NEURAL COMPOSITIONAL RULE LEARNING FOR KNOWLEDGE GRAPH REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning logic rules is critical to improve reasoning in KGs. This is due to their ability to provide logical interpretable explanations when used for predictions, as well as their ability to generalize to other tasks, domains, and data. While recent methods have been proposed to learn logic rules, the majority of these methods are either restricted by their computational complexity and cannot handle the large search space of large-scale KGs, or show poor generalization when exposed to data outside the training set. In this paper, we propose an end-to-end neural model for learning compositional logic rules called NCRL . NCRL detects the best compositional structure of a rule body, and breaks it into small compositions in order to infer the rule head. By recurrently merging compositions in the rule body with a recurrent attention unit, NCRL finally predicts a single rule head. Experimental results show that NCRL learns high-quality rules, as well as being generalizable. Specifically, we show that NCRL is scalable, efficient, and yields state-of-the-art results for link prediction on large-scale KGs. Moreover, we test NCRL for systematic generalization by learning to reason on small-scale observed graphs and evaluating on larger unseen ones.

1 INTRODUCTION

Knowledge Graphs (KGs) provide a structured representation of real-world facts (Ji et al., 2021), and they are remarkably useful in various applications (Graupmann et al., 2005; Lukovnikov et al., 2017; Xiong et al., 2017; Yih et al., 2015). Since KGs are usually incomplete, knowledge graph reasoning is a crucial problem on KGs, where the goal is to infer the missing knowledge using the observed facts. This paper investigates how to learn logic rules for KG reasoning. Learning logic rules is critical for reasoning tasks in KGs and has received recent attention. This is due to their ability to: (1) provide interpretable explanations when used for prediction, and (2) generalize to new tasks, domains, and data (Qu et al., 2020; Lu et al., 2022; Cheng et al., 2022). For example, in Figure 1, the learned rules can be used to infer new facts related to objects that are unobserved in the training stage.

Moreover, logic rules naturally have an interesting property - called *compositionality*: where the meaning of a whole logical expression is a function of the meanings of its parts and of the way they are combined (Hupkes et al., 2020). To concretely explain compositionality, let us consider the family relationships shown in Figure 2. In Fig. 2(a), we show that the rule (hasUncle $\leftarrow$ hasMother $\wedge$ hasMother $\wedge$ hasSon) forms a composition of smaller logical expressions, which can be expressed as a hierarchy, where predicates (i.e., relations) can be combined and replaced by another single predicate. For example, predicates hasMother and hasMother can be combined and replaced by predicate hasGrandma as shown in Figure 2(a). As such, by recursively combining predicates into a composition and reduce the composition into a single predicate, we can finally infer the rule head (i.e., hasUncle) from the rule body. While, there are various possible hierarchical trees to represent such rules, not all of them are valid given the observed relations in the KG. For example, in Figure 2(b), given a KG which only contains relations {hasMother, hasSon, hasGrandma, hasUncle}, it is possible to combine hasMother and hasSon first. However, there is no proper predicate to represent it in the KG. Therefore, learning a high-quality compositional structure for a given logical expression is critical for rule discovery, and it is the focus of our work.

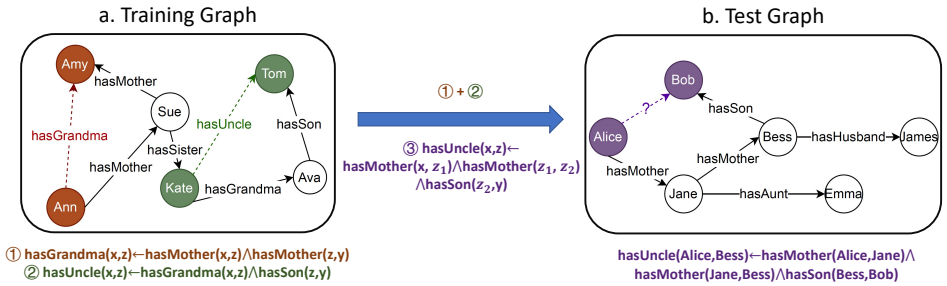

Figure 1: Illustration of how the compositionality of logical rules help improve systematic generalization. (a) logical rule extraction from observed graph (i.e., training stage) and (b) Inference on an unseen graph (i.e., test stage). The train and the test graphs have disjoint sets of entities. By combining logical rules ① and ② we can successfully learn rule ③ for prediction on unseen graphs.

In this work, our objective is to learn rules that generalize to large-scale tasks and unseen graphs. Let us consider the example shown in Figure 1. From the training KG, we can extract two underlying rules – rule ①: $\text{hasGrandma}(x, y) \leftarrow \text{hasMother}(x, z) \wedge \text{hasMother}(z, y)$ and rule ②: $\text{hasUncle}(x, y) \leftarrow \text{hasGrandma}(x, z) \wedge \text{hasSon}(z, y)$. We also observe that the necessary rule to infer the relation between *Alice* and *Bob* in the test KG is rule ③: $\text{hasUncle}(x, y) \leftarrow \text{hasMother}(x, z_1) \wedge \text{hasMother}(z_1, z_2) \wedge \text{hasSon}(z_2, y)$, which is not observed in the training graph. However, using compositionality to combine rules ① and ②, we can successfully learn rule ③ which is necessary for inferring the relation between *Alice* and *Bob* in the test KG. The successful prediction in the test KG shows the model's ability for systematic generalization (i.e., learning to reason on smaller graphs and making predictions on unseen graphs) (Sinha et al., 2019).

Although compositionality is crucial for learning logic rules, most of existing logic rule learning methods fail to exploit it. In traditional AI, inductive Logic Programming (ILP) (Muggleton & De Raedt, 1994; Muggleton et al., 1990) is the most representative symbolic method. Given a collection of positive examples and negative examples, an ILP system aims to learn logic rules which are able to entail all the positive examples while excluding any of the negative examples. However it is difficult for ILP to scale beyond small rule sets due to their restricted computational complexity to handle the large search space of compositional rules. There are also some recent neural-symbolic methods that extend ILP, e.g., neural logic programming methods (Yang et al., 2017; Sadeghian et al., 2019) and principled probabilistic methods (Qu et al., 2020). Neural logic programming simultaneously learns logic rules and their weights in a differentiable way. Alternatively, principled probabilistic methods separate rule generation and rule weight learning by introducing a rule generator and a reasoning predictor. However, most of these approaches are particularly designed for the KG completion task. Moreover, since they require enumeration of rules given a maximum rule length T, the complexity of these methods grows exponentially as max rule length increases, which severely limits their systematic generalization capability. To overcome these issues, several works such as conditional theorem provers (CTPs) (Minervini et al., 2020), recurrent relational reasoning (R5) (Lu et al., 2022) focused on the model's systematicity instead. CTPs learn an adaptive strategy for selecting subsets of rules to consider at each step of the reasoning via gradient-based optimization while R5 performs rule extraction and logical reasoning with deep reinforcement learning equipped with a dynamic rule memory. Despite their strong generalizability to larger unseen graphs beyond the training sets (Sinha et al., 2019), they cannot handle KG completion tasks for large-scale KGs due to their high computational complexity.

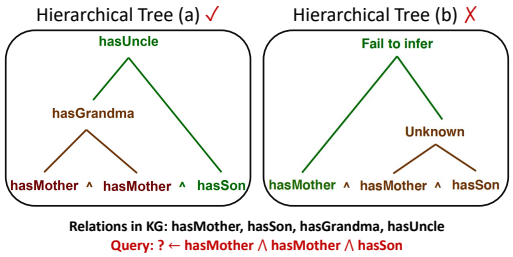

Figure 2: Learning an accurate hierarchical structure is significant for rule discovery: (a) a good compositional structure leading to the correct conclusion; (b) an improper compositional structure resulting in the failure of inference.

In this paper, we propose an end-to-end neural model to learn compositional logic rules for KG reasoning. Our proposed NCRL method is scalable and yields state-of-the-art (SOTA) results for link prediction on large-scale KGs. NCRL shows strong systematic generalization when tested

on larger unseen graphs beyond the training sets. NCRL views a logic rule as a composition of predicates and learns a hierarchical tree to express the rule composition. More specifically, NCRL breaks the rule body into small atomic compositions in order to infer the rule head. By recurrently merging compositions in the rule body with a recurrent attention unit, NCRL finally predicts a single rule head. The main contributions of this paper are summarized as follows:

- We formulate the rule learning problem from a new perspective and define the score of a logical rule based on the semantic consistency between rule body and rule head.
- NCRL presents an end-to-end neural approach to exploit the compositionality of a logical rule in a recursive way to improve models' systematic generalizability.
- NCRL is scalable and yields state-of-the-art (SOTA) results for link prediction on large-scale KGs, and demonstrates strong systematic generalization to larger unseen graphs beyond training sets.

2 NOTATION & PROBLEM DEFINITION

Knowledge Graph. A KG, denoted by $\mathcal{G} = \{E, R, O\}$, consists of a set of entities E, a set of relations R and a set of observed facts O. Each fact in O is represented by a triple (e_i, r_k, e_j).

Horn Rule. Horn rules, as a special case of first order logic rules, are composed of a body of conjunctive predicates (i.e., relations are called also predicates) and a single head predicate. In this paper, we are interested in mining chain-like **compositional Horn rules** [1] in the following form.

$$s(r_h, \mathbf{r_b}) : r_h(x, y) \leftarrow r_{b_1}(x, z_1) \wedge \cdots \wedge r_{b_n}(z_{n-1}, y) \tag{1}$$

where $s(r_h, \mathbf{r_b}) \in [0, 1]$ is the confidence score associated with the rule , and $r_h(x, y)$ is called **rule head** and $r_{b_1}(x, z_1) \wedge \cdots \wedge r_{b_n}(z_{n-1}, y)$ is called **rule body**. Combining rule head and rule body, we denote a Horn rule as $(r_h, \mathbf{r_b})$ where $\mathbf{r_b} = [r_{b_1}, \ldots, r_{b_n}]$.

Logic Rule Learning. Logic rule learning aims to learn a confidence score $s(r_h, \mathbf{r_b})$ for each rule $(r_h, \mathbf{r_b})$ in **rule space** to measure its plausibility. During rule extraction, top k rules with highest scores will be selected as the learned rules.

3 NEURAL COMPOSITIONAL RULE LEARNING (NCRL)

In this section, we introduce our NCRL to learn compositional logic rules. Instead of using the frequency of *rule instances* to measure the plausibility of logical rules, we define the score of a logical rule as the probability that the rule body can be replaced by the rule head based on their semantic consistency. An overview of NCRL is shown in Figure 3. NCRL starts by sampling a set of paths from a given KG, and further splitting each path into short compositions using a sliding window. Then, NCRL uses a reasoning agent to reason over all the compositions to select one composition. NCRL uses a recurrent attention unit to transform the selected composition into a single relation represented as a weighted combination of existing relations. By recurrently merging compositions in the path, NCRL finally predicts the rule head. Algorithm 1 outlines the learning procedure of NCRL . A detailed example to illustrate Figure 3 is given in Appendix A.1.

3.1 LOGIC RULE LEARNING WITH RECURRENT ATTENTION UNIT

As discussed in Section 1, while the rule body can be viewed as a sequence, it naturally exhibits a rich hierarchical structure. The semantics of the rule body is highly dependent on its hierarchical structure, which cannot be exploited by most of the existing rule learning methods. To explicitly allow our model to capture the hierarchical nature of the rule body, we need to learn how the relations in the rule body are combined as well as the principle to reduce each composition in the hierarchical tree into a single predicate.

3.1.1 HIERARCHICAL STRUCTURE LEARNING

Hierarchical structure of logic rules is learned in an iterative way. At each step, NCRL selects only one composition from the rule body and replaces the selected composition by another single predicate based on the recurrent attention unit to reduce the rule body. Although rule body is hierarchical,

[1] An instance of rule body of a chain-like compositional Horn rules is corresponding to a path in KG

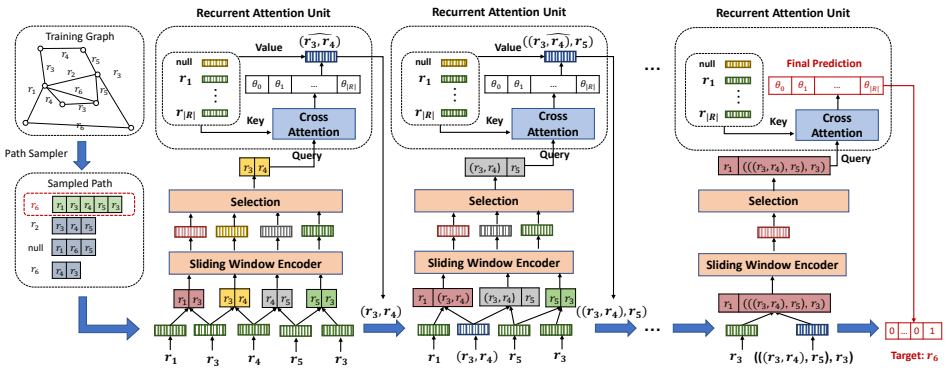

Figure 3: An overview of NCRL . It samples paths from KG (e.g.,$[r_1, r_3, r_4, r_5, r_3]$), and predicts the relations that directly connect the sampled paths (e.g.,r_6) based on the learned rules.

when operations are very local (i,e., leaf-level composition), a composition is strictly sequential. To identify a composition from a sampled path, we use a sliding window with different lengths to decompose the sampled paths into compositions of different sizes. In our implementation, we vary the size of the sliding window among $\{2, 3\}$. Given a fixed window size s, sliding windows are generated by a size s window which slides through the rule body $\mathbf{r_b} = [r_{b_1}, \ldots, r_{b_n}]$.

Sliding Window Encoder. When operations are over a local sliding window (i.e., composition), the relations within a sliding window should strictly follow a chain structure. Therefore, we utilize a RNN (Schuster & Paliwal, 1997) to encode a sliding window. For example, taking i-th sliding window whose size is 2 (i.e., $w_i = [r_{b_i}, r_{b_{i+1}}]$) as the input, RNN outputs:

$$[\mathbf{h}_i, \mathbf{h}_{i+1}] = \text{RNN}(w_i) \tag{2}$$

where $\mathbf{h}_i \in \mathbb{R}^d$ is a hidden-state corresponding to predicate r_{b_i} in w_i. The final hidden-state $\mathbf{h}_{i+1}$ is used as the representation of sliding window w_i. Thus, we have $\mathbf{w}_i = \mathbf{h}_{i+1}$.

Composition Selection. $\mathbf{w}_i$ is useful to estimate how likely the relations in i-th window appear together. If these relations always appear together, they have higher probability to form a meaningful composition. To incorporate this observation into our model, we select the sliding window by computing:

$$\mu = \text{softmax}([f(\mathbf{w}_1), f(\mathbf{w}_2), \ldots, f(\mathbf{w}_{n+1-s})]) \tag{3}$$

where f is a fully connected neural network. It learns the probability of i-th window to be a meaningful composition from its representation $\mathbf{w}_i$. w_i with highest μ_i will be selected as the input to the recurrent attention unit.

3.1.2 Recurrent Attention Unit

Note that rule induction following its underlying hierarchical structure is a recurrent process. Therefore, we propose a novel recurrent attention unit to recurrently reduce the selected composition into a single predicate until it outputs a final relation.

Attention-based Induction. The goal of a recurrent attention unit is to reduce the selected composition into a single predicate, which can be modeled as matching the composition with another single predicate based on their semantic consistency. Since attention mechanisms yield impressive results in Transformer models by capturing the semantic correlation between every pair of tokens in natural language sentence (Vaswani et al., 2017), we propose to utilize attention to reduce the selected composition $\mathbf{w}_i$. Note that we may not always find an existing relation to replace the selected composition. For example, given the composition [hasBrother, hasWife], none of the existing relations can be used to represent it. As such, in order to accommodate unseen relations, we incorporate a "null" predicate into potential rule heads and denote it as r_0. Let $H \in \mathbb{R}^{|R|+1 \times d}$ be the matrix of the concatenation of all head relations. By taking $\mathbf{w}_i$ as a query and H as the content, the scaled dot-product attention θ can be computed to estimate the semantic consistency between the selected composition and its potential heads:

$$\theta = \text{softmax}(\frac{\mathbf{w}_i W_Q (H W_K)^T}{\sqrt{d}}) \tag{4}$$

where $W_Q, W_K \in \mathbb{R}^{d\times d}$ are learnable parameters that project the inputs into the space of query and key. $\theta \in \mathbb{R}^{|R|+1}$ is the learned attention, in which θ_i measures $p(r_j|w_i)$ - the probability that the selected composition can be replaced by the predicate r_j based on their semantic consistency. Given θ, we are able to compute a new representation for the selected composition as a weighted combination of all head relations each weighted by its attention weight.

$$\widehat{\mathbf{w}_i} = \theta H W_V \tag{5}$$

where $\widehat{\mathbf{w}_i} \in \mathbb{R}^d$ is the new representation of the selected composition. We project the key and value to the same space by requiring $W_V = W_K$.

As shown in Figure 3, we can reduce the long rule body $[r_{b_1}, r_{b_2} \ldots, r_{b_n}]$ by recursively applying the attention unit to replace its composition $(r_{b_i}, r_{b_{i+1}})$ with a single predicate. In the final step of the prediction, the attention θ computed following Eq. 4 collects the probability that the rule body can be replaced by each of the head relations.

3.2 Training and Rule Extraction

NCRL is trained in an end-to-end fashion. It starts by sampling paths from an input KG, and predicts the relation which directly closes the sampled paths based on learned rules.

Path Sampling. We utilize a random walk (Spitzer, 2013) sampler to sample paths that connect two entities from the KG. Formally, given a source entity x_0, we simulate a random walk of max length n. Let x_i denote the i-th node in the walk, which are generated by the following distribution:

$$p(x_i = e_i | x_{i-1} = e_j) = \begin{cases} \frac{1}{|\mathcal{N}(e_j)|}, & \text{if } (e_i, e_j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $|\mathcal{N}(e_j)|$ is the neighborhood size of entity e_j. Different from random walk, each time after we sample the next entity x_i, we add all the edges which can directly connect x_0 and x_i in KG. We denote the path connecting two nodes x_0 and x_n as p, where $p = [r_{b_1}, \ldots, r_{b_n}]$, indicating $x_0 \xrightarrow{r_1} \ldots \xrightarrow{r_n} x_n$. We also denote the relation that directly connects x_0 and x_n as r_h. If none of the relations directly connects the x_0 and x_n, we set r_h as "null".

Objective Function. Our goal is to maximize the likelihood of the observed relation r_h, which directly closes the sampled path p. The attention θ collects the predicted probability for p being closed by each of the head relations. We formulate the objective using the cross-entropy loss as:

$$-\sum_{(p,r_h)\in\mathcal{P}} \sum_{k=0}^{|R|} \mathbf{y}_k^{r_h} \log \theta_k^p \tag{7}$$

where $\mathcal{P}$ denotes a set of sampled paths from a given KG, $\mathbf{y}^{r_h} \in \{0,1\}^{|R|+1}$ is the one-hot encoded vector such that only the r_h-th entry is 1, and $\theta^p \in \mathbb{R}^{|R|+1}$ is the learned attention for the sampled path p. In particular, θ_0^p represents the probability that the sampled path cannot be closed by any existing relations in KG.

Rule Extraction. To recover logical rules, we calculate the score $s(r_h, \mathbf{r_b})$ for each rule $(r_h, \mathbf{r_b})$ in rule space based on the learned model. Given a candidate rule $(r_h, \mathbf{r_b})$, we reduce the rule body $\mathbf{r_b}$ into a single head r_h by recursively merge compositions in path $\mathbf{r_b}$. At the final step of the prediction, we learn the attention $\theta = [\theta_0, \ldots, \theta_{|R|}]$, where θ_k is the score of rule $(r_k, \mathbf{r_b})$. Top k rules with highest score will be selected as learned rules.

Algorithm 1: Learning Algorithm

Input: Observed triples in KG O
Output: Relation embeddings

```
P = SamplePaths(O)
for (p, r_h) ∈ P do
    while len(p) > s do
        // Decompose p with a sliding window, whose size is s
        [w_1, ..., w_{n+1-s}] = Decompose(p)
        // Select a composition
        [w_1, ..., w_{n+1-s}] = RNN([w_1, ..., w_{n+1-s}])
        w_i = Select([w_1, ..., w_{n+1-s}])
        // Apply recurrent attention unit
        ŵ_i = Attn(w_i)
        // Reduce the sampled path p
        p = [r_{b_1}, ..., w_i, ..., r_{b_n}]
    end
    // Final prediction
    w = RNN(p)
    Take w as the query and compute θ based on Eq. 4
    Minimize the loss in Eq. 7
end
```

4 EXPERIMENTS

Logic rules are valuable for various downstream tasks, such as (1) *KG completion task*, which aims to infer the missing entity given the query $(h, r, ?)$ or $(?, r, t)$; (2) A more challenging *inductive relational reasoning task*, which tests the systematic generalization capability of the model by inferring the missing relation between two entities (i.e., $(h, ?, t)$) with more hops than the training data. A majority of existing methods can handle only one of these two tasks (e.g., RNNLogic is designed for KG completion task while R5 is designed for inductive relational reasoning task). In this section, we show that our method is superior to existing SOTA algorithms on both tasks. In addition, we also empirically assess the interpretability of the learned rules.

4.1 KNOWLEDGE GRAPH COMPLETION

KG completion is a classic task widely used by logic rule learning methods such as Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019) and RNNLogic (Qu et al., 2020) to evaluate the quality of learned rules. An existing algorithm called forward chaining (Salvat & Mugnier, 1996) can used to predict missing facts from logic rules.

Datasets. We use six widely used benchmark datasets to evaluate our NCRL in comparison to state-of-the-art methods from knowledge graph embedding and rule learning methods. Specifically, we use the Family (Hinton et al., 1986), UMLS (Kok & Domingos, 2007), Kinship (Kok & Domingos, 2007), WN18RR (Dettmers et al., 2018), FB15K-237 (Toutanova & Chen, 2015), YAGO3-10 (Suchanek et al., 2007) datasets. The statistics of the datasets are given in Appendix A.3.1.

Evaluation Metrics. We mask the head or tail entity of each test triple, and require each method to predict the masked entity. During evaluation, we use the filtered setting (Bordes et al., 2013) and three evaluation metrics, i.e., Hit@1, Hit@10 and MRR.

Comparing with Other Methods. We evaluate our method against SOTA methods, including: (1) traditional KG embedding (KGE) methods (e.g., TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), ConvE (Dettmers et al., 2018), ComplEx (Trouillon et al., 2016) and RotatE (Sun et al., 2019)); (2) logic rule learning methods (e.g., Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), RNNLogic (Qu et al., 2020) and RLogic (Cheng et al., 2022)). The systematic generalizable methods (e.g., CTPs and R5) cannot handle KG completion tasks due to their high complexity.

Results. The comparison results are presented in Table 1. We observe that: (1) Although NCRL is not designed for KG completion task, compared with traditional KGE models, it achieves comparable result on all datasets, especially on Family, UMLS and WN18RR datasets; (2) NCRL consistently outperforms all other rule learning methods with significant performance gain in most cases.

4.1.1 ABLATION STUDY

Performance w.r.t. Data Sparsity. We construct sparse KG by randomly removing θ triples from original dataset. Following this approach, we vary the sparsity ratio θ among $\{0.33, 0.66, 1\}$ and report performance on different methods over KG completion task on Kinship dataset. As presented in Fig. 4, the performance of NCRL does not vary a lot with different sparsity ratio θ, which is appealing in practice. More analysis on other datasets are given in Appendix A.3.2.

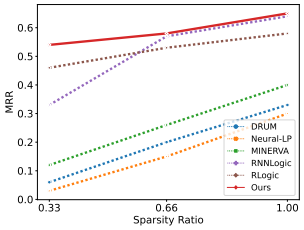

Figure 4: Performance of KG completion vs sparsity ratio on Kinship.

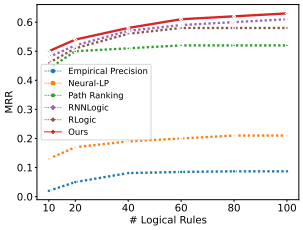

Figure 5: Performance of KG completion vs # logic rules on Kinship.

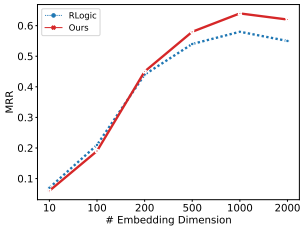

Figure 6: Performance of KG completion vs embedding dimension on Kinship.

Table 1: Link prediction. The red numbers represent the best performances among all methods, while the blue numbers represent the best performances among all *rule learning* methods.

Methods	Models	Family			Kinship			UMLS		
		MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
KGE	TransE	0.45	22.1	87.4	0.31	0.9	84.1	0.69	52.3	89.7
	DistMult	0.54	36.0	88.5	0.35	18.9	75.5	0.391	25.6	66.9
	ComplEx	0.81	72.7	94.6	0.42	24.2	81.2	0.41	27.3	70.0
	RotatE	0.86	78.7	93.3	**0.65**	**50.4**	93.2	0.74	63.6	93.9
Rule Learning	Neural-LP	0.88	80.1	98.5	0.30	16.7	59.6	0.48	33.2	77.5
	DRUM	0.89	82.6	99.2	0.33	18.2	67.5	0.55	35.8	85.4
	RNNLogic	0.86	79.2	95.7	0.64	**49.5**	92.4	0.75	63.0	92.4
	RLogic	0.88	81.3	97.2	0.58	43.4	87.2	0.71	56.6	93.2
	NCRL	**0.92**	**85.6**	**99.6**	**0.65**	49.4	**93.6**	**0.78**	**66.1**	**95.2**

Methods	Models	WN18RR			FB15K-237			YAGO3-10		
		MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
KGE	TransE	0.23	2.2	52.4	0.29	18.9	46.5	0.36	25.1	58.0
	DistMult	0.42	38.2	50.7	0.22	13.6	38.8	0.34	24.3	53.3
	ConvE	0.43	40.1	52.5	**0.32**	21.6	50.1	0.44	35.5	61.6
	ComplEx	0.44	41.0	51.2	0.24	15.8	42.8	0.34	24.8	54.9
	RotatE	0.47	42.9	55.7	**0.32**	**22.8**	**52.1**	**0.49**	**40.2**	**67.0**
Rule Learning	Neural-LP	0.38	36.8	40.8	0.24	17.3	36.2	-	-	-
	DRUM	0.38	36.9	41.0	0.23	17.4	36.4	-	-	-
	RNNLogic	0.46	41.4	53.1	0.29	20.8	44.5	-	-	-
	RLogic	0.47	44.3	53.7	**0.31**	20.3	**50.1**	0.36	25.2	50.4
	NCRL	**0.67**	**56.8**	**85.2**	**0.31**	**22.0**	48.2	**0.38**	**27.8**	**54.1**

† Neural-LP, DRUM and RNNLogic exceeds the memory capacity of our machines on YAGO3-10 dataset

KG completion performance w.r.t. the Number of Learned Rules. To investigate how the number of learned rules affect the KG completion performance, we generate k rules with highest qualities per query relation, and use them to predict missing links. We vary k among $\{10, 20, 40, 60, 80, 100\}$. The results on Kinship are given in Fig. 5. We observed that even with only 10 rules per relation, NCRL still givens competitive results. More analysis on other datasets are given in Appendix A.3.2.

Performance w.r.t. Embedding Dimension. To investigate how the dimension of relation embeddings affect the KG completion performance, we vary the dimension of relation embeddings among $\{10, 100, 200, 500, 1000, 2000\}$ and present the results on Kinship in Fig. 6, comparing against RLogic (Cheng et al., 2022). We see that embedding dimension has a significant impact on KG completion performance. The best performance is achieved at $d = 1000$.

4.2 Training Efficiency

To demonstrate the scalability of NCRL , we give the training time of NCRL against other logic rule learning methods on three benchmark datasets in Table 2. We observe that: (1) Neural-LP and DRUM do not perform well in terms of efficiency as they apply a sequence of large matrix multiplications for logic reasoning. They cannot handle YAGO3-10 dataset due to the memory issue; (2) It is also challenging for RNNLogic to scale to large scale KGs as it relies on all ground rules to evaluate the generated rules in each iteration. It is difficult for it to handle KG with hundreds of relations (e.g., FB15K-237) nor KG with million entities (e.g., YAGO3-10); (3) our NCRL is on average 100x faster than state-of-the-art baseline methods.

Table 2: Training time (s) of rule learning methods

	NeuralLP	DRUM	RNNLogic	NCRL
WN18RR	1,308	1,146	1,044	**77**
FB15k-237	23,708	22,428	-	**410**
YAGO3-10	-	-	-	**190**

4.3 Systematic Generalization

We test NCRL for systematic generalization to demonstrate the ability of NCRL to perform reasoning over graphs with more hops than the training data, where the model is trained on smaller graphs and tested on larger unseen ones. The goal of this experiment is to infer the relation between nodel pair query. We use two benchmark datasets: (1) CLUTRR (Sinha et al., 2019) which is a dataset for inductive relational reasoning over family relations, and (2) GraphLog (Sinha et al., 2020) is a benchmark suite for rule induction and it consists of logical worlds and each world contains graphs

Table 3: Results of inductive relational reasoning on CLUTRR dataset. Trained on path samples with hops $\{2, 3, 4\}$ and evaluated on path samples with hops $\{5, \ldots, 10\}$. The red numbers represent the best performances while the brown numbers represent the second best performances.

Model \ # Hops	**5 Hops**	**6 Hops**	**7 Hops**	**8 Hops**	**9 Hops**	**10 Hops**
RNN	0.93±0.06	0.87±0.07	0.79±0.11	0.73±0.12	0.65±0.16	0.64±0.16
LSTM	0.98±0.03	0.95±0.04	0.89±0.10	0.84±0.07	0.77±0.11	0.78±0.11
GRU	0.95±0.04	0.94±0.03	0.87±0.8	0.81±0.13	0.74±0.15	0.75±0.15
Transformer	0.88±0.03	0.83±0.05	0.76±0.04	0.72±0.04	0.74±0.05	0.70±0.03
GNTP	0.68±0.28	0.63±0.34	0.62±0.31	0.59±0.32	0.57±0.34	0.52±0.32
GAT	0.99±0.00	0.85±0.04	0.80±0.03	0.71±0.03	0.70±0.03	0.68±0.02
GCN	0.94±0.03	0.79±0.02	0.61±0.03	0.53±0.04	0.53±0.04	0.41±0.04
CTP_L	**0.99±0.02**	0.98±0.04	0.97±0.04	**0.98±0.03**	0.97±0.04	0.95±0.04
CTP_A	0.99±0.04	**0.99±0.03**	0.97±0.03	0.95±0.06	0.93±0.07	0.91±0.05
CTP_M	0.98±0.04	0.97±0.06	0.95±0.06	0.94±0.08	0.93±0.08	0.90±0.09
RLogic	**0.99±0.02**	0.98±0.02	0.97±0.04	0.97±0.03	0.94±0.06	0.94±0.07
R5	**0.99±0.02**	0.99±0.04	**0.99±0.03**	**1.0±0.02**	**0.99±0.02**	**0.98±0.03**
NCRL	**1.0±0.01**	**0.99±0.01**	**0.98±0.02**	**0.98±0.03**	**0.98±0.03**	**0.97±0.02**

Table 4: Results of inductive relational reasoning on GraphLog datasets for robustness analysis.

	CTP		RLogic		R5		NCRL	
	ACC	Recall	ACC	Recall	ACC	Recall	ACC	Recall
World 2	0.685±0.03	0.80±0.05	0.726±0.02	0.95±0.00	0.755 ±0.02	1.0±0.00	**0.774±0.01**	1.0±0.00
World 3	0.624±0.02	0.85±0.00	0.737±0.02	1.0±0.00	0.791±0.03	1.0±0.00	**0.797±0.02**	1.0±0.00
World 6	0.533 ±0.03	0.85±0.00	0.638 ±0.03	0.90±0.00	0.687±0.05	0.9±0.00	**0.702±0.02**	0.95±0.00
World 8	0.545±0.02	0.70±0.00	0.605±0.02	0.90±0.00	0.671±0.03	0.95±0.00	**0.687±0.02**	0.95±0.00

Table 5: Top rules learned on YAGO3-10. We highlight the composition and predicate which share the same semantic meaning with boldface.

$\text{isLocatedIn}(x, y) \leftarrow \textbf{isLocatedIn}(x, z) \wedge \text{isLocatedIn}(z, y)$
$\text{isLocatedIn}(x, y) \leftarrow \textbf{hasAcademicAdvisor}(x, z_1) \wedge \textbf{isLocatedIn}(z_1, z_2) \wedge \text{isLocatedIn}(z_2, y)$
$\text{isAffiliatedTo}(x, y) \leftarrow \text{isKnownFor}(x, z) \wedge \textbf{isAffiliatedTo}(z, y)$
$\text{isAffiliatedTo}(x, y) \leftarrow \text{isKnownFor}(x, z_1)\ \textbf{isAffiliatedTo}(z_1, z_2) \wedge \textbf{isLeaderOf}(z_2, y)$
$\text{playsFor}(x, y) \leftarrow \text{isKnownFor}(x, z) \wedge \textbf{isAffiliatedTo}(z, y)$
$\text{playsFor}(x, y) \leftarrow \text{isKnownFor}(x, z_1)\ \textbf{playsFor}(z_1, z_2) \wedge \textbf{owns}(z_2, y)$
$\text{influences}(x, y) \leftarrow \text{isPoliticianOf}(x, z) \wedge \textbf{influences}(z, y)$
$\text{influences}(x, y) \leftarrow \text{isPoliticianOf}(x, z_1) \wedge \textbf{influences}(z_1, z_2) \wedge \textbf{influences}(z_2, y)$

generated under different set of rules. Note that most of existing rule learning methods lack systematic generalization. CTPs (Minervini et al., 2020), R5 (Lu et al., 2022) and RLogic (Cheng et al., 2022) are the only comparable rule learning methods for this task. The detailed statistics and the description of the datasets is summarized in Appendix A.4.1.

Systematic Generalization on CLUTRR. Table 3 shows the results of NCRL against SOTA algorithms. The detailed information about the SOTA algorithms are given in Appendix A.4.2. We observe that the performances of sequential models and embedding-based models drop severely when the path length grows longer while NCRL still predicts successfully on longer paths without significant performance degradation. Comparing with systematic generalizable rule learning methods, NCRL has better generalization capability than CTPs especially when the paths grows longer. Even though R5 gives invincible results over CLUTRR dataset, NCRL shows comparable performance.

Systematic Generalization on GraphLog. Table 4 shows the results on 4 selected worlds. We observed that NCRL consistently outperforms other rule learning baselines over all 4 worlds.

4.4 Case Study of Generated Logic Rules

Finally, we show a case study of logic rules that are generated by NCRL on the YAGO3-10 dataset in Table 5. We can see that these logic rules are meaningful and diverse. Two rules with different lengths are presented for each head predicate. We highlight the composition and predicate which share the same semantic meaning with boldface.

5 RELATED WORK

Inductive Logic Programming. Mining Horn clauses has been extensively studied in the Inductive Logic Programming (ILP) (Muggleton & De Raedt, 1994; Muggleton et al., 1990; Muggleton, 1992; Nienhuys-Cheng & De Wolf, 1997; Quinlan, 1990) . Given a set of positive examples, as well as a set of negative examples, an ILP system aims to learn logic rules which are able to entail all the positive examples while exclude any of the negative examples. Although ILP has shown its power in many areas (Tsunoyama et al., 2008; Zelle & Mooney, 1993), scalability is a central challenge for ILP methods as they involves several steps that are NP-hard.

Neural-Symbolic Methods. Very recently, several methods are proposed to extend the idea of ILP by simultaneously learning logic rules and the weights in a differentiable way. Most of them are based on neural logic programming. For example, Neural-LP (Yang et al., 2017) enables logical reasoning via sequences of differentiable tensor multiplication. An neural controller system based on attention is used to learn the score of a specific logic. However, Neural-LP could learn higher score of a meaningless rule because it shares an atom with a useful rule. To address this problem, RNNs are utilized in DRUM (Sadeghian et al., 2019) to prune the potential incorrect rule bodies. In addition, Neural-LP can learn only chain-like Horn rules while NLIL (Yang & Song, 2019) extends Neural-LP to learn Horn rules in more general form. Because neural logic programming approaches involve large matrix multiplication and simultaneously learn logic rules and their weights, which is nontrivial in terms of optimization, they cannot handle large KGs, such as YAGO3-10. To address this issue, RNNLogic (Qu et al., 2020)) is proposed to separate rule generation and rule weight learning by introducing a rule generator and a reasoning predictor respectively. Although the introduction of the rule generator reduces the search space, it is still challenging for RNNLogic to scale to KGs with hundreds of relations (e.g., FB15K-237) or millions of entities (e.g., YAGO3-10).

Systematic Generalizable Methods. All the above methods cannot generalize to larger graph beyond training sets. To improve models' systematicity, Conditional Theorem Provers (CTPs) is proposed to learn an optimal rule selection strategy via gradient-based optimisation. For each sub-goal, a select module produces a smaller set of rules, which is then used during the proving mechanism. However, since the length of the learned rules influences the number of parameters of the model, it limits the capability of CTPs to handle the complicated rules whose depth is large. In addition, due to its high computational complexity, CTPs cannot handle KG completion tasks for large-scale KGs. Another reinforcement learning based method - R5 (Lu et al., 2022) is proposed to provide recurrent relational reasoning solution to learn compositional rules. However, R5 cannot generalize to KG completion task due to the lack of scalability. It requires pre-sampling for the paths that entails the query. Considering that all the triples in a KG share the same training graph, even a relatively small scale KG contains a huge number of paths. Thus, it is impractical to applied R5 to even small scale KG for rule learning. In addition, R5 employs a hard decision mechanism for merging a relation pair into a single relation, which makes it challenging to handle the widely existing uncertainty in KGs. For example, given the rule body $\textit{hasAunt}(x, z) \wedge \textit{hasSister}(z, y)$, both $\textit{hasMother}(x, y)$ and $\textit{hasAunt}(x, y)$ can be derived as the rule head. The inaccurate merging of a relation pair may result in the error propagation when generalize to longer paths. Although RLogic (Cheng et al., 2022) are generalizable across multiple tasks, including KG completion and inductive relation reasoning. However, the performance is not satisfying compared to our proposed NCRL .

6 CONCLUSION

In this paper, we propose NCRL , an end-to-end neural model for learning compositional logic rules. NCRL treats logic rules as a hierarchical tree, and breaks the rule body into small atomic compositions in order to infer the head rule. By recurrently merging compositions in the rule body with a recurrent attention unit, NCRL finally predicts a single rule head. Experimental results show that NCRL is scalable, efficient, and yields SOTA results for link prediction on large-scale KGs. Moreover, we test NCRL for systematic generalization for inductive relational reasoning, by learning to reason on small observed graphs and evaluate on larger unseen graphs. NCRL demonstartes strong generalization across tasks compared to SOTA rule learning methods.

REFERENCES

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pp. 2787–2795, 2013.

Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. Rlogic: Recursive logical rule learning from knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 179–189, 2022.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jens Graupmann, Ralf Schenkel, and Gerhard Weikum. The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents. In *Proceedings of the 31st international conference on very large data bases*, pp. 529–540. VLDB Endowment, 2005.

Geoffrey E Hinton et al. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, pp. 12. Amherst, MA, 1986.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.

Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th international conference on Machine learning*, pp. 433–440, 2007.

Shengyao Lu, Bang Liu, Keith G Mills, SHANGLING JUI, and Di Niu. R5: Rule discovery with reinforced and recurrent relational reasoning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=2eXhNpHeW6E`.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pp. 1211–1220. International World Wide Web Conferences Steering Committee, 2017.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *International Conference on Machine Learning*, pp. 6938–6949. PMLR, 2020.

Stephen Muggleton. *Inductive logic programming*. Number 38. Morgan Kaufmann, 1992.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

Stephen Muggleton, Cao Feng, et al. *Efficient induction of logic programs*. Citeseer, 1990.

Shan-Hwei Nienhuys-Cheng and Ronald De Wolf. *Foundations of inductive logic programming*, volume 1228. Springer Science & Business Media, 1997.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*, 2020.

J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5(3):239–266, 1990.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *arXiv preprint arXiv:1911.00055*, 2019.

Eric Salvat and Marie-Laure Mugnier. Sound and complete forward and backward chainings of graph rules. In *International Conference on Conceptual Structures*, pp. 248–262. Springer, 1996.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text. *arXiv preprint arXiv:1908.06177*, 2019.

Koustuv Sinha, Shagun Sodhani, Joelle Pineau, and William L Hamilton. Evaluating logical generalization in graph neural networks. *arXiv preprint arXiv:2003.06560*, 2020.

Frank Spitzer. *Principles of random walk*, volume 34. Springer Science & Business Media, 2013.

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

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

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

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

Kazuhisa Tsunoyama, Ata Amini, Michael JE Sternberg, and Stephen H Muggleton. Scaffold hopping in drug discovery using inductive logic programming. *Journal of chemical information and modeling*, 48(5):949–957, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pp. 1271–1279. International World Wide Web Conferences Steering Committee, 2017.

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

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pp. 2319–2328, 2017.

Yuan Yang and Le Song. Learn to explain efficiently via neural logic inductive learning. *arXiv preprint arXiv:1910.02481*, 2019.

Wentau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *IJCNLP*, pp. 1321–1331, Beijing, China, July 2015.

John M Zelle and Raymond J Mooney. Learning semantic grammars with constructive inductive logic programming. In *AAAI*, pp. 817–822, 1993.