Logical Rule Learning from Knowledge Graphs

Aditya Jain [*1] **Manish Reddy Gottimukkula** [*1] **Nilay Pochhi** [*1]

Abstract

Often logical rules represent the domain knowledge more accurately and forms the basis for widely used logical reasoning. Learning logical rules from Knowledge Graphs (KGs) would help in understanding the data better and improve the performance of upstream tasks. Many previous attempts to learn logical rules automatically from Knowledge Graphs (KGs) rely on the complete set of rule instances for rule evaluation and thus suffer from severe computational inefficiency. To evaluate rules in a more efficient way, a novel framework *RLogic* was recently developed which aims to learn the rules directly at the schema level. In addition, RLogic incorporates deductive nature into rule learning. Through experiments Rlogic is shown to be more efficient and effective than existing state-of-the-art algorithms. Despite the visible improvement, a closer look at RLogic framework would reveal the possibility of improving it further. In this project we aim to experiment with multiple ideas on existing RLogic architecture for even more improvement in logical rule learning. Github Code Link - https://github.com/npochhi/CS249-RLogic-Attention

1. Introduction

Logical rule mining is the process to deduce logical rules from the given data. Logical Reasoning leverages those logical rules to derive missing knowledge and allows easy generalization to unobserved objects. Significant efforts are being made to integrate logical reasoning into neural network learning for various real-world applications.

Due to the benefits of logical rules, studying the underlying patterns in the data to learn logical rules automatically is very important. Note that logical rule is a schema level concept, while only instance level evidence can be directly observed from KGs. To bridge the gap between instance level observation and schema level abstraction, the frequency of *rule instances* is utilized to define the plausibility of the logical rules. Traditional methods are represented by association rule mining (Galárraga et al., 2013), where the score is defined as the ratio between the total number of rule instances and the total number of body instances for the corresponding rule. Then the rule mining process is to search over the entire rule space and select the rules with highest plausibility score. For example, in the KG given in Fig. 1, below two rules can be found with the high plausibility score:

$$\delta_1 := hasGrandma(x, y) \leftarrow hasMother(x, z) \wedge hasMother(z, y)$$
$$\delta_2 := hasUncle(x, y) \leftarrow hasMother(x, z_1) \wedge hasMother(z_1, z_2) \wedge hasSon(z_2, y)$$

Although many efforts have been made to learn logical rules automatically, there are two limitations for the existing studies. First, relying on all rule instances for rule evaluation is not scalable considering the huge sizes of KGs. Second, the existing methods are unable to mine rules that have no support from rule instances. For example, due to the lack of support evidence, the following rule cannot be learned from Fig. 1:

$$\delta_3 := hasUncle(x, y) \leftarrow hasGrandma(x, z) \wedge hasSon(z, y)$$

RLogic approach alleviates this problem by pushing *deductive reasoning* deeper into rule learning. Using deductive reasoning, RLogic will be able to mine δ_3. Below are the main contributions made by RLogic framework:

- Learn logical rules directly at the schema level via sampled rule instances.
- Pushing deductive reasoning deeper into rule learning to validate a rule when it lacks supporting evidence.

In this project, we would like propose a few methods to improve the existing RLogic framework. First, we would like to incorporate attention based mechanism over the predicates in the rule body. And then we will use transforms based model in the existing representation learning for better prediction of intermediate relations.

2. Related Work

There are different ways in which one can accomplish the goal of rule mining. The subsequent sections describe the related work utilizing different techniques.

2.1. Association Rule Mining

Association rule mining learns rules by searching over a large rule space. The frequency of rule instances is used to estimate the plausibility of a specific logic. AMIE (Galárraga et al., 2013) proposed PCA confidence as a criteria to prune rules under open world assumption. Efficiency is still a central challenge because association rule mining relies on the complete set of rule instances for rule evaluation.

2.2. Neural Logic Programming

Very recently, neural logic pro- gramming approaches are proposed to extend the rule mining prob- lem from counting to learning. Methods such as Neural-LP (Yang et al., 2017) enables rule instances to be softly counted via sequences of differ- entiable tensor multiplication. 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.

2.3. Inductive Logic Programming

Mining Horn clauses has been extensively studied in the Inductive Logic Programming (ILP) (Muggleton & de Raedt, 1994) (Nienhuys-Cheng et al., 1997). 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. Scalability is a big issue for these methods.

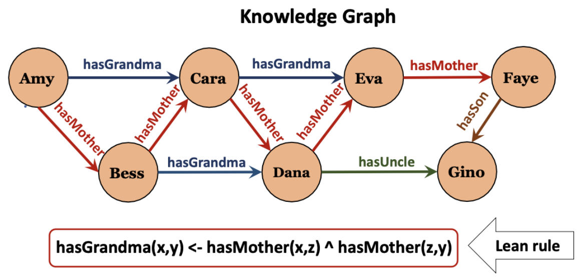

Figure 1. An example KG and a detected rule with length 2.

3. RLogic Model

In this section, we'll discuss the architecture and training details of base RLogic model. Instead of relying on all rule instances for rule evaluation, RLogic learns a classifier to determine the plausibility of logic rules. As a small amount of sampled closed paths are enough to train the classifier, it greatly improves the efficiency. In addition, RLogic incorporates a significant property of logical rules - deductive nature into rule learning. The introduction of deductive nature allows RLogic to follow logical deduction to validate a rule, which is critical when a rule lacks supporting evidence.

3.1. Problem Definition

A KG, denoted by 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),where $e_i, r_j \in E$ and $r_j \in R$. Rules are called **closed paths** in the language of KG. For example, Fig1 has a closed paths between Amy, Bess and Cara. Such closed paths forms the base for us to learn the logical rules from the knowledge base. To bridge the gap between instance level observation and schema level abstract, we introduce **relation path** and **target relation** as follows. By considering the relations in a closed path, we can divide a closed path into two components: (1) a relation path, which is defined as a sequence of relations $[r_1, ..., r_n]$ through which two entities and e_i and e_j can be connected on the graph, and (2) a target relation, which is defined as a single relation r_h . It can close the relation path by connecting two entities e_i and e_j directly. In Fig1, [*hasMother, hasMother, hasSon*] is a relation path through which *Dana* and *Gino* can be connected and relation *hasUncle* is the target relation. The problem of logical rule learning aims to assign a plausibility score s(δ) for each rule δ in rule space.

3.2. Training Data Generation

Rather than enumerate all closed paths in KG, RLogic utilizes a closed path sampler to sample only a small portion of closed paths to train the classifier. A random walk based procedure is used to efficiently sample the close paths. Formally, given a source entities x_0, we simulate a random walk of fixed length *n*. Let x_i denote the *i*th node in the walk. Nodes x_i are generated by the following distribution:

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

where $|\mathcal{N}(e_j)|$ is the neighbor size of entity e_j. Different from random walk, each time after we sample the next node x_i , we add all edges which can directly connect the source entity x_0 and x_i in KG to construct the closed paths. Once we sample such closed paths, if rule instances count is similar to body instances count (as in Fig2), we will consider it as a rule for training.

3.3. Training model

Each sampled closed path can be regarded as a labeled example, whose target relation is the ground truth label. There-

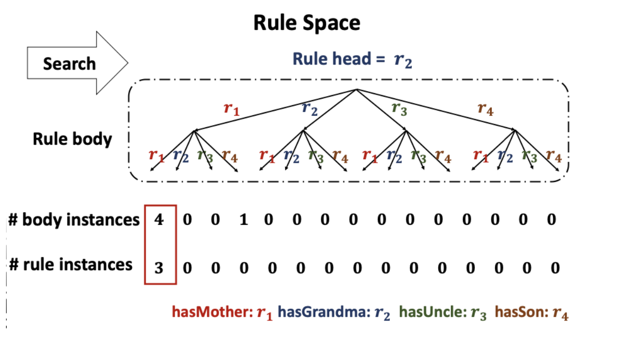
Rule Space
Search
Rule head = r_2
Rule body
r_1 r_2 r_3 r_4
# body instances 4 0 0 1 0 0 0 0 0 0 0 0 0 0 0
# rule instances 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
hasMother: r_1 hasGrandma: r_2 hasUncle: r_3 hasSon: r_4

Figure 2. A search tree for rules with length 2 and head relation r_2. A path from root to leaf corresponds to a rule body. Together with the head relation, it forms a candidate rule..

fore, the training contains two steps: (1) learn an encoder to represent a relation path; and (2) learn a classifier to predict the "label" (i.e., target relation) of a relation path. Note that we may not always find a proper target relation to represent a relation path and hence to accommodate unseen relations a "null" predicate is introduced. The training objective for each sampled close path can be formulated as the cross-entropy loss: L = -$\sum_{k=0}^{|R|} y_k log\theta_k$, where $y \in 0, 1^{|R|+1}$ is the one-hot encoded label vector such that only the r_h-th entry is 1, and $\theta \in R^{|R|+1}$ is the predicted probability over all predicates in KG, which is computed by one layer MLP with softmax activation over the sequence embedding. $\theta[0]$ represents the probability for "null" predicate.

To incorporate deductive nature, RLogic proposes to push deductive reasoning deeper into confidence prediction. The main idea of logical deduction is to decompose the inference of long rules into simply steps. Let us consider a relation path $[r_1, r_2, ...r_n]$. Instead of deriving r_h directly in one step, RLogic decomposes the inference into multiple steps. For each step i, we only need to infer target relation $r_I^{(i)}$ for prefix $r_{x\leq i}$ based on the result in previous step $r_I^{(i-1)}$ and predicate r_i. Using this recursive definition, RLogic has the model as follows (Fig3). It contains two substeps for each time step i: (1) target relation prediction, which aims to predict the target relation $r_I^{(i)}$ at step i given two-relation sequence $[r_I^{(i-1)}, r_i]$ as input; (2) weighted sum mechanism, which aims to learn embedding $\tilde{r}^i$ to represent target relation $r_I^{(i)}$ at step *i* by softly adding up all possible derivations. These two components are detailed below.

Target Relation Prediction Target relation prediction aims to predict target relation $r_I^{(i)}$ for prefix $r_{x\leq i}$ at each step *i*. Using the recursive definition above, instead of taking the whole prefix $r_{x\leq i}$ as input, we can predict $r_I^{(i)}$ based on previous calculation $r_I^{(i-1)}$.

Weighted sum. To construct a reliable representation of target relation at step *i*, weighted sum mechanism learns embedding $\tilde{r}^i$ to represent target relation $r_I^{(i)}$ by softly adding up all possible derivations at step *i*, which can be defined as:

$$\tilde{r}^{(i)} = \sum_{k=0}^{|R|} \theta^{(i)}[k].\mathbf{r}_k \tag{2}$$

where $\theta^{(i)}$ is the prediction computed at step i and $\mathbf{r}_k$ is the embedding of predicate r_k. Note that we introduce a "null" relation into the prediction. Its corresponding relation embedding can be modeled as the concatenation of the embedding of $\tilde{\mathbf{r}}^{(i-1)}$ and $\mathbf{r}_i$.

4. Proposed methods

Our aim is to add basic attention mechanism to the predicates in the rule body such that contributions from multiple predicates would contribute to result in the unknown intermediate relations. And then we would like to use the transformer based attention in place of current fully connected neural network layers to add more global emphasis on every relation. Also, we have proposed a GCN based approach to learn relation embeddings in which graph is constructed using the links between relations in the sampled training data. Combining above ideas, we aim to increase the performance of existing RLogic model.

4.1. Recursive Prediction using GCN

We plan to use Graph Convolutional neural network (GCN) (Kipf & Welling, 2016) for encoding of relations which are then processed recursively as given in RLogic. The processed graph node embeddings are passed on to a MLP which outputs probability scores which are then used to get the body embedding to be used in tandem with the next relation embedding. The graph convolutional neural network processes the relation embeddings in the following manner.

$$Z = \hat{A}\,\mathrm{ReLU}(\hat{A}XW^{(0)})W^{(1)}$$

Here, Z are the learned features of GCN which are to be passed on to RLogic, $\hat{A}$ is the aggregation function which we get by doing operation on A which is an adjacency matrix. $W^{(0)}$ and $W^{(1)}$ are the learnable parameters and are calculated by backpropogating from the RLogic head prediction loss. The input features of this GCN are the embeddings which are also trained using gradient descent. The graph on which this GCN is applied is constructed in the following manner. As we sample the paths from the knowledge graph to be inputted to RLogic, those paths can be utilized to construct a graph. If the path sampled is as

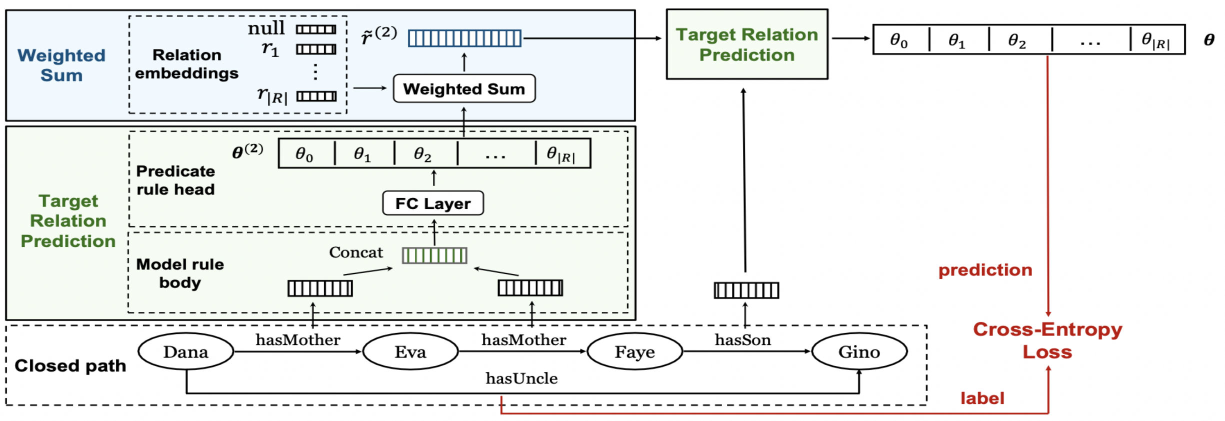

Figure 3. Illustration of the RLogic framework.

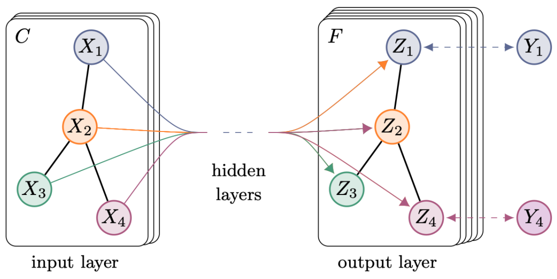

Figure 4. Graph Convolutional Neural Network

follows -

$$R_3(a,c) \leftarrow R_1(a,b) \wedge R_2(b,c)$$

Here, R_1 and R_2 are the relations connecting (a,b) and (b,c) respectively. The total number of nodes in the graph are $|R|$ where R are the total number of relations. There exists an edge between R_1 and R_2 if path of the above type is sampled from the knowledge graph. The graph is constructed once before starting the training and is saved for the validation and testing phase.

GCN based embedding learning would be useful for taking in the context of the neighbors all at once. The recursive learning takes into consideration the immediate neighbor of a relation but not other one hop neighbors. GCN helps us solve this shortcoming and would contribute to a performance gain. GCN compromises on the scalability a bit but that would be solved by using sampling techniques like ((Zou et al., 2019), (Zeng et al., 2019) etc).

4.2. RLogic with naive attention

(Bahdanau et al., 2014) introduced the attention mechanism for sequence to sequence tasks. Rather than building a single context vector out of the encoder's last hidden state, the secret sauce invented by attention is to create shortcuts between the context vector and the entire source input. The weights of these shortcut connections are customizable for each output element. We plan to adapt this attention mechanism for our RLogic framework. Attention would help us gain context during body embedding prediction phase. The attention mechanism we devise is as follows. Let

$$\begin{aligned}
\mathrm{e}_i &= FC\_layer(e_{i-1}, r_i)\\
\mathrm{q}_i &= FC_q(e_i)\\
\mathrm{K}_i &= q_i\dot{R}\\
\mathrm{Pr}_i &= softmax(K_i)
\end{aligned}$$

Here, e are the body embeddings till now and r are the relation embeddings. R is the matrix containing embeddings for all the relations,. FC is a fully connected layer and $softmax$ depicts a standard softmax function. By adding naive attention, the RLogic framwork would take into consideration the global context and would better predict the next relations.

4.3. RLogic with transformer based attention

As we are learning a sequence of relations, the latest advancements like attention based approach should improve the target relation prediction. Inspired from (Vaswani et al., 2017) (Attention is all you need), we have added attention layer in RLogic before it predicts the target relation at each step. In every target relation prediction step, we pass the output context embedding into the transformer attention layer. Inside transformer attention layer we obtain key and

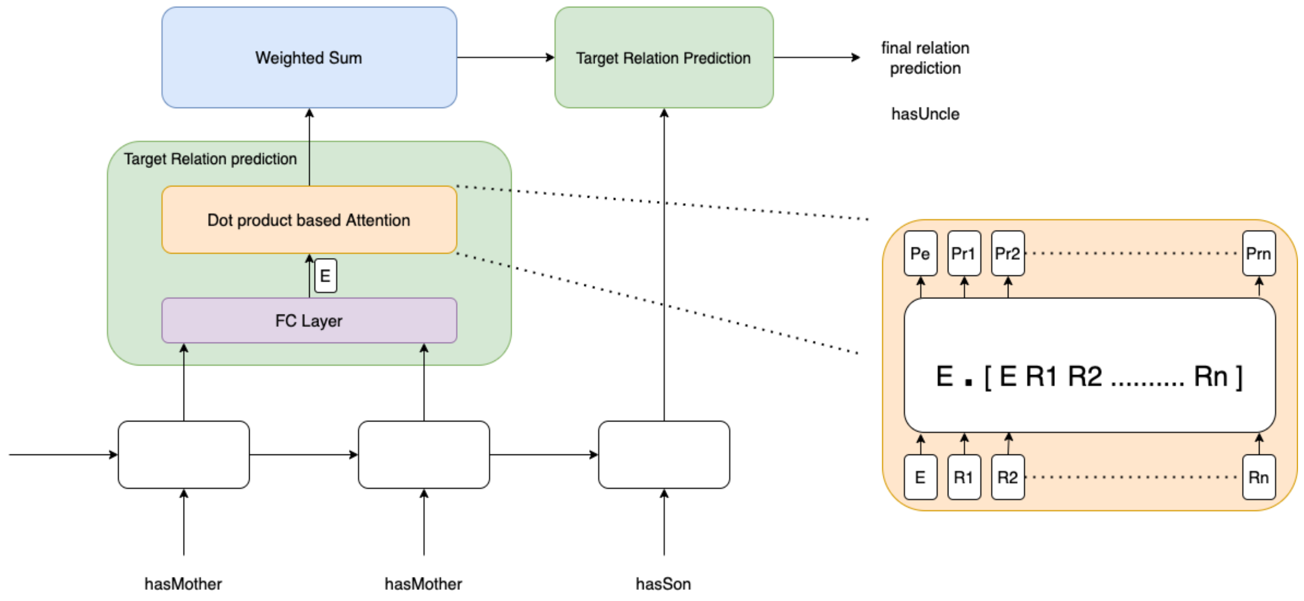
Weighted Sum
Target Relation Prediction
final relation prediction
hasUncle
Target Relation prediction
Dot product based Attention
E
FC Layer
Pe
Pr1
Pr2
Prn
E . [E R1 R2 .... Rn]
E
R1
R2
Rn
hasMother
hasMother
hasSon

Figure 5. Illustration of the RLogic framework with naive attention.

query vectors corresponding to every relation and we apply similar logic as in (Vaswani et al., 2017) to calculate the attention of context vector with every other relation. And normalizing these attention values with a softmax will give the probabilites as below:

$$\begin{aligned} \mathrm{e}_i &= FC\_layer(e_{i-1}, r_i) \\ \mathrm{q}_i &= FC_q(e_i) \\ \mathrm{K}_i &= FC_k([r_1, r_2, ...r_n]) \\ \mathrm{Pr}_i &= softmax(q_i K_i^T) \end{aligned}$$

where $K_i = [k_1, k_2, ...k_n]$, the key vectors of every relation and *FC_layer* is used to combine the previous context with the current relation embedding and Pr_i is the final probabilities corresponding to relations after step *i*. By adding this transformer attention layer, the model will be able to understand the global context at every step and thus should result in better target relation prediction. One can also experiment with add more FC layers in the attention to learn more complex key and query vectors to better estimate the attention.

5. Experiments

5.1. Knowledge Graph Completion

Knowledge Graph Completion is a widely used task for evaluating the learned rules. Knowledge graph G is represented as a collection of triples $\{(h, r, t)\} \subseteq E \times R \times E$, where E and R are the entity set and relation set respectively. The task of Knowledge Graph Completion is to predict the tail entity given the head entity and the query relation: (h, r, ?).

We use forward chaining(Rules et al., 1997) which can used to derive missing facts from logical rules efficiently. Given a query (*h, r*, ?), let $\mathcal{A}$ be the set of candidate answers which can be discovered by any learned rule using forward chaining. For each candidate answer $e \in \mathcal{A}$, the score of triple (*h, r, e*) an be calculated as $\sum_{\delta \in rulespace} \sum_{path \in P(h,r,e)} s(\delta)$. The conjunctive body of chain-like Horn rules is essentially a path in a KG and can be extracted efficiently using sparse matrix multiplication. We have selected top 1600 rules with the highest score predicted by our approaches for the KG completion task.

To evaluate our results we use two evaluation metrics, MRR and Hit@10.

- Mean Reciprocal Rank (MRR): the average of the reciprocal of the ranks of all positives (higher is better, best is 1).

- Hits@10: the fraction of positives that rank in the top 10 among their negatives (higher is better, best is 1).

5.2. Datasets

Three popular datasets were used for evaluation of our proposed logic. WN18-RR (Dettmers et al., 2018), FB15K-237 (Toutanova & Chen, 2015) and Family (Hinton et al., 1986). Below is the dataset statistics.

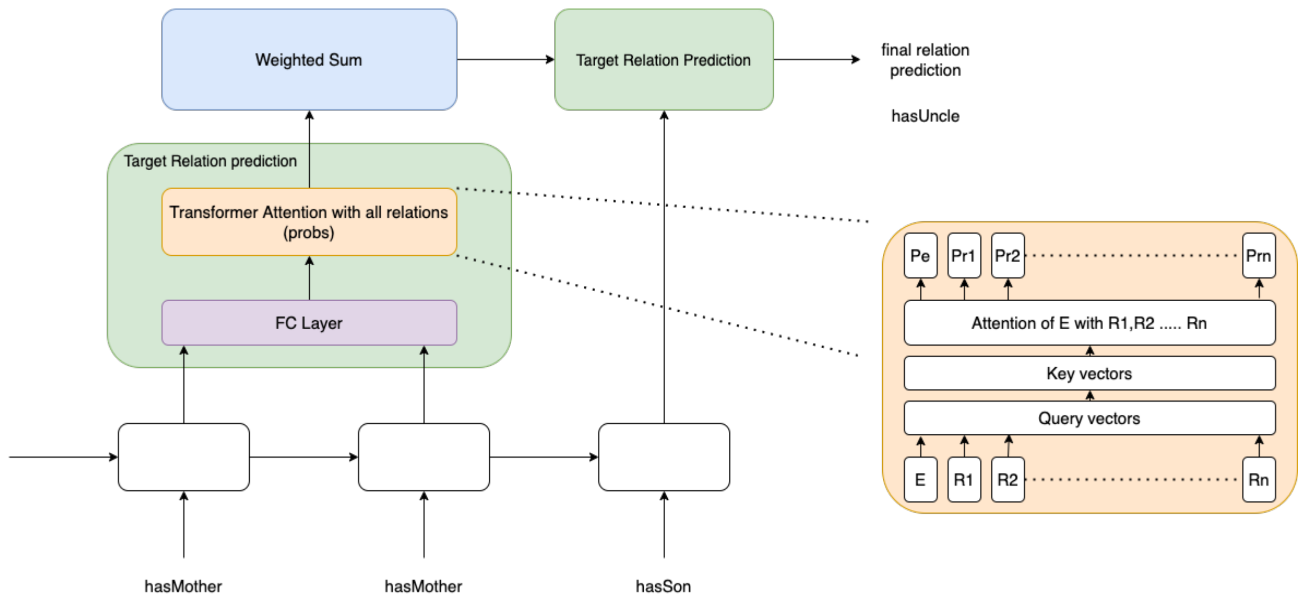
Weighted Sum
Target Relation Prediction
final relation prediction
hasUncle
Target Relation prediction
Transformer Attention with all relations (probs)
FC Layer
hasMother
hasMother
hasSon
Pe
Pr1
Pr2
Prn
Attention of E with R1,R2 ..... Rn
Key vectors
Query vectors
E
R1
R2
Rn

Figure 6. Illustration of the RLogic framework with transformer based attention.

Dataset	Data	Relation	Entity
FB15K-237	310,116	237	14,541
WN18RR	93,003	11	40,943
Family	28,356	12	3007

Table1: Datasets description

FB15K237, WN18RR are the most widely used benchmark datasets for KGE models, which don't suffer from test triple leakage in the training set. Family is used for its interpretability. Since inverse relations are required to learn rules, we preprocess the KGs to add inverse links.

- WN18RR is a link prediction dataset created from WN18, which is a subset of WordNet. WN18 consists of 18 relations and 40,943 entities. The evaluation dataset of WN18RR does not have inverse relation test leakage. In summary, WN18RR dataset contains 93,003 triples with 40,943 entities and 11 relation type. Its entities correspond to word senses, and relationships define lexical relations between them.

- The FB15k dataset contains knowledge base relation triples and textual mentions of Freebase entity pairs. It is an online collection of structured data harvested from many sources, including individual, user-submitted wiki contribution. It has a total of 592,213 triplets with 14,951 entities and 1,345 relationships.

- Family contains family relationships among members of a family.

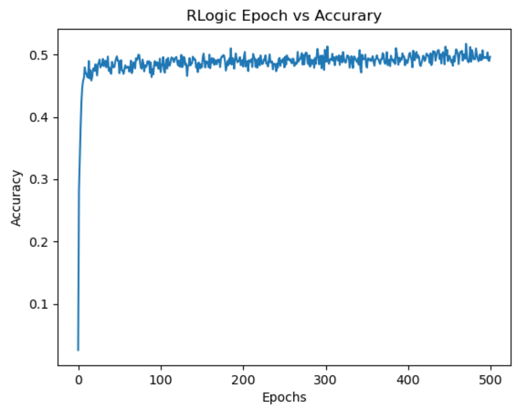

Figure 7. Epoch vs Accuracy for RLogic

5.3. Results

We have run knowledge graph completion tasks on Family, FB15k-237 and WN18RR datasets. Due to the computation limitations we have to limit the number of rules applied for knowledge graph completion on FB15k-237 and WN18RR datasets. We have selected top 1600 predicted rules for each of the model RLogic(baseline), RLogic with naive attention, Rlogic with Advanced attention and RLogic GCN. We observe from the table that our methods are comparable to the baseline RLogic. On Family dataset, with naive attention and Advanced attention we were able to achieve slighly better results as compared the original RLogic model. And on

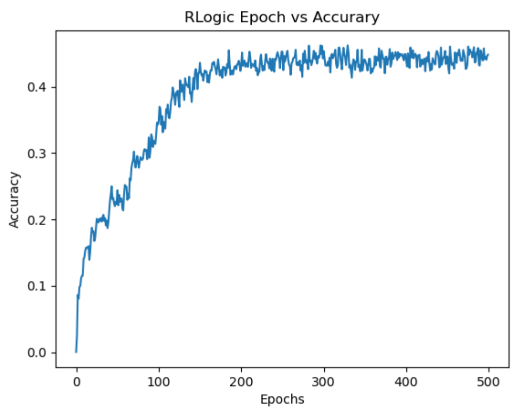

Figure 8. Epoch vs Accuracy for RLogic with Naive Attention

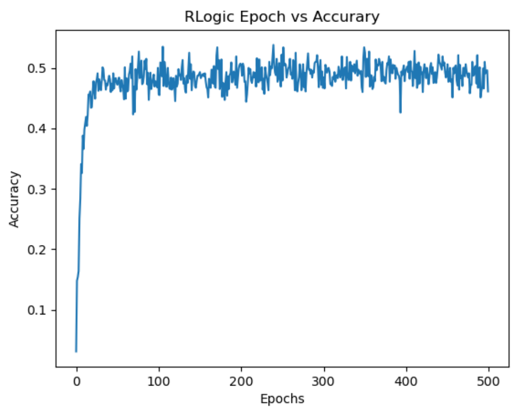

Figure 9. Epoch vs Accuracy for RLogic with Transformer Attention

FB15K-237 we were able to see considerable improvement with even 10000 rules applied out of 1M rules. Given more computation power, adding more rules should definitely improve the results further more. Similary with our newly proposed method RLogic GCN, we are able to obtain on par results with RLogic original model with fewer training time.

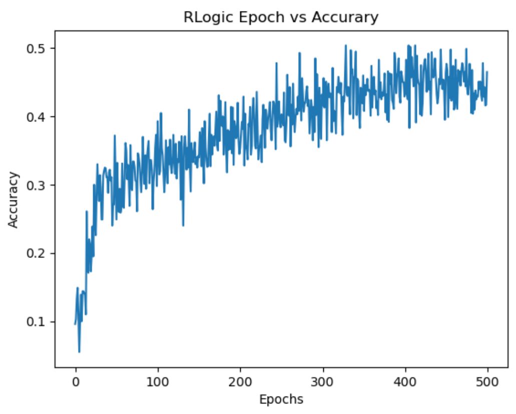

Figure 10. Epoch vs Accuracy for RLogic with GCN

Model	Dataset	MRR	Hits@10
Rlogic	Family	0.857	0.9527
	FB15K-237[1]	0.056	0.079
	WN18RR[2]	0.0563	0.109
Rlogic Naive Attn	Family	0.858	0.9527
	FB15K-237[1]	0.0536	0.140
	WN18RR[2]	0.0507	0.107
Advanced Attn	Family	0.856	0.953
	FB15K-237[1]	0.146	0.356
	WN18RR[2]	0.0465	0.097
Rlogic GCN	Family	0.475	0.602
	FB15K-237[1]	0.1646	0.2875
	WN18RR[2]	-	-

Table2: MRR and Hit@10 results for RLogic, RLogic Naive Attn, Rlogic Advanced Attn and RLogic GCN models. All the results are capture on rules of length 2.

6. Conclusion & Future Work

All existing methods rely on the complete set of rule instances for rule evaluation and thus suffer from severe computational inefficiency. Rlogic solves this issue by learning rules directly at the schema level. In addition, RLogic incorporates one of the most significant properties of logical rules, the deductive nature, into rule learning. We implemented attention methods in RLogic and evaluated the results for a variety of different datasets. We also proposed a GCN based encoding method for RLogic and the results we got are comparable if not better to the baseline. We can improve the results further by using attention based techniques for GCN. We can also further improve the RLogic framework by incorporating relation text in the training process. Cur-

[1]Evaluated on 100 test cases with top 10000 (1%) out of 10M predicted rules

[2]Evaluated on 1000 test cases with top 500(10%) out of 5K predicted rules

rently, the relation text is not taken into account and may result in a semantic gap. We can utilize mBART ((Liu et al., 2020)) based embeddings for solving that issue. We also saw the need to scale knowledge graph completion using rules and therefore an exploration into that would also deem useful.

7. Task Distribution

Task Name	Task Done By
Literature Survey	All
Baseline Codes	All
Naive attention	Nilay
Advanced Attention	Manish
GCN Method	All
KG Completion	Aditya
Evaluation	All
Report Writing	All

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, 2014. URL https://arxiv.org/abs/1409.0473.

Dettmers, T., Pasquale, M., Pontus, S., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pp. 1811–1818, February 2018. URL https://arxiv.org/abs/1707.01476.

Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pp. 413–422, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351. doi: 10.1145/2488388.2488425. URL https://doi.org/10.1145/2488388.2488425.

Hinton, G. E. 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.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2016. URL https://arxiv.org/abs/1609.02907.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. Multilingual denoising pre-training for neural machine translation, 2020. URL https://arxiv.org/abs/2001.08210.

Muggleton, S. and de Raedt, L. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994. ISSN 0743-1066. doi: https://doi.org/10.1016/0743-1066(94)90035-3. URL https://www.sciencedirect.com/science/article/pii/0743106694900353. Special Issue: Ten Years of Logic Programming.

Nienhuys-Cheng, S.-H., Wolf, R. d., Siekmann, J., and Carbonell, J. G. *Foundations of Inductive Logic Programming*. Springer-Verlag, Berlin, Heidelberg, 1997. ISBN 3540629270.

Rules, C., Salvat, E., and Mugnier, M.-L. Sound and complete forward and backward chainings of graph rules. *Lecture Notes in AI*, 1115, 08 1997.

Toutanova, K. and Chen, D. 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, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4007. URL https://aclanthology.org/W15-4007.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base reasoning, 2017. URL https://arxiv.org/abs/1702.08367.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method, 2019. URL https://arxiv.org/abs/1907.04931.

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., and Gu, Q. Layer-dependent importance sampling for training deep and large graph convolutional networks, 2019. URL https://arxiv.org/abs/1911.07323.