) | 47.2 | 20.1 | 23.2 (+16%) | 18.1 (**-10**%) |
| NELL-995 | 12.9 | 55.7 | 62.1 (+12%) | 56.9 (+2%) | 87.1 | 81.4 | 80.7 (**-1**%) | 80.5 (**-1**%) |

Table 4: MRR evaluation of different relation types (to-many vs. to-one) on five datasets. The % columns show the percentage of examples of each relation type found in the development split of the corresponding dataset. In general, our proposed techniques improve the prediction results for to-many relations more significantly.

reaching a good performance level. The speed of path discovery slowly decreases as training progresses. On smaller KGs (UMLS and Kinship), the rate of encountering new paths is significantly lower after a certain number of epochs, and the dev set accuracy plateaus correspondingly. On much larger KGs (FB15k-237, WN18RR, and NELL-995), we did not observe a significant slowdown before severe overfitting occurs and the dev set performance starts to drop. A possible reason for this is that the larger KGs are also more sparsely connected compared to the smaller KGs (Table 1), therefore it is less efficient to gain generalizable knowledge from the KG by exploring a limited proportion of the path space through sampling.

Second, it is interesting to see that while removing action dropout significantly lowers the effectiveness of path exploration (orange vs. green), removing reward shaping slightly improves the # paths visited during training for all datasets. This indicates that the correlation between # paths explored and dev set performance is not strictly positive. The best performing model in general is not the model that explored the largest # paths. It also demonstrates the role of reward shaping as a regularizer which guides the agent to avoid noisy paths with its prior knowledge.

### 5.3.3 Performance w.r.t. Relation Types

We investigate the behaviors of our proposed approach w.r.t different relation types. For each KG, we classify its set of relations into two categories based on the answer set cardinality. Specifically, we define the metric $\xi_r$ as the average answer set cardinality of all queries with topic relation $r$. We count $r$ as a "to-many" relation if $\xi_r > 1.5$, which indicates that most queries in relation $r$ has more than 1 correct answer; we count $r$ as a "to-one" relation otherwise, meaning most queries of this relation have only 1 correct answer.

Table 4 shows the percentage of examples of to-many and to-one relations on each dev dataset and the MRR evaluation metrics of previously studied models on examples of each relation type. Since UMLS and Kinship are densely connected,

| Dataset | Seen Queries | | | | Unseen Queries | | | |
|---|---|---|---|---|---|---|---|---|
| | % | Ours(ConvE) | −RS | −AD | % | Ours(ConvE) | −RS | −AD |
| UMLS | 97.2 | 73.1 | 67.9 (**-7**%) | 61.4 (**-16**%) | 2.8 | 68.5 | 61.5 (**-10**%) | 58.7 (**-14**%) |
| Kinship | 96.8 | 75.1 | 66.5 (**-11**%) | 65.8 (**-12**%) | 3.2 | 73.6 | 64.3 (**-13**%) | 53.3 (**-27**%) |
| FB15k-237 | 76.1 | 28.3 | 24.3 (**-14**%) | 20.6 (**-27**%) | 23.9 | 70.9 | 69.1 (**-2**%) | 63.9 (**-10**%) |
| WN18RR | 41.8 | 60.8 | 62.0 (+2%) | 53.4 (**-12**%) | 58.2 | 31.5 | 33.9 (+7%) | 28.8 (**-9**%) |
| NELL-995 | 15.3 | 40.4 | 45.9 (+14%) | 42.5 (+5%) | 84.7 | 85.5 | 84.7 (**-1**%) | 84.3 (**-1**%) |

Table 5: MRR evaluation of seen queries vs. unseen queries on five datasets. The % columns show the percentage of examples of seen/unseen queries found in the development split of the corresponding dataset.

they almost exclusively contain to-many relations. FB15k-237 mostly contains to-many relations. In Figure 4, we observe the biggest relative gains from the ablated models on these three datasets. WN18RR is more balanced and consists of slightly more to-many relations than to-one relations. The NELL-995 dev set is an exception which almost exclusively consists of to-one relations. There is no general performance pattern over the two relation types: on some datasets all models perform better on to-many relations (UMLS, WN18RR) while others reveal the opposite trend (FB15k-237, NELL-995). We leave the study of such differences to future work.

We show the relative performance change of the ablated models −RS and −AD w.r.t. Ours(ConvE) in parentheses. We observe that in general our proposed enhancements are effective in improving query-answering over both relation types (more effective for to-many relations). However, adding the ConvE reward shaping module on WN18RR hurts the performance over both to-many and to-one relations (worse for to-one relations). On NELL-995, both techniques hurt the performance over to-many relations.

#### 5.3.4 Performance w.r.t. Seen Queries vs. Unseen Queries

Since most benchmark datasets randomly split the KG triples into train, dev and test sets, the queries that have multiple answers may fall into multiple splits. As a result, some of the test queries $(e_s, r_q, ?)$ are seen in the training set (with a different set of answers) while the others are not. We investigate the behaviors of our proposed approach w.r.t. seen and unseen queries.

Table 5 shows the percentage of examples associated with seen and unseen queries on each dev dataset and the corresponding dev set MRR evaluation metrics of the previously studied models. On most datasets, the ratio of seen vs. unseen queries is similar to that of to-many vs. to-one relations (Table 4) as a result of random data split, with the exception of WN18RR. On some datasets, all models perform better on seen queries (UMLS, Kinship, WN18RR) while others reveal the opposite trend. On NELL-995 and our proposed enhancements are not effective over the seen queries. We leave the investigation of these model behaviors to future work. Our proposed enhancements improves the performance over unseen queries in most cases, with −AD being more effective.

## 6 Conclusions

We propose two modeling advances for end-to-end RL-based knowledge graph query answering: reward shaping and action dropout. Our approach improves over state-of-the-art multi-hop reasoning models consistently on several benchmark KGs. A detailed analysis indicates that the access to a more accurate environment representation (reward shaping) and a more thorough exploration of the search space (action dropout) are important to the performance boost.

On the other hand, the performance gap between RL-based approaches and the embedding-based approaches on KGQA remains. For future work, we would like to investigate learnable reward shaping and action dropout schemes and apply model-based RL to this domain.

## Acknowledgements

We thank Mark O. Riedl, Yingbo Zhou, James Bradbury and Vena Jia Li for their feedback on early draft of the paper, and Mark O. Riedl for helpful conversations on reward shaping. We thank the anonymous reviewers and the Salesforce research team members for their thoughtful comments and discussions.

## References

Regina Barzilay and Min-Yen Kan, editors. 2017. *Proceedings of the 55th Annual Meeting of the Associa-*

*tion for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In (Yarowsky et al., 2013), pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795, USA. Curran Associates Inc.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. *CoRR*, abs/1803.06581.

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*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M. Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In (Yarowsky et al., 2013), pages 833–838.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org.

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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 318–327. The Association for Computational Linguistics.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In (Barzilay and Kan, 2017), pages 1051–1062.

Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors. 2017. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In (Barzilay and Kan, 2017), pages 1766–1776.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Stanley Kok and Pedro M. Domingos. 2007. Statistical predicate invention. In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 433–440. ACM.

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*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando C. N. Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1017–1026. ACL.

Andrew McCallum, Arvind Neelakantan, Rajarshi Das, and David Belanger. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 132–141. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation rank-

ing: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In (Guyon et al., 2017), pages 3791–3803.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. Reinforcewalk: Learning to walk in graph with monte carlo tree search. *CoRR*, abs/1802.04394.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 926–934, USA. Curran Associates Inc.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

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 *EMNLP*, pages 1499–1509. The Association for Computational Linguistics.

Kristina Toutanova, Xi 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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.

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

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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 564–573. Association for Computational Linguistics.

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

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In (Guyon et al., 2017), pages 2316–2325.

David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors. 2013. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL.

# A Appendix

## A.1 Model Robustness

We run the best embedding-based model ConvE and Ours(ConvE) on all datasets using 5 different random seeds with all other hyperparameters fixed. Table 6 reports the mean and standard deviation of each model. We observe that both models demonstrate a small standard deviation ($< 0.01$) on all datasets.

| Dataset | ConvE | Ours(ConveE) |
|---|---|---|
| UMLS | 95.5±0.4 | 93.7±0.2 |
| Kinship | 86.9±0.3 | 86.2±0.7 |
| FB15k-237 | 43.5±0.1 | 40.7±0.2 |
| WN18RR | 45.3±0.4 | 44.7±0.2 |
| NELL-995 | 76.2±0.3 | 72.7±0.4 |

Table 6: Test set MRR×100 mean and standard deviation across five runs on all datasets.

## A.2 Development Set Evaluation Using Complete KGs

Comparing Table 2 and Table 3 reveals that the dev set MRRs are significantly lower than the test set MRRs on some datasets (UMLS, Kinship and FB15k-237). Such discrepancies are caused by the multi-answer queries in these datasets. As most benchmark datasets randomly split the KG triples into train/dev/test sets, the queries that have multiple answers may fall into multiple splits. Because we hide all triples in the test set during the dev set evaluation, some predictions generated during dev set evaluation were wrongly punished as false negatives. In contrast, the test set evaluation metrics are computed using the complete KGs. Access to the complete KG eliminates most of the false negatives cases and hence increases the performance.

| Model | UMLS | Kinship | FB15k237 | WN18RR | NELL995 |
|---|---|---|---|---|---|
| Ours(ConvE) | **95.1** | **86.8** | **41.8** | 44.1 | **78.8** |
| −RS | 85.6 | 75.7 | 37.1 | **46.1** | 78.4 |
| −AD | 76.2 | 75.9 | 32.4 | 39.3 | 76.1 |

Table 7: Comparison of dev set MRR computed using the complete KGs of Ours(ConvE) and models without reward shaping and action dropout.

Table 7 shows the dev set MRR of the same systems shown in Table 3 with the MRRs computed using the complete KGs. On four of the datasets, the evaluation metrics increases significantly to the level that is comparable to those on the test set, with the relative improvement correlated with the average node fan-out in the KG (Table 1).

Notice that Table 7 is generated after all hyperparameters were fixed and the purpose is to show the effects of such dataset peculiarities. To avoid potential test set leakage, hyperparameter search should be done with the test set triples hidden (Table 1) instead of with the full KG.

## A.3 Action Dropout Rates Used for Different KGs

Table 8 show the action dropout rates used for all KG datasets in our experiments. In general, larger action dropout rates are necessary for KGs that are densely connected. We find a positive correlation between the optimal action dropout rate and the average node fan-out (Table 1).

For UMLS and Kinship, we tried setting the action dropout rate to 1.0 (completely random sampling) and observed small but significant performance drop. Random sampling performs reasonably well on these two datasets possibly due to the fact that they are small. For larger KGs (FB15k-237, WN18RR, NELL-995), policy-guided sampling is necessary.

| Dataset | $\alpha$ |
|---|---|
| UMLS | 0.95 |
| Kinship | 0.9 |
| FB15k-237 | 0.5 |
| WN18RR | 0.1 |
| NELL-995 | 0.1 |

Table 8: Action dropout rates used in our experiments.