Information Fusion 88 (2022) 12–21

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

Heterogeneous relational reasoning in knowledge graphs with reinforcement learning

Mandana Saebi [a], Steven Kreig [a], Chuxu Zhang [b], Meng Jiang [a], Tomasz Kajdanowicz [c], Nitesh V. Chawla [a,*]

[a] *University of Notre Dame, Notre Dame, 46556, IN, USA*
[b] *Brandeis University, 415 South St, Waltham, 02453, USA*
[c] *Wroclaw University of Science and Technology, 415 South St, Wroclaw, 53-370, Poland*

ARTICLE INFO

Keywords:
Knowledge graphs
Relational reasoning
Reinforcement learning

ABSTRACT

Path-based relational reasoning over knowledge graphs has become increasingly popular due to a variety of downstream applications such as question answering in dialogue systems, fact prediction, and recommendation systems. In recent years, reinforcement learning (RL) based solutions for knowledge graphs have been demonstrated to be more interpretable and explainable than other deep learning models. However, the current solutions still struggle with performance issues due to incomplete state representations and large action spaces for the RL agent. We address these problems by developing HRRL (Heterogeneous Relational reasoning with Reinforcement Learning), a type-enhanced RL agent that utilizes the local heterogeneous neighborhood information for efficient path-based reasoning over knowledge graphs. HRRL improves the state representation using a graph neural network (GNN) for encoding the neighborhood information and utilizes entity type information for pruning the action space. Extensive experiments on real-world datasets show that HRRL outperforms state-of-the-art RL methods and discovers more novel paths during the training procedure, demonstrating the explorative power of our method.

1. Introduction

Relational reasoning is an important goal of machine learning and artificial intelligence [1–4]. In the context of large-scale knowledge graphs, relational reasoning addresses a number of important applications, such as question answering [5,6], dialogue systems [7,8], and recommender systems [9–11]. Most knowledge graphs are incomplete and the problem of inferring missing relations, or knowledge graph reasoning, has become an increasingly important research topic [12]. Several works have treated this as a link prediction problem and attempted to solve it using graph embedding or deep learning approaches [13–20]. These methods embed the knowledge graph into a vector space and use a similarity measure to identify the entities that are likely to be connected. However, they are unable to discover multi-hop relations. Besides, they do not provide an explicit explanation for their predictions and often rely on other analytical methods to provide interpretation for their results. As a result, it is often hard to trust the predictions made by those embedding-based methods.

Recent advances in the area of deep reinforcement learning (RL) have inspired RL-based solutions for the knowledge graph reasoning problem [5,6,21–25]. RL-based methods formulate the task of knowledge graph reasoning as a sequential decision-making process in which the goal is to train an RL agent to walk over the graph by taking a sequence of actions (i.e., choosing the next entity) that connects the source to the target entity. The sequences of entities and relations can be directly used as a logical reasoning path for interpreting model predictions. For example, in order to answer the query *(Reggie Miller, plays sport, ?)*, the agent may find the following reasoning path in the knowledge graph: *Reggie Miller* $\xrightarrow{competes\ with}$ *Michael Jordan* $\xrightarrow{plays\ sport}$ *Basketball*. In this case, *Reggie Miller*, *Michael Jordan*, and *Basketball* are all entities in the knowledge graph, and *competes with* and *plays sport* are relations. The agent is thus learning to navigate the entities and relations of the knowledge graph. The RL solutions demonstrate competitive accuracy with other deep learning methods and improve the interpretability of the reasoning process. However, there remain some fundamental and open challenges that we will address in this work:

Large action space. In knowledge graphs, facts are represented as binary relations between entities. Real-world knowledge graphs contain

* Corresponding author.
E-mail addresses: msaebi@nd.edu (M. Saebi), skrieg@nd.edu (S. Kreig), chuxuzhang@brandeis.edu (C. Zhang), mjiang2@nd.edu (M. Jiang), tomasz.kajdanowicz@pwr.edu.pl (T. Kajdanowicz), nchawla@nd.edu (N.V. Chawla).

https://doi.org/10.1016/j.inffus.2022.07.001
Received 13 July 2021; Received in revised form 26 February 2022; Accepted 7 July 2022
Available online 13 July 2022

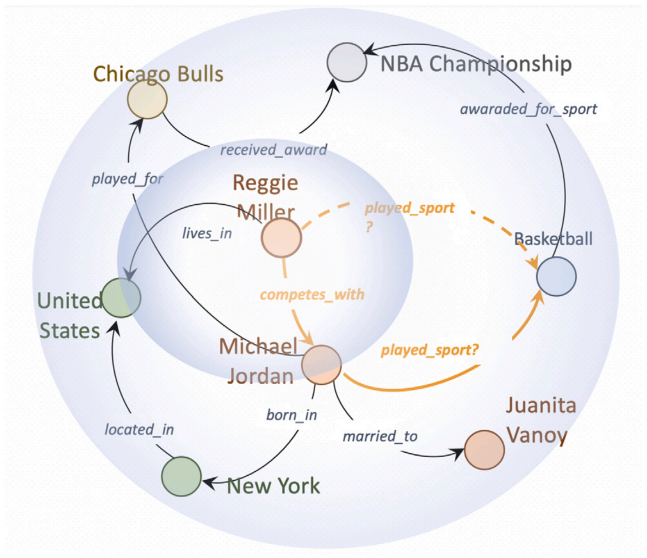

Fig. 1. Given the query *(Reggie Miller, plays sport, ?)*, an RL-based solution may choose actions that lead to an incorrect answer, such as: *Reggie Miller* $\xrightarrow{competes\ with}$ *Michael Jordan* $\xrightarrow{played\ for}$ *Chicago Bulls*, *Reggie Miller* $\xrightarrow{competes\ with}$ *Michael Jordan* $\xrightarrow{born\ in}$ *New York City* and *Reggie Miller* $\xrightarrow{competes\ with}$ *Michael Jordan* $\xrightarrow{married\ to}$ *Juanita Vanoy*). HRRL is more likely to choose the correct path: *Reggie Miller* $\xrightarrow{competes\ with}$ *Michael Jordan* $\xrightarrow{plays\ sport}$ *Basketball* by paying attention to the entity type (entities with the same type are colored the same) and entity's neighborhood. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a large number of entities and relations. As a result, the RL agent often encounters nodes with large a degree, which increases the complexity of choosing the next action. In these cases, exploring the possible paths to determine the optimal action is computationally expensive, and in many cases beyond the memory limit of a single GPU. Previous studies have shown that type information can improve the knowledge graph reasoning performance [26–29] using deep learning approaches. To improve the search efficiency, we first introduce the type representation to encode the entity type information, which we include in the representation of the state space. We then prune the action space based on the type information. This guides the RL agent to constrain the search space to the entities whose type best matches the previously taken actions and, as a result, avoids incorrect reasoning paths. In the above example of reasoning path for query *(Reggie Miller, plays sport, ?)*, suppose the entity *Michael Jordan* has a relatively high degree, and is connected to several other entities through different relations (e.g., *Michael Jordan* $\xrightarrow{played\ for}$ *Chicago Bulls*, *Michael Jordan* $\xrightarrow{born\ in}$ *New York City*, *Michael Jordan* $\xrightarrow{married\ to}$ *Juanita Vanoy*). None of these additional entities are useful for answering the query and may mislead the agent. However, the RL agent may be able to learn that the next entity type is a *sport* rather than a *person* or a *location* (Fig. 1).

Accurate representation of entity's neighborhood. Existing RL-based methods for knowledge graph reasoning do not capture the entity's neighborhood information. Previous studies [30,31] have shown that the local neighborhood structure can improve the fact prediction performance. This motivated us to apply graph neural network (GNN) [32] to encode the entity's neighborhood information and leverage the state representation with the type and neighborhood information of the entity. We demonstrate that utilizing the local heterogeneous neighborhood information improves the performance of the RL agent on the long-tailed relations, which in turn significantly improves model performance for the knowledge graph reasoning task. In the example query *(Reggie Miller, plays sport, ?)*, the target entity *Basketball* is located within a two-hop distance of the entity *Reggie Miller* (Fig. 1).

We propose Heterogeneous Relational reasoning with Reinforcement Learning (HRRL) to address the aforementioned challenges. To summarize, our main **contributions** as part of HRRL include:

1. Designing an expressive vector representation for entity type-embeddings and improving the choice of next actions using the entity type information.
2. Develop an efficient action space pruning strategy using the entity type information.
3. Incorporating GNN for capturing the local neighborhood information in the state representation.
4. Comprehensive evaluation of our method on three public datasets against several state-of-the-art RL methods.

The rest of the paper is organized as follows. We survey the related works Section 2, and present the details of our model in Section 3. Section 4 presents experimental results and related discussions, followed by the conclusion and future work discussion in Section 6.

2. Related work

Relational reasoning over knowledge graphs has attracted significant attention over the past few years. Below we provide a brief overview of different approaches.

2.1. Rule-based methods

Rule-Based methods such as Neural LP [33], NTP [34], and PoLo [35] generate reasoning rules and then apply them to fill missing links based on the extracted rules. Neural LP uses a differential rule learning system that allows end-to-end training. However, it is computationally expensive since it works on symbols, and differential memory requires access to the full memory. Therefore, it does not scale to large-scale graphs. Neural Theorem Provers (NTP) on the other hand, operates on vectors (instead of symbols), but it still suffers from computational complexity as the backward chaining inference can be performed for any pairs of vectors.

2.2. Embedding methods

Other recent works [13–20,36] approached this problem by embedding the relation and entities into a vector space and identifying related entities by similarity in the vector space. However, these methods have some important drawbacks. In particular, they cannot perform multi-hop reasoning. That is, they only consider pairwise relationships and cannot reason along a path. Furthermore, they cannot explain the reasoning behind their predictions. Because they treat the task as a link prediction problem, the output of their prediction is a probabilistic value.

2.3. Path-based methods

With the recent success of deep RL such as AlphaGO [37], researchers began to adopt RL to solve a variety of problems that were conventionally addressed by deep learning methods, such as ad recommendation [9–11], dialogue systems [7,8], and question answering [5, 6]. As a result, more recent methods use RL to solve the relational reasoning problem in knowledge graphs by framing it as a sequential decision-making process [5,6,22,24,38,39]. DeepPath [5] was the first method that uses RL to find relation paths between two entities in knowledge graphs. It walks from the source entity, chooses a relation, and translates to any entity in the tail entity set of the relation. DeepPath was inspired by PRA [40], which is a non-RL path-finding approach that uses random walk with restart strategies for multi-hop reasoning. MINERVA [6], addressed the limitations of DeepPath by jointly selecting an entity-relation pair via a policy network. Li et al. [39] use a multi-agent RL approach where two agents are used to perform the relation selection and entity selection iteratively. Lin et al. [22] employ reward shaping to address the problem of the sparse reward signal and action dropout to reduce the effect of incorrect paths. Xian et al. [25] use relational reasoning for recommender systems and designed both a multi-hop scoring function and a user-conditioned action pruning strategy to improve the efficiency of RL-based recommendation. More recent methods [41–44] attempt to improve the performance by incorporating the attention mechanism, such as AttPath in [41], which uses attention for knowledge graph reasoning as memory components to avoid pre-training the entity embeddings. Bai et al. [45] leverage temporal information for improving the reasoning performance, and Liao et al. [46] proposes a solution by modifying the stop condition during the reasoning process. Because these RL models treat the knowledge graph reasoning problem as a path reasoning problem instead of link prediction, they can overcome both drawbacks of embedding methods that are outlined above. However, the RL models have drawbacks of their own, the most notable of which are computational cost and predictive accuracy. Many of these RL methods have tried to combine the representational power of embeddings and the reasoning power of RL by training an agent to navigate an embedding space. For example, Lin et al. [22] built an agent-based model on top of pre-trained embeddings generated by ConvE [17]. While we take a similar modular approach, our solution enriches the state representation with additional information about entity types and local neighborhood information and improves the efficiency of the RL algorithm by pruning the action space based on the heterogeneous context of entities. In light of recent works on heterogeneous networks that have demonstrated the importance of heterogeneous information [26,27,47–49] and local neighborhood information [30,31] in graph mining, we take a broader approach. We propose to include entity type information in the state representation to help improve the search efficiency for the RL agent by taking more informed actions considering the heterogeneous context. We also learn the heterogeneous neighborhood information simultaneously with training the RL agent to improve the predictions by enriching the state representation with local neighborhood information.

3. Model

In this section, we formally define the problem of relational reasoning in a knowledge graph and provide an overview of our RL solution. We then detail our use of entity type embeddings and a heterogeneous neighbor encoder. An overview of the model is displayed in Fig. 2.

3.1. Problem formulation

Knowledge graphs consist of facts represented as triples. We formally define a knowledge graph $\mathcal{G} = \{(e_s, r, e_d)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ is a set of entities and $\mathcal{R}$ is a set of relations. Given a query $(e_s, r, ?)$, e_s is called the source entity and r is the query relation. Our goal is to predict the target entity $e_d \in \mathcal{E}$. In most cases, the output of each query is a list of candidate entities, $\hat{E}_d = \{\hat{e_1}, \dots, \hat{e_n}\}$ for some fixed $n < |\mathcal{E}|$, ranked in descending order by probability. The prediction can be represented as a function $\mathcal{F} : \mathcal{E} \times \mathcal{R} \to \mathcal{E}^n$. We assume that our knowledge graph is static and no new entities or relations will be added in the future.

In this work, we are not only interested in accurate prediction of the target entity e_d, but also in understanding the reasoning path the model uses to predict e_d. This is a key advantage that RL methods offer over embedding-based models which are able to perform relation predictions but cannot give interpretable justification for them. Rather than treating this task as a form of link prediction, RL models instead train an agent to traverse the nodes of a knowledge graph via logical reasoning paths. We provide the details of our problem formulation in the following. Algorithm 1 describes the workflow of HRRL.

3.2. A reinforcement learning solution

Similar to Das et al. [6], Lin et al. [22] and Xiong et al. [5], we formulate this problem as a Markov Decision Process (MDP), in which the goal is to train a policy gradient agent (using REINFORCE [50]) to learn an optimal reasoning path to answer a given query $(e_s, r, ?)$. We denote the RL framework as a set of states, actions, rewards, and transitions.

States. The state s_t at time t is represented as tuple $((e_s, r), e_t, h_t)$, where (e_s, r) is the input query, e_t is the entity at which the agent is located at time t and h_t is the history of the entities and relations traversed by agent until time t. The agent begins at the source entity with initial state $s_0 = ((e_s, r), e_s, h_0)$. We refer to the terminal state as $s_T = ((e_s, r), e_T, h_T)$, where e_T is the agent's answer to the input query and h_T is the full reasoning path. Each entity and relation is represented by an embedding vector. In our solution, we enrich the state representation with entity type and neighborhood information, which we describe in Section 3.3.

Actions. At each time step, the agent performs an action by either traversing an edge to a neighboring entity, or staying at the current entity. The action space $A_t \subseteq A$ given state s_t is thus the set of all neighbors of the current node e_t, and the node itself, i.e., $A_t(s_t) = \mathcal{N}_{e_t} \cup \{e_t\}$, where $\mathcal{N}_e$ is the set of all neighbors from node e. The inclusion of the current node e_t in the action space represents the agent's decision to terminate and select e_t as its answer to the input query. Additionally, the graph is directed, so $\mathcal{N}$ only includes nodes adjacent on out-edges. Following previous work [5,6,51], for each edge (triple) (e_s, r, e_d), we add an inverse edge (e_d, r^{-1}, e_s) in order to facilitate graph traversal.

In most real-world knowledge graphs, a small percentage of entities have a large degree while the majority of the entities have a small degree. However, the entities with a high degree are crucial to query answering. For performance reasons, many RL models are forced to cap the size of the action space and do so via a pre-computed heuristic. For example, Lin et al. [22] pre-computes PageRank scores for each node, and narrows the action space to a fixed number of highest-ranking neighbors. In this work, we use entity type information to constrain the search to the entities with best-matching types, given the previous actions. We provide more details in Section 3.3.

Rewards. The agent evaluates the quality of an action based on the expected reward it will produce. Previous works [5,6] define a terminal reward of +1 only if the agent reaches the correct answer. However, since knowledge graphs are incomplete, a binary reward cannot model the potentially missing facts. As a result, the agent receives low-quality rewards as it explores the environment. Inspired by Ng et al. [52] and Lin et al. [22], we use pre-trained knowledge graph embeddings for a soft reward function for the terminal state s_T:

$$R_T(s_T) = \begin{cases} 1 & \text{if } (e_s, r, e_T) \in \mathcal{G} \\ S(e_s, r, e_T) & \text{otherwise} \end{cases} \tag{1}$$

In other words, if e_T is the correct answer, the agent receives the positive binary reward. Otherwise, the reward is calculated based on $S(e_s, r, e_T)$, in which is S a fact score function based on pre-trained embeddings. We represent S in a generic form, and it can be replaced by an embedding model such as [14,17]. We explore a variety of embedding methods on multiple datasets in our experiments. More details are provided in Section 4.

Transitions. At state s_t, the agent chooses an action $a_t \in A_t$ based on a policy $\pi : \mathcal{S} \to \mathcal{A}$ where $\mathcal{S}$ and $\mathcal{A}$ are the sets of all possible states and actions, respectively. The transition function $\delta : S \times A \to S$ is defined by $\delta(s_t, A_t) = \delta(e_t, e_s, r, A_t)$.

3.2.1. Policy network

Following [22], we use an LSTM to encode the history $h_t = \{e_{t-k}, r_{t-k+1}, \dots, e_{t-1}, r_t\}$ of the past k steps taken by the agent in solving the query. The history embedding for h_t is represented as:

$$\mathbf{h_t} = LSTM(h_{t-1}, a_{t-1}). \tag{2}$$

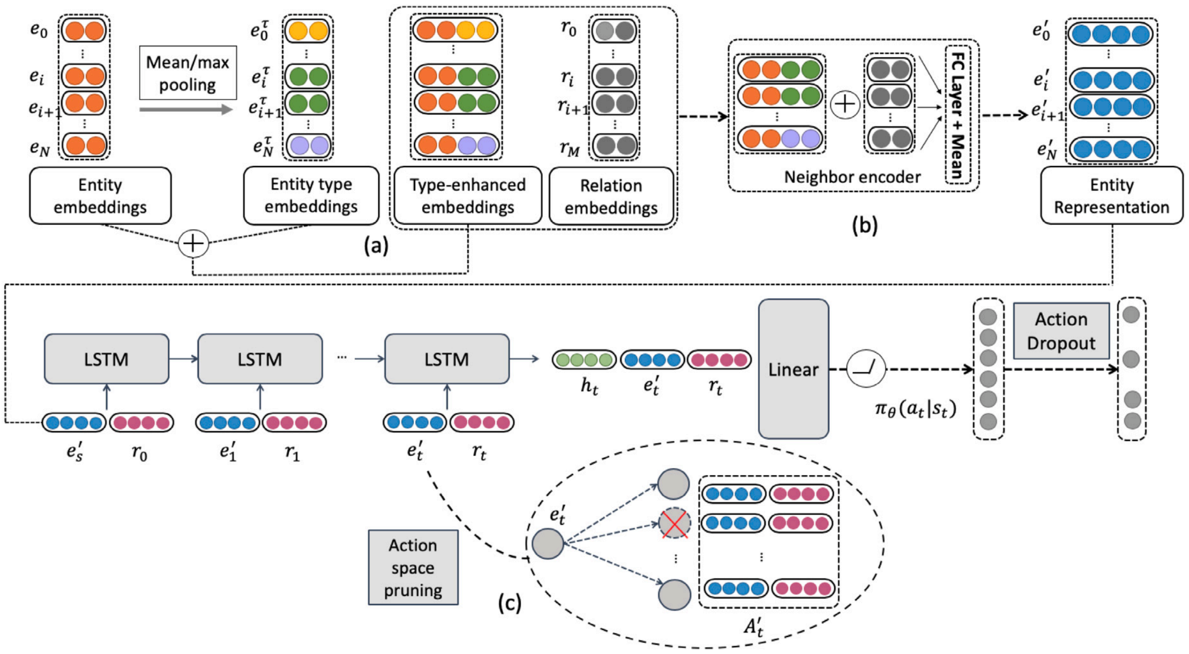

Fig. 2. Model overview. (a) The type embeddings are first created by max/mean pooling on the entities with a similar type. The type embeddings are then concatenated with the entity embeddings to create the type-enhanced embeddings. (b) The type-enhanced embeddings are then passed to the neighbor encoder to create the final entity representation fed to RL. (c) The action-space is pruned using the type-enhanced entity embeddings.

We define the policy network π with weight parameters θ as follows:

$$\pi_\theta(a_t|s_t) = \text{softmax}(A'_t \times W_2 ReLU(W_1[e_t \oplus h_t \oplus r])), \tag{3}$$

where $\oplus$ represents the vector concatenation operator. The transition to a new state is thus given by:

$$s_{t+1} = ((e_s, r), \underset{a'_t \in A'_t}{\text{argmax}}\, \pi_\theta(a'_t|s_t)). \tag{4}$$

To reduce the potential impact of argmax leading to the overuse of incorrect paths, we utilize random action dropout as described in [22].

3.2.2. Optimization

The policy network is trained to maximize the expected reward over all queries. To encourage path diversity, following [6], we add an entropy regularization term to the cost function scaled by a constant factor.

$$J(\theta) = \mathbb{E}_{(e_s,r,e_d)\in G}\mathbb{E}_{a_1,\dots,a_T\sim\pi_\theta}[R_T(s_T) + \beta H(\pi_\theta(s_T))], \tag{5}$$

where $H(\pi_\theta(s_T)) = \sum_a \pi_\theta(a|s_t)\log \pi_\theta(a|s_t)$ is the entropy term.

We use REINFORCE by Williams [50] to solve the optimization problem over all queries in the training data and update θ using the following gradient (refer to Ahmed et al. [53] for further details):

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t [R(s_T|e_s, r) + \beta H(\pi_\theta(s_T))]\pi_\theta(a_t|s_t). \tag{6}$$

3.3. Type-enhanced entity representation

Many knowledge graphs contain rich heterogeneous context information as *entity types* that can be used as prior information to guide the agent through the reasoning process. We argue that the type information can help reduce the action space, especially for nodes with a high degree, by constraining the search only to the entities that best match the previously visited entities and actions. To achieve this, we measure the similarity of all possible actions given the entity type embedding of current and possible target entities and only keep the top n candidates. In order to build the entity type representation e^τ, we propose to aggregate the vector representation of the entities with a similar type. Below we propose two simple mechanisms for doing so:

1. Take the average of the embedding vectors for all the N entities e_i that share the same entity type τ (mean-pooling).
2. Take the maximum value of each element in the embedding vectors for all the N entities e_i that share the same entity type (max-pooling).

To measure the similarity of the current entity representation e_t with the candidate neighboring entity representation e_k, we use the cosine similarity of two entity embeddings with respect to their type-enhanced embedding vectors:

$$g(e_t, e_k) = \left\langle [e_t \oplus e_t{}^\tau] + r_{t,k}, [e_k \oplus e_k{}^\tau] \right\rangle, \tag{7}$$

where $\langle\cdot,\cdot\rangle$ is the dot product operation and $e, r \in \mathbb{R}^d$ are d-dimensional vector representations of the entity e and relation r. We call $e_t{}^{\tau'} = [e_t \oplus e_t{}^\tau]$ the type-enhanced entity representation of entity e_t. We then rank the possible actions and prioritizes the ones that are more likely to result in a correct answer based on the $g(e_t, e_k)$ score. We thus create a pruned action space A'_t by keeping the nodes with the highest value of g.

3.4. Heterogeneous neighbor encoder

After generating the type embeddings, we feed the type-enhanced embeddings together with the relation embeddings to the heterogeneous neighbor encoder to generate the enriched entity representation. Although many works such as [51,54] have been proposed to learn entity embeddings using relational information, recent studies such as [30,31] have demonstrated that explicitly encoding neighborhood structure can benefit entity embedding learning in knowledge graphs. Inspired by this, we propose a heterogeneous neighbor encoder to learn the enriched entity embedding by aggregating entity's neighbors information. Specifically, we denote the set of relational neighbors $(relation, entity)$ of a given entity e_t as $\mathcal{N}_{e_t} = \left\{(r_k, e_k)|(e_t, r_{t,k}, e_k) \in \mathcal{G}\right\}$, where $\mathcal{G}$ is the background knowledge graph, $r_{t,k}$ and e_k represent the kth relation and the corresponding neighboring entity embedding of e_t, respectively. Note that we also include the entity e_t as its zero-hop neighbor in $\mathcal{N}_{e_t}$. The heterogeneous neighbor encoder should be able to encode $\mathcal{N}_{e_t}$ and output a feature representation of e_t by considering different relational neighbors $(r_{t,k}, e_k) \in \mathcal{N}_{e_t}$. To achieve this goal, we formulate the enriched entity embedding as follows:

$$f(e) = \sigma \left\{ \frac{1}{|\mathcal{N}_{e_t}|} \sum_k \left(\mathcal{W}_{rk}(r_{t,k} \oplus e_k^{\tau'}) + b_{rk} \right) \right\}, \tag{8}$$

where σ denotes activation unit (we use $tanh$), $e_k^{\tau'}$ is the type-enhanced entity embedding of e_k, and $r_{t,k} \in \mathbb{R}^{d\times 1}$ is the relation embedding. Further, $\mathcal{W}_{rk} \in \mathbb{R}^{d\times 2d}$ and $b_{rk} \in \mathbb{R}^{d\times 1}$ are parameters of neighbor encoder. The output of $f(e)$ is the entity representation that is seen by the RL agent as the state representation in Section 3.2.

3.5. Training algorithm

We present the HRRL algorithm as shown in Algorithm 1. We start by initializing the entity embeddings using pre-trained ConvE [17] and ComplEx [14] embeddings (depending on the dataset). We then calculate the type embeddings using max/mean pooling in line 3 and generate the type-enhanced embeddings in line 4. Lines 5–22 describe the training procedure for the RL algorithm. After initialization (lines 6–9), we update the entity representation based on its neighborhood (line 11). We then sample the action from the current policy according to line 12. If we select an invalid action, we revert to the previous entity. We terminate the search if we reach the correct answer or if we reach the maximum number of steps. Line 18 and 22 updates the policy for invalid actions and valid actions, respectively. The training stops once we reach a given number of epochs.

Algorithm 1 HRRL algorithm

```
1:  Initialize pre-trained entity embeddings
2:  for each entity type do
3:      Calculate the type-embeddings by mean/max pooling
4:      Generate type-enhanced embeddings using Eq. (7)
5:  for $episode \leftarrow 1$ to $N$ do
6:      Initialize LSTM's hidden state $h_0$ to 0
7:      Initialize GNN with random weights
8:      Initialize state vector $s_0 = ((e_s, r), e_0, h_0)$
9:      Initialize Num_Steps to 0
10:     while Num_Steps < Max_Steps do
11:         Update entity representation with Eq. (8)
12:         Randomly sample action (with action dropout) $a \sim \pi_\theta(a_t|s_t) \cdot$
            $\mathbf{m}$ (where $m_i \sim Bernoulli(1-\alpha), i = 1, \cdots, |A_t|$)
13:         if Action $a$ is invalid then
14:             Go back to the previous entity
15:             Increment Num_Steps
16:         if success or Num_Steps = Max_Steps then
17:             break
18:     Update $\theta$ using $g \propto \sum_{a\_} (-1) \log_\pi(a_t|s_t; \theta)$ for invalid actions.
19:     if success then
20:         $R_T(s_T) = 1$
21:     else $R_T(s_T) = f(e_s, r, e_T)$ (using Eq. (1))
22:         Update $\theta$ using Eq. (6) for valid actions.
```

4. Experiments

In this section, we describe and discuss the experimental results of our proposed approach. We compare against several baseline methods: ConvE (embedding-based) [17], ComplEx (embedding-based) [14], DistMult [16] (embedding-based), MINERVA (agent-based) [6], and Lin et al. (agent-based) [22]. We also tried different variations of our model by removing different model components. The type-enhanced embeddings are removed in HRRL(-T) and the heterogeneous neighbor encoder is removed in HRRL(-N).

4.1. Data & metrics

The experiments utilize three datasets presented in Table 1. Among the standard datasets used in the relational reasoning task, NELL-995 and FB15k-237 are the only ones that explicitly encode entity types. Therefore, in addition to NELL-995 and FB15k-237, we incorporated two datasets from the Amazon e-commerce collection [55]. Each Amazon data contains a set of users, products, brands, and other information, which the authors of Xian et al. [25] use to make product recommendations to users. Their task is a specialized instance of knowledge graph reasoning that only focuses on user-product relations, so we do not include it in our baseline results. Additionally, we found these datasets were too large for efficient computation in the broader knowledge graph reasoning task, so we shrunk them for our experiments. To do this, we discarded all entities of type "RelatedWord", then induced a subgraph on a random selection of 20% of the remaining nodes. While this might result in a sparser graph that makes predictions more difficult, this was the best option given the lack of other relevant data containing type information. For FB15k-237 data, we follow a similar approach as [29] to extract the type information.

The full knowledge graph is represented by the number of $Facts$ in Table 1. Before training, we partition $Facts$ into a training set and a test set, which we call $Queries$. In NELL-995, this split already exists as part of the standard dataset. For Amazon datasets, we populate $Queries$ with 2.5% of the triples in $Facts$, chosen at random. Each model is then trained on the set $(Facts{-}Queries)$, and tested on the set $Queries$. Recall that each fact is a triple in the form (e_s, r, e_d). Each triple is presented to the model in the form $(e_s, r, ?)$, and, as described in Section 3.1, the model outputs a list of ranked candidate entities $\hat{E_d} = \{e_1, \dots, e_n\}$. Also recall that we describe the prediction as a function $\mathcal{F} : (e_s, r) \rightarrow \hat{E_d}$.

We measure performance for each experiment with the standard knowledge graph reasoning metrics, namely, Hits@k for k={1,5,10}, Mean Reciprocal Rank (MRR), and Hit Ratio. Hits@k is measured as the percentage of test cases in which the correct entity e_d appears in the top k candidates in $\hat{E_d}$, i.e.,

$$Hits@k = \frac{|\{(e_s, r, e_d) \in Q : rank(e_d, \mathcal{F}(e_s, r)) \leq k\}|}{|Q|} \times 100 \tag{9}$$

where $Q = Queries$ and $rank(e_d, \hat{E_d})$ is a function that returns the position of entity e_d in the set of ordered predictions $\hat{E_d}$. MRR is a related metric, defined as the multiplicative inverse of the rank of the correct answer, i,e.:

$$MRR = \frac{1}{|Q|} \sum_{(e_s, r, e_d) \in Q} \frac{1}{e_d, \mathcal{F}(e_s, r)} \times 100. \tag{10}$$

We also measure *Hit Ratio*, which is equivalent to the average obtained reward over all the queries:

$$HitRatio = \frac{1}{|Q|} \sum_{(e_s, r, e_d) \in Q} R(e_s, r). \tag{11}$$

Because none of these models generalize to unknown entities, followed by previous works [6,22], we measure Hits@k and MRR only for queries for which both e_s and e_d have already been seen at least once by the model during training. In other words, if either of the query entities is missing from the training set $(Facts - Queries)$, we discard it from testing. Additionally, we reserve a small portion of the $Facts$ as a development set to estimate performance during training.

4.2. Parameter selection

For NELL-995 and FB15k-237 datasets, we utilize the same hyperparameters described in [22] when training ConvE, ComplEx, Distmult, and Lin et al. [22] baselines. For MINERVA, we utilize the same hyperparameters described in [6] and train the model for 3000 epochs. For the two Amazon datasets, we perform a grid search for our method and all baselines and report the best performance for each. For all

Table 1
Description of the datasets used for our experiments. $\mathcal{E}$ is the set of nodes in the knowledge graph, Types is the number of entity types, $\mathcal{R}$ is the set of relations (edge types), and Facts is the set of all edges. Queries is the test set, a subset of Facts which are removed from the knowledge graph for testing. Queries Discarded is the subset of Queries for which at least one of the entities does not appear in the training set.

Dataset	$\|\mathcal{E}\|$	$\|Types\|$	$\|\mathcal{R}\|$	$\|Facts\|$	$\|Queries\|$	$\|QueriesDiscarded\|$
NELL-995	75,492	268	200	154,213	3,992	1152
Amazon beauty	16,345	5	7	52,516	1,325	174
Amazon cellphones	13,837	5	7	31,034	951	205
FB15k-237	14,505	182	237	272,115	20,466	28

Table 2
Experimental results on NELL-995, FB15k-237, Amazon Beauty, and Amazon Cellphones datasets. @{1, 5, 10} and MRR are standard knowledge graph reasoning metrics and are described in Section 4.1. The methods are separated into embedding-based (ConvE, ComplEx, and DistMult) and agent-based (MINERVA, Lin et al. and HRRL) groups. Bolded numbers indicate the best-performing method of the RL-based group and underlined numbers indicate the best-performing method of the embedding-based group.

Dataset	NELL-995			FB15k-237			Amazon beauty			Amazon cellphones		
Metric (%)	@1	@10	MRR	@1	@10	MRR	@1	@10	MRR	@1	@10	MRR
ConvE [17]	68.2	88.6	76.1	34.1	62.2	43.5	25.8	55.3	35.2	16.9	44.8	25.7
ComplEx [14]	63.0	86.0	71.8	32.8	61.6	42.5	27.6	58.0	37.5	17.4	45.2	26.6
DistMult [16]	65.1	85.7	73.4	32.4	60.0	41.7	26.7	58.1	37.1	18.8	48.4	28.8
Lin et al. [22]	65.6	84.4	72.7	32.7	56.4	40.7	20.6	39.5	27.1	12.2	27.6	17.5
MINERVA [6]	59.8	82.1	68.9	21.7	45.6	29.3	17.5	38.2	24.3	6.8	22.7	11.6
HRRL(-T)	66.9	85.2	74.1	33.6	57.8	41.9	21.2	40.5	27.9	12.6	28.1	17.9
HRRL(-N)	67.1	85.1	73.1	32.9	57.2	41.2	20.7	39.6	27.2	12.3	27.9	17.6
HRRL	**68.9**	**86.7**	**74.8**	**34.0**	**58.1**	**42.1**	**21.8**	**40.7**	**28.2**	**12.9**	**28.5**	**18.2**

datasets, we train the knowledge graph embedding models (ConvE and ComplEx) for 1000 epochs each. These embeddings are then used to make predictions directly but also serve as pre-trained inputs for the RL agent, which we train for 30 epochs per experiment for all datasets. We initially test with different embedding methods for the pre-trained embeddings and settled on those that achieved the best performance: ComplEx for NELL-995 and FB15k-237 and Distmult for both Amazon datasets.

Regarding the HRRL hyperparameters, we set entity, relation, and history embedding dimensions to 200. We set the batch size to 100 and the learning rate to e^{-3}. We tune the action drop-out ratio between 0.1 and 0.9. We set a maximum limit of 200 for the action-space pruning. We use a two-layer LSTM as the path encoder and set its hidden dimension to 200. We use Adam [56] to optimize the RL agent.

5. Experimental results

Our experimental results are described in Table 2. For NELL-995 and FB15k-237 data, We quote the results reported in [6,22]. Embedding-based methods show an overall better performance compared to the RL-based methods. We can see that in all three datasets, our results outperform both RL baselines ([22] and MINERVA [6]). FB15k-237 and Amazon datasets, on the other hand, are far more challenging. We notice that even the embedding-based methods are struggling with low performance on these datasets. On Amazon data, the performance of all methods is significantly lower. Our method results in a 4% improvement in MRR (and 5.43% in Hits@1) over the best RL baseline on Amazon Cellphones and a 4% improvement in MRR (and 5.8% in Hits@1) on Amazon Beauty. On NELL-995 dataset, our method results in 2.9% improvement in MRR and 4.7% improvement in Hits@1 over the best performing baseline. We achieve 3.4% improvement in MRR and 4% improvement in Hits@1 over [22] on FB15k-237 data. We also perform ablation studies to analyze the effect of each module in our model. We notice that removing the heterogeneous neighbor encoder results in a higher drop in performance in FB15k-237 and Amazon datasets. This gap is relatively smaller in NELL-955 data.

Our results show that pruning the action space based on the entity type information results in a larger boost in performance on Amazon datasets. We believe due to the sparsity of these two knowledge graphs, type information is more effective for action space pruning than entity page rank (as done in [22]). Note that there are only 5 entity types in Amazon datasets. As a result, the number of entities that can potentially get discarded due to type mismatch is higher, and this assists the agent to discover a better path. We generate the type embeddings using max-pooling for NELL-995 dataset and mean-pooling for both Amazon datasets. We analyze the speedup of our method with different pruning strategies in Section 5.2.

5.1. Path diversity and convergence

In order to show the explorative power of our RL agent, we compare the number of unique paths discovered from the development set during the training procedure. Fig. 3 shows that path diversity (top row) improves across all models as the model performance (bottom row) improves. For this analysis, we compare our ablation models (HRRL(-N) and HRRL(-T)) with the best performing RL baseline by Lin et al. [22]. Our method is more successful in discovering novel paths and obtains a better hit ratio on the development set. On Amazon Beauty data, the number of unique paths discovered by HRRL(-T) is higher than both combined (HRRL) while in Amazon Cellphones the combined model performs better, but similar to Amazon Beauty, HRRL(-T) performs better than HRRL(-N). NELL-955 and FB15k-237 show a different trend, where removing the type information results in a larger drop in the number of unique paths, compared to the heterogeneous neighbor encoder. This is expected since NELL-955 and FB15k-237 contain far more entity types than Amazon datasets, and the inclusion of type information may be a positive factor for discovering new paths. In terms of convergence, Amazon Beauty and Amazon Cellphones show a similar trend and removing the type information significantly reduces the hit ratio. This gap is smaller for NELL-995 and FB15k-237 data, though our model still shows improvement in hit ratio on these datasets. For FB15k-237 data, all our ablation models achieve a higher hit ratio in the earlier epochs, but as the number of epochs increases the hit ratio obtained by Lin et al. [22] reaches HRRL(-T).

5.2. Benefit of using the heterogeneous type information

We found that using type information for pruning the action space is a better strategy compared to using page rank. Although our model is more complex than [22] due to incorporating GNN, we were able to achieve relatively same run time using a smaller action space. This is due to the fact that using heterogeneous type information can result in

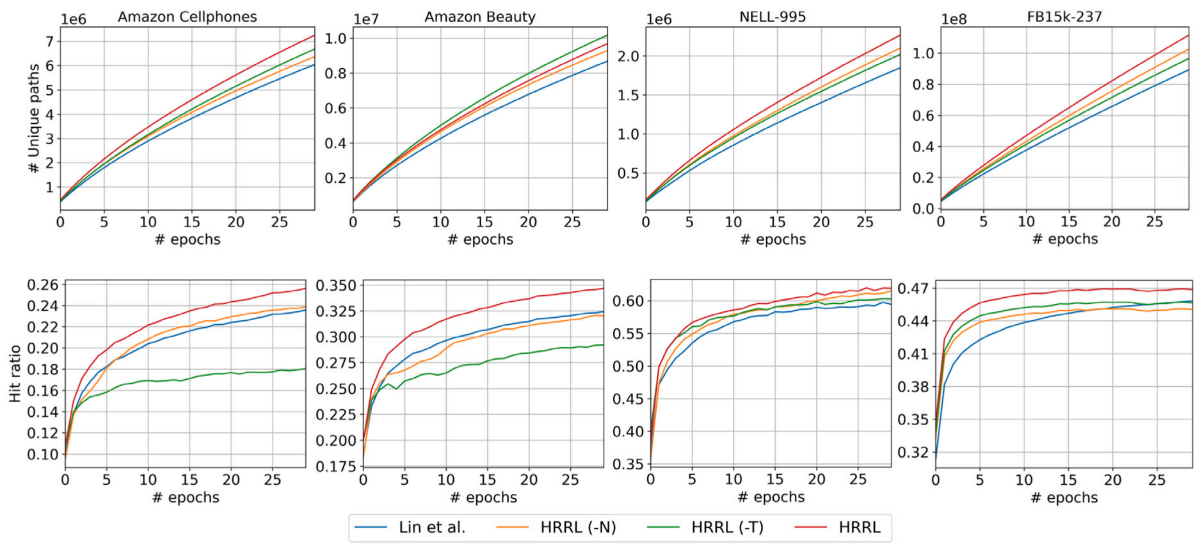

Fig. 3. Performance of our model on the development set at different training epochs. The top figures show the number of unique paths visited during each epoch. The bottom rows show the hit ratio for the entire development set. For the top row, all the numbers on the y-axis should be multiplied by the number on the top left corner of the figure.

Table 3
Running time comparison for HRRL (with page rank pruning and entity type-based pruning) and other RL-based baselines. All values are reported in minutes.

Dataset	NELL-995	FB15k-237	Amazon beauty	Amazon cellphones
Lin et al. [22]	225.21	2255.84	1145.24	926.55
MINERVA [6]	92.48	495.01	376.23	285.37
HRRL (PR pruning)	315.29	3468.24	1639.53	1261.18
HRRL	163.95	2122.65	1138.45	947.04

more relevant candidates, which allows us to choose a smaller limit for the maximum action space. This is shown in Table 3. We can see that MINERVA [6] is the fastest RL method, but it is the lowest-performing RL baseline. HRRL (with page rank pruning) is a lot slower than [22], due to the addition of the heterogeneous neighbor encoder. Finally, we can see that HRRL with entity type-based pruning achieves a significant speedup since we were able to reduce the maximum action space size to 100, which is half of what was used in page rank pruning to achieve the same performance. The speedup is more significant in NELL-995 (51.7%) and FB15k-237 (61.2%) datasets compared to Amazon datasets, as NELL-995 and FB15k-237 data have a richer and more diverse heterogeneous type content, and contain a higher number of facts.

5.3. Performance on different relations

We evaluate our proposed model on different relation types and compare our results with the best performing RL baseline. We take a similar approach as [22] to extract *to-many* and *to-one* relations. A relation r is considered *to-many* if queries containing relation r can have more than 1 correct answer, otherwise, it is considered a *to-one* relation. Table 4 shows the MRR values on the development set for all three datasets. We notice that most relations in FB15k-237 and Amazon datasets are *to-many*, while a large portion of NELL-995 data consists of *to-one* relations. Overall, *to-many* relations show lower performance, regardless of the model. Our proposed model consistently shows a better performance than [22]. Both *to-one* and *to-many* are more sensitive to removing the neighbor-encoder rather than removing the type information. We also compared the ablation models on seen and unseen queries, and observed the same pattern as the one shown in Table 4. This is expected, since *to-many* relations are more likely to be among the seen queries and *to-one* relations are more likely to be an unseen query.

5.4. Additional case studies

In this section, we present a few case studies that show the strength of our proposed method. We first focus on NELL-995 dataset which has a high number of entity types. Table 5 shows top frequent path types discovered by HRRL and the best performing RL baseline [22] for a few example queries in NELL-995 data. In this table p_1 and p_2 show the occurrence probability for each path type discovered by HRRL and [22], respectively. We notice that our method is more successful in discovering better paths as it takes advantage of the heterogeneous content. As an example, for the query (Knicks, *team plays sport*, ?), our method discovers the path: Knicks (sports team) [$\underline{67}$] $\xrightarrow{team\ plays\ against\ team}$ Trail Blazers (sports team) [$\underline{30}$] $\xrightarrow{athlete\ plays\ for\ team^{-1}}$ Steve Blake (athlete) [4] $\xrightarrow{plays\ sport}$ basketball (sport) [$\underline{242}$], in which the underlined number in the bracket shows the entity's degree. This path matches the first row in Table 5 for this query, which has the highest probability of selection by HRRL. Lin et al. method, on the other hand, is more likely to select the second path type, which cannot reach the correct answer.

As another example, we consider the query: (Amazon, *company sector*, ?) for which our method discovers the path: Amazon (company) [$\underline{4}$] $\xrightarrow{acquired}$ AbeBooks (company) [$\underline{2}$] $\xrightarrow{company\ sector}$ internet (economic sector) [$\underline{26}$]. Therefore, HRRL is able to infer the company sector using the sector of the company that was acquired by Amazon. Table 5 shows the other possible path types. We can see that Lin et al. method has a higher probability of selecting an incorrect path which leads to a (city) entity, as opposed to an (economic sector) entity.

The third query in Table 5 shows the top path types for query: (CNN, *journalist works for*$^{-1}$, ?). For this query, HRRL correctly navigates to the answer using the path: CNN (company) [$\underline{46}$] $\xrightarrow{journalist\ works\ for^{-1}}$ Kyra Phillips (journalist) [$\underline{2}$] $\xrightarrow{journalist\ works\ for}$ CNN (company) [$\underline{46}$] $\xrightarrow{journalist\ works\ for^{-1}}$ Anderson Cooper (journalist) [$\underline{2}$]. In this case, the (company) entity has a relatively higher degree, resulting in additional complexity for selecting the next action. However, by leveraging the heterogeneous context, HRRL is more likely to select a (journalist) entity rather than a (city) entity or a (TV station) entity after reaching the (company) entity.

Finally, for the query: (MSU, *located in state*, ?), HRRL discovers the following path: MSU (university) [$\underline{5}$] $\xrightarrow{person\ belongs\ to\ org^{-1}}$ Tom Izzo (coach) [$\underline{2}$] $\xrightarrow{works\ for}$ Michigan State (sports team) [$\underline{23}$] $\xrightarrow{located\ in\ state}$ Michigan (state) [$\underline{137}$], corresponding to the first row for the fourth

Table 4
MRR on three datasets for different relation types. The percentage of one-to-one and one-to-many relations in the development set for each data is shown in the % column. The numbers in parenthesis indicate the relative performance change compared to the best results.

Dataset	NELL-995		FB15k-237		Amazon beauty		Amazon cellphones	
	To-Many	To-One	To-Many	To-One	To-Many	To-One	To-Many	To-One
Percentage	12.9%	87.1%	76.6%	23.4%	89.6%	10.4%	95.5%	4.5%
Lin et al.	55.7	81.4	28.3	72	24.2	36.8	16.5	78.2
HRRL	**57.4**	**82.8**	**29.2**	**73.4**	**25.5**	**39.1**	**17.8**	**83.4**
HRRL(-T)	56.9	82.0	28.9	72.8	23.8	39.2	15.9	**83.4**
HRRL(-N)	55.2	81.0	28.1	71.7	21.7	33.5	13.6	66.6

Table 5
Top 3 frequent paths for our method ($P1$) and Lin et al. ($P2$) along with the occurrence probability. Note that for each query, the number of all possible path types can vary between 73–277 in these examples. As a result, the probability of each path type is low.

Query	Reasoning path types	$p_1(\%)$	$p_2(\%)$
HRRL path:	Knicks [$\underline{67}$] $\xrightarrow{team\ plays\ against\ team}$ Trail Blazers [$\underline{30}$] $\xrightarrow{athlete\ plays\ for\ team^{-1}}$ Steve Blake [4] $\xrightarrow{plays\ sport}$ basketball [$\underline{242}$]		
e_s: Knicks	sports team $\xrightarrow{team\ plays\ against\ team}$ sportsteam $\xrightarrow{athlete\ plays\ for\ team^{-1}}$ athlete $\xrightarrow{plays\ sport}$ sport	5.92	3.19
r: *plays sport*	sports team $\xrightarrow{team\ home\ stadium}$ event venue $\xrightarrow{sport\ uses\ stadium^{-1}}$ sport $\xrightarrow{plays\ sport^{-1}}$ sports team	5.77	5.19
e_t: ?	sports team $\xrightarrow{sports\ game\ team^{-1}}$ sports game $\xrightarrow{sports\ game\ team}$ sports team	3.79	2.28
HRRL path:	Amazon [$\underline{4}$] $\xrightarrow{acquired}$ AbeBooks [$\underline{2}$] $\xrightarrow{company\ sector}$ internet [$\underline{26}$]		
e_s: Amazon	company $\xrightarrow{acquired}$ company $\xrightarrow{company\ sector}$ economic sector	4.88	3.52
r: *company*	company $\xrightarrow{has\ office\ in\ city}$ city $\xrightarrow{has\ office\ in\ city^{-1}}$ company $\xrightarrow{has\ office\ in\ city}$ city	3.71	4.01
sector	company $\xrightarrow{has\ office\ in\ city}$ city $\xrightarrow{has\ office\ in\ city^{-1}}$ newspaper $\xrightarrow{journalist\ works\ for^{-1}}$ journalist	2.93	2.32
e_t: ?			
HRRL path:	MSU [$\underline{5}$] $\xrightarrow{person\ belongs\ to\ org^{-1}}$ Tom Izzo [$\underline{2}$] $\xrightarrow{works\ for}$ Michigan State [$\underline{23}$] $\xrightarrow{located\ in\ state}$ Michigan [$\underline{137}$]		
e_s: MSU	university $\xrightarrow{person\ belongs\ to\ org^{-1}}$ coach $\xrightarrow{works\ for}$ sports team $\xrightarrow{located\ in\ state}$ state	22.65	7.41
r: *located in*	university $\xrightarrow{person\ belongs\ to\ org^{-1}}$ coach $\xrightarrow{works\ for}$ sports team $\xrightarrow{team\ plays\ against\ team}$ sports team	10.21	8.64
state	university $\xrightarrow{is\ acronym\ for}$ university $\xrightarrow{language\ of\ university^{-1}}$ language $\xrightarrow{language\ of\ country}$ country	7.52	28.4
e_t: ?			

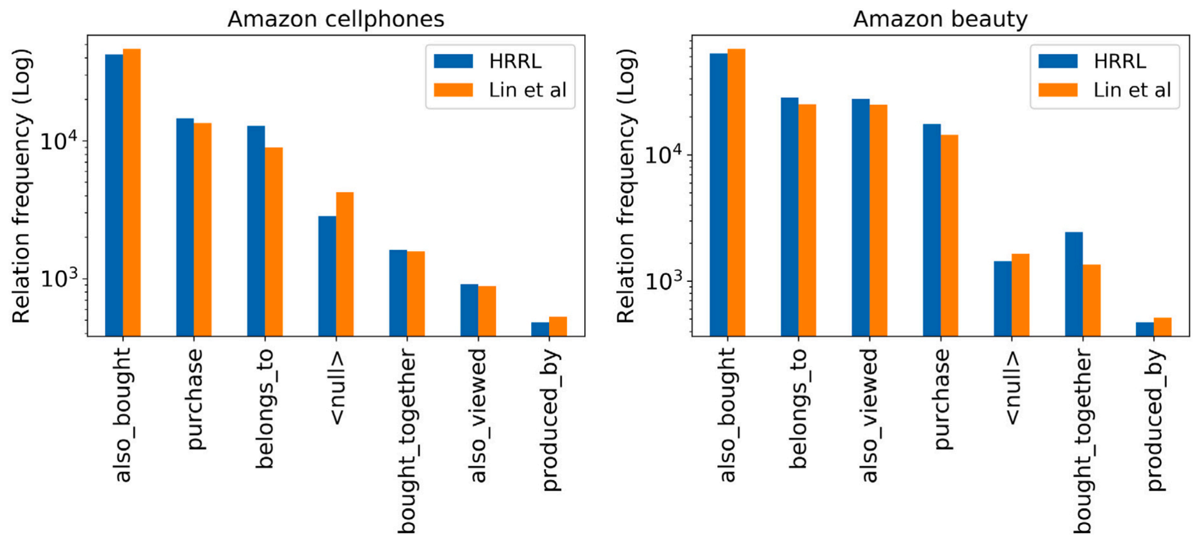

Fig. 4. Relation frequencies in the discovered paths of the development set for Amazon datasets. Each item on the x-axis is a relation, and the y-axis shows, in log-scale, the number of discovered paths that include that relation.

query in Table 5. Lin et al. method on the other hand, is more likely to select a path type that leads to a (country) or a (sport team) entity, as opposed to a (state) entity. For this query, number of all possible path types is smaller (75 for HRRL and 73 for [22]). As a result, the top frequent paths have relatively higher probabilities.

In Amazon datasets, there are fewer entity and relation types. As a result, we observe several frequent path types that can lead to the correct answer which all RL baselines are able to discover with similar probabilities. Therefore, we focus on the diversity of the relations used in our method and the best performing baseline [22] for the discovered paths in the development set. Fig. 4 displays the inference results. On Amazon cellphones data, our method uses fewer *<null>*, *produced-by* and *also-bought* relations while it utilizes more of other relations, in particular, *belongs-to* relations. Similarly, on Amazon Beauty data, our method utilizes fewer *also-bought*, *produced-by* and *<null>* relations, while it uses other relations more frequently, especially the *bought-together* relation. We believe one reason for the success of our method is the diverse use of different relation types for discovering new path types.

6. Conclusion

We proposed HRRL for improving the performance of path-based reasoning using RL. HRRL addresses the key challenges stemming from large action space and accurate representation of an entity's heterogeneous neighborhood. The key contributions of HRRL include an efficient vector representation for heterogeneous entity type-embeddings, pruning the action space for improving the choice of next actions, and leveraging a GNN for incorporating the neighborhood information.

We evaluate HRRL on four contemporary datasets. Our results show that incorporating information about the heterogeneous neighborhood results in improved performance for the query answering task. We show that the type information is important for faster convergence and finding more diverse paths, and the neighborhood information improves the performance on *to-many* relations. Despite these improvements, HRLL still requires a longer training time than embedding-based methods. As part of our future work, we plan to explore more efficient strategies for action-space pruning and improving GPU performance (such as [57]) to improve the scalability of existing RL solutions. Furthermore, we plan to develop more effective type-embeddings that models the hierarchical structure of entity types. Finally, HRRL is unable to handle scenarios in which new entities are added to the knowledge graph dynamically. As a result, we plan to adapt our framework for inductive question answering to address such scenarios.

CRediT authorship contribution statement

Mandana Saebi: Conceptualization, Methodology, Software, Writing – review & editing. **Steven Kreig:** Conceptualization, Data curation, Software, Writing – review & editing. **Chuxu Zhang:** Methodology, Software, Writing – review & editing. **Meng Jiang:** Conceptualization, Supervision, Writing – review & editing. **Tomasz Kajdanowicz:** Writing – review & editing. **Nitesh V. Chawla:** Conceptualization, Supervision, Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported in part by National Science Foundation, USA Grants IIS-1447795 and CRI-1629914, the Army Research Laboratory, USA under Cooperative Agreement no. W911NF-09-2-0053 (the ARL Network Science CTA), and by the National Science Centre, Poland under the research project no. 2016/23/B/ST6/01735 and 2016/21/D/ST6/02948. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Funding

This work is based on research supported in part by the University of Notre Dame Office of Research via Environmental Change Initiative (ECI) and NSF awards EF-1427157 and IIS-1447795; the research was also supported in part by the Army Research Laboratory, USA under Cooperative Agreement no. W911NF-09-2-0053 [the ARL Network Science CTA (Collaborative Technology Alliance)] and the National Science Centre, Poland under the research project no. 2016/23/B/ST6/01735 and 2016/21/D/ST6/02948.

References

[1] D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, D. Heckerman, C. Meek, et al., Introduction to Statistical Relational Learning, MIT Press, 2007.
[2] S. Muggleton, Inductive logic programming, New Gener. Comput. 8 (4) (1991) 295–318.
[3] C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, N. Ueda, Learning systems of concepts with an infinite relational model, in: Proceedings of AAAI Conference on Artificial Intelligence, 2006, pp. 381–388.
[4] Z. Xu, V. Tresp, K. Yu, H.-P. Kriegel, Infinite hidden relational models, 2012, arXiv preprint arXiv:1206.6864.
[5] W. Xiong, T. Hoang, W.Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, 2017, arXiv preprint arXiv:1707.06690.
[6] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, A. McCallum, Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning, 2017, arXiv preprint arXiv:1711.05851.
[7] I.V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N.R. Ke, et al., A deep reinforcement learning chatbot, 2017, arXiv preprint arXiv:1709.02349.
[8] B. Peng, X. Li, L. Li, J. Gao, A. Celikyilmaz, S. Lee, K.-F. Wong, Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning, 2017, arXiv preprint arXiv:1704.03084.
[9] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, DRN: A deep reinforcement learning framework for news recommendation, in: Proceedings of World Wide Web Conference, 2018, pp. 167–176.
[10] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, Deep reinforcement learning based recommendation with explicit user-item interactions modeling, 2018, arXiv preprint arXiv:1810.12027.
[11] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, E.H. Chi, Top-k off-policy correction for a REINFORCE recommender system, in: Proceedings of ACM International Conference on Web Search and Data Mining, 2019, pp. 456–464.
[12] S. Ji, S. Pan, E. Cambria, P. Marttinen, S.Y. Philip, A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[13] R. Socher, D. Chen, C.D. Manning, A. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Proceedings of Conference on Neural Information Processing Systems, 2013, pp. 926–934.
[14] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: Proceedings of International Conference on Machine Learning, 2016, pp. 2071–2080.
[15] Y. Jia, Y. Wang, X. Jin, H. Lin, X. Cheng, Knowledge graph embedding: A locally and temporally adaptive translation-based approach, ACM Trans. Web 12 (2) (2018) 8.
[16] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: Proceedings of International Conference on Learning Representations, 2015.
[17] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Proceedings of AAAI Conference on Artificial Intelligence, 2018.
[18] N. Guan, D. Song, L. Liao, Knowledge graph embedding with concepts, Knowl.-Based Syst. 164 (2019) 38–44.
[19] Z. Li, H. Liu, Z. Zhang, T. Liu, N.N. Xiong, Learning knowledge graph embedding with heterogeneous relation attention networks, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[20] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo, Knowledge graph embedding for link prediction: A comparative analysis, ACM Trans. Knowl. Discov. Data (TKDD) 15 (2) (2021) 1–49.
[21] A. Sharma, K.D. Forbus, Graph-based reasoning and reinforcement learning for improving Q/A performance in large knowledge-based systems, in: 2010 AAAI Fall Symposium Series, 2010.
[22] X.V. Lin, R. Socher, C. Xiong, Multi-hop knowledge graph reasoning with reward shaping, in: Proceedings of Conference on Empirical Methods in Natural Language Processing, 2018, pp. 3243–3253.
[23] M. Qu, J. Tang, J. Han, Curriculum learning for heterogeneous star network embedding via deep reinforcement learning, in: Proceedings of ACM International Conference on Web Search and Data Mining, 2018, pp. 468–476.
[24] Y. Shen, J. Chen, P.-S. Huang, Y. Guo, J. Gao, M-walk: Learning to walk over graphs using monte carlo tree search, in: Proceedings of Conference on Neural Information Processing Systems, 2018, pp. 6786–6797.
[25] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 285–294.
[26] Y. Shen, N. Ding, H.-T. Zheng, Y. Li, M. Yang, Modeling relation paths for knowledge graph completion, IEEE Trans. Knowl. Data Eng. (2020).
[27] K. Lei, J. Zhang, Y. Xie, D. Wen, D. Chen, M. Yang, Y. Shen, Path-based reasoning with constrained type attention for knowledge graph completion, Neural Comput. Appl. (2019) 1–10.

[28] G. Wan, S. Pan, C. Gong, C. Zhou, G. Haffari, Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning, in: IJCAI, 2020, pp. 1926–1932.

[29] R. Xie, Z. Liu, M. Sun, et al., Representation learning of knowledge graphs with hierarchical types, in: IJCAI, 2016, pp. 2965–2971.

[30] W. Xiong, M. Yu, S. Chang, X. Guo, W.Y. Wang, One-shot relational learning for knowledge graphs, in: Proceedings of Conference on Empirical Methods in Natural Language Processing, 2018, pp. 1980–1990.

[31] C. Zhang, L. Yu, M. Saebi, M. Jiang, N. Chawla, Few-shot multi-hop relation reasoning over knowledge bases, in: Proceedings of Conference on Empirical Methods in Natural Language Processing: Findings, 2020, pp. 580–585.

[32] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of International Conference on Learning Representations, 2017.

[33] F. Yang, Z. Yang, W.W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: Proceedings of Conference on Neural Information Processing Systems, 2017, pp. 2319–2328.

[34] T. Rocktäschel, S. Riedel, End-to-end differentiable proving, in: Proceedings of Conference on Neural Information Processing Systems, 2017, pp. 3788–3800.

[35] Y. Liu, M. Hildebrandt, M. Joblin, M. Ringsquandl, R. Raissouni, V. Tresp, Neural multi-hop reasoning with logical rules on biomedical knowledge graphs, in: European Semantic Web Conference, Springer, 2021, pp. 375–391.

[36] H. Yao, C. Zhang, Y. Wei, M. Jiang, S. Wang, J. Huang, N.V. Chawla, Z. Li, Graph few-shot learning via knowledge transfer, in: Proceedings of AAAI Conference on Artificial Intelligence, 2020.

[37] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484.

[38] Y. Shen, J. Chen, P.-S. Huang, Y. Guo, J. Gao, ReinforceWalk: Learning to walk in graph with Monte Carlo tree search, in: Proceedings of ICLR Workshops, 2018.

[39] Z. Li, X. Jin, S. Guan, Y. Wang, X. Cheng, Path reasoning over knowledge graph: A multi-agent and reinforcement learning based method, in: Proceedings of ICDM Workshops, 2018, pp. 929–936.

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

[41] H. Wang, S. Li, R. Pan, M. Mao, Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning, in: Proceedings of Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 2623–2631.

[42] Q. Wang, Y. Hao, J. Cao, ADRL: An attention-based deep reinforcement learning framework for knowledge graph reasoning, Knowl.-Based Syst. (2020) 105910.

[43] R. Wang, B. Li, S. Hu, W. Du, M. Zhang, Knowledge graph embedding via graph attenuated attention networks, IEEE Access 8 (2019) 5212–5224.

[44] P. Tiwari, H. Zhu, H.M. Pandey, DAPath: Distance-aware knowledge graph reasoning based on deep reinforcement learning, Neural Netw. 135 (2021) 1–12.

[45] L. Bai, W. Yu, M. Chen, X. Ma, Multi-hop reasoning over paths in temporal knowledge graphs using reinforcement learning, Appl. Soft Comput. 103 (2021) 107144.

[46] J. Liao, X. Zhao, J. Tang, W. Zeng, Z. Tan, To hop or not, that is the question: Towards effective multi-hop reasoning over knowledge graphs, World Wide Web 24 (5) (2021) 1837–1856.

[47] Y. Dong, N.V. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 135–144.

[48] C. Zhang, Learning from heterogeneous networks: Methods and applications, in: Proceedings of ACM International Conference on Web Search and Data Mining, 2020, pp. 927–928.

[49] C. Zhang, D. Song, C. Huang, A. Swami, N.V. Chawla, Heterogeneous graph neural network, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019, pp. 793–803.

[50] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256.

[51] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of Conference on Neural Information Processing Systems, 2013, pp. 2787–2795.

[52] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: Proceedings of International Conference on Machine Learning, 1999, pp. 278–287.

[53] Z. Ahmed, N. Le Roux, M. Norouzi, D. Schuurmans, Understanding the impact of entropy on policy optimization, in: International Conference on Machine Learning, PMLR, 2019, pp. 151–160.

[54] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: Proceedings of International Conference on Learning Representations, 2015.

[55] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, in: Proceedings of World Wide Web Conference, 2016, pp. 507–517.

[56] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[57] H.-N. Tran, E. Cambria, A survey of graph processing on graphics processing units, J. Supercomput. 74 (5) (2018) 2086–2115.