Perform Like an Engine: A Closed-Loop Neural-Symbolic Learning Framework for Knowledge Graph Inference

Guanglin Niu[1], **Bo Li**[1,2]*, **Yongfei Zhang**[3,4], **Shiliang Pu**[5]
[1] Institute of Artificial Intelligence, Beihang University, Beijing, China
[2] Hangzhou Innovation Institute, Beihang University, Hangzhou, China
[3] Beijing Key Laboratory of Digital Media, Beihang University, Beijing, China
[4] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China [5] Hikvision Research Institute, Hangzhou, China
{beihangngl, boli, yfzhang}@buaa.edu.cn, pushiliang.hri@hikvision.com

Abstract

Knowledge graph (KG) inference aims to address the natural incompleteness of KGs, including rule learning-based and KG embedding (KGE) models. However, the rule learning-based models suffer from low efficiency and generalization while KGE models lack interpretability. To address these challenges, we propose a novel and effective closed-loop neural-symbolic learning framework **EngineKG** via incorporating our developed KGE and rule learning modules. KGE module exploits symbolic rules and paths to enhance the semantic association between entities and relations for improving KG embeddings and interpretability. A novel rule pruning mechanism is proposed in the rule learning module by leveraging paths as initial candidate rules and employing KG embeddings together with concepts for extracting more high-quality rules. Experimental results on four real-world datasets show that our model outperforms the relevant baselines on link prediction tasks, demonstrating the superiority of our KG inference model in a neural-symbolic learning fashion. The source code and datasets of this paper are available at https://github.com/ngl567/EngineKG.

1 Introduction

Typical knowledge graphs (KGs) store triple facts and some of them also contain concepts of entities (Bollacker et al., 2008). The KGs have proven to be incredibly effective for a variety of applications such as dialogue system (Zhou et al., 2018) and question answering (Huang et al., 2019). However, the existing KGs are always incomplete which restricts the performance of knowledge-based applications. Thus, KG inference plays a vital role in completing KGs for better applications of KGs.

The existing KG inference approaches are usually classified into two main categories: (1) Rule learning-based models such as AMIE+ (Galárraga et al., 2015) and AnyBurl (Meilicke et al., 2019) mine rules from KGs and employ these rules to predict new triples by deduction. However, rule learning-based models suffer from **low efficiency** of the rule mining process and the **poor generalization** caused by the limited coverage of inference patterns. (2) KGE technique learns the embeddings of entities and relations to predict the missing triples via scoring each triple candidate, including TransE (Bordes et al., 2013), HAKE (Zhang et al., 2020) and DualE (Cao et al., 2021). The previous KGE models perform in a data-driven fashion, contributing to good efficiency and generalization but **lacking interpretability**.

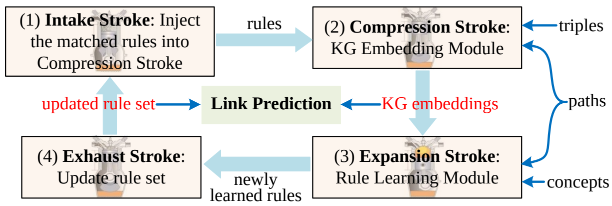

Figure 1: The brief architecture of our closed-loop framework for KG inference EngineKG that performs like a four-stroke engine.

Some recent researches attempt to combine the advantages of rule learning-based and KGE-based models to complement each other in a **neural-symbolic learning fashion**. An idea is to introduce logic rules into KGE models, such as RUGE (Guo et al., 2018) and its advanced model IterE (Zhang et al., 2019b). These approaches all convert the rules into formulas by t-norm based fuzzy logic to obtain newly labeled triples. However, these models **cannot maintain the interpretability which is a vital feature of symbolic rules**. On the other hand, some rule learning-based models succeed in leveraging KG embeddings to extract rules via numerical calculation rather than discrete graph search, including RNNlogic (Qu et al., 2021), RLvLR (Omran et al., 2019), DRUM (Sadeghian et al., 2019) and RuLES (Ho et al., 2018). Although

*Corresponding author.

Proceedings of the 29th International Conference on Computational Linguistics, pages 1391–1400
October 12–17, 2022.

the efficiency of mining rules is improved, **the performance especially generalization of purely employing rules to implement KG inference is still limited**.

To address the above challenges, we propose a closed-loop neural-symbolic learning framework **EngineKG** via combining an embedding-based rule learning and a rule-enhanced KGE, in which paths and concepts are utilized. Our model is named **EngineKG** because it performs like an engine as shown in Figure 1: (1) **Intake Stroke**. The closed-path rules (or named chain rules) are injected into the KGE module (analogous to intake) to guide the procedure of learning KG embeddings, where the initial seed rules are mined by any rule learning tool, and the rule set would grow via our designed rule learning module from the first iteration. (2) **Compression Stroke**. The KGE module leverages the rules and paths to learn the low-dimensional embeddings (analogous to compression) of entities and relations, improving interpretability and accuracy. (3) **Expansion Stroke**. The novel rule learning module outputs newly learned rules (analogous to exhaust) by the effective rule pruning strategy based on paths, relation embeddings and concepts. (4) **Exhaust Stroke**. Update the rule set (analogous to exhaust) by merging the previous rule set and the newly learned rules for boosting KGE and KG inference in the next iteration.

Our research makes three contributions:

- We propose a novel and effective closed-loop neural-symbolic learning framework that performs embedding-based rule learning and rule-enhanced KGE iteratively, balancing good accuracy, interpretability and efficiency.
- Paths and ontological concepts are well exploited for supplementing the valuable semantics to both KGE and rule learning, facilitating the better performance of KG inference.
- The link prediction results and the effectiveness of rule learning on four datasets illustrate that our model outperforms various state-of-the-art KG inference approaches.

2 Related Work

2.1 Rule Learning-Based Models

According to the symbolic characteristics of KG, some rule learning techniques specific to KGs are applied to KG inference with relatively good accuracy and interpretability, including AMIE+ (Galárraga et al., 2015), Anyburl (Meilicke et al., 2019), DRUM (Sadeghian et al., 2019), RLvLR (Omran et al., 2019) and RNNLogic (Qu et al., 2021). AMIE+ (Galárraga et al., 2015) introduces optimized query writing techniques into traditional inductive logic programming algorithms to generate horn rules efficiently. Anyburl learns closed-path rules from KGs in a reinforcement learning framework. DRUM, RLvLR and RNNLogic employ KG embeddings for enhancing the efficiency and scalability of rule learning. Whereas, all the previous rule learning algorithms lack generalization since the number of rules mined at one time is limited.

2.2 KG Embedding Models

The typical KG embedding (KGE) models learn the embeddings of entities and relations to measure the plausibility of each triple. TransE (Bordes et al., 2013) regards the relations as translation operations from head to tail entities. ComplEx (Trouillon et al., 2016) embeds the KG into a complex space while DualE (Cao et al., 2021) embeds relations into the quaternion space to model the symmetric and antisymmetric relations. HAKE (Zhang et al., 2020) embeds entities into the polar coordinate system and is able to model the semantic hierarchies of KGs. RUGE (Guo et al., 2018) and IterE (Zhang et al., 2019b) both convert rules into formulas by t-norm fuzzy logic to infer newly labeled triples. Particularly, IterE iteratively conducts rule learning and KG embedding, but **the significant distinctions between our model EngineKG and IterE** include: (1) **Usage of rules**: our model leverages rules to compose paths for learning KG embeddings while IterE uses rules to produce labeled triples. Meanwhile, we maintain the interpretability of symbolic rules, while IterE does not. (2) **Additional information**: our model introduces paths and concepts into both rule learning and KG embedding while IterE simply depends on triples.

2.3 Path-Enhanced Models

In terms of the graph structure of KGs, paths denote the associations between entities apart from relations and are applied to multi-hop reasoning (Lin et al., 2018; Xiong et al., 2017; Neelakantan et al., 2015). PTransE (Lin et al., 2015) extends TransE by measuring the similarity between relation and path embeddings. MultiHopKG (Lin et al., 2018) explores the answer entities via searching corresponding paths with reinforcement learning. However, these models represent paths in a data-driven fashion, lacking interpretability and accuracy.

3 Methodology

In this section, we first describe the problem formulation and notation of our work in section 3.1. Then, following the workflow of EngineKG as shown in Figure 2, we introduce the rule-enhanced KGE module in section 3.2 and the embedding-based rule learning module in section 3.3.

3.1 Problem Formulation and Notation

Definition of Closed-Path Rule. The closed-path (CP) rule or named chain rule is a fragment of the horn rule, which we are interested in for the KGE module and the inference. A CP rule is of the form

$$Head(x, y) \Leftarrow B_1(x, z_1) \wedge B_2(z_1, z_2) \wedge \cdots \wedge B_n(z_{n-1}, y) \quad (1)$$

where $B_1(x, z_1)$, $B_2(z_1, z_2)$, $\cdots$, $B_n(z_{n-1}, y)$ denote the atoms in the rule body $Body(x, y)$, and $Head(x, y)$ is the rule head. B_i and $Head$ indicate relations. Standard confidence (SC) and head coverage (HC) are two predefined statistical measurements to assess rules (Galárraga et al., 2015; Omran et al., 2019), which are defined as follows:

$$Support = \#(e, e') : Body(e, e') \wedge Head(e, e') \quad (2)$$

$$SC = \frac{Support}{\#(e, e') : Body(e, e')} \quad (3)$$

$$HC = \frac{Support}{\#(e, e') : Head(e, e')} \quad (4)$$

where $\#(e, e')$ indicates the number of entity pairs (e, e') that satisfy the condition on the right side of the colon. In general, the rules with SC and HC both higher than 0.7 are regarded as high-quality rules (Zhang et al., 2019b).

Definition of Path. A path between an entity pair (h, t) is in the form of $[h \rightarrow r_1 \rightarrow e_1 \rightarrow \cdots \rightarrow r_n \rightarrow t]$ where r_i and e_i are the intermediate relation and entity, and the length of a path is the number of the intermediate relations.

3.2 Rule-Enhanced KGE Module

We aim to learn the entity and relation embeddings from triple facts, rules and paths via neural-symbolic learning. Firstly, we extract the paths via PCRA algorithm (Lin et al., 2015). Apart from other path-finding approaches such as PRA (Lao et al., 2011), PCRA algorithm could measure the reliability of each path for KGE module. Particularly, we develop a joint logic and data-driven path representation mechanism to learn path embeddings.

Logic-Driven Path Representation (Intake Stroke). The CP rules could compose paths into shorter and more accurate ones for enhancing the representation of paths. For instance, a length-2 path $[The\ Pursuit\ of\ Happiness \xrightarrow{CastActor} Will\ Smith \xrightarrow{PersonLanguage} English]$ as shown in Figure 2 could be composed into a shorter path (actually a triple) $[The\ Pursuit\ of\ Happiness \xrightarrow{TVLanguage} English]$ via the CP rule $TVLanguage(x, y) \Leftarrow CastActor(x, z) \wedge PersonLanguage(z, y)$. Furthermore, the relation $TVLanguage$ could signify the original multi-hop path.

Data-Driven Path Representation. For the scenario that the path cannot be further composed by rules such as the path $[The\ Pursuit\ of\ Happiness \xrightarrow{CountryOfOrigin} U.S.A. \xrightarrow{LanguageSpoken} English]$ in Figure 2, we represent this path by summing all the relation embeddings along the path. With the entity pair (h, t) together with the linking path set $\mathcal{P}$, the energy function for measuring the plausibility of the path-specific triple $(h, t, \mathcal{P})$ is designed as

$$E_p(h, t, \mathcal{P}) = \sum_{pi \in \mathcal{P}} \frac{R(p_i|h, t)}{\sum_{pi \in \mathcal{P}} R(p_i|h, t)} \|\mathbf{h} + \mathbf{p}_i - \mathbf{t}\| \quad (5)$$

where $\mathbf{h}$ and $\mathbf{t}$ are the head and tail entity embeddings. p_i denotes the i-th path in the path set $\mathcal{P}$ and $\mathbf{p}_i$ is the embedding of p_i achieved by the joint logic and data-driven path representation. $R(p_i|h, t)$ indicates the reliability of path p_i between the given entity pair (h, t) obtained by the PCRA algorithm.

Optimization Objective (Compression Stroke). Along with the translation-based KGE models, the energy function for formalizing the plausibility of a triple fact (h, r, t) is given as

$$E_t(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\| \quad (6)$$

in which $\mathbf{r}$ is the embedding of the relation r.

The existing KGE techniques neglect the semantic association between relations. Remarkably, the length-1 rules model the causal correlations between two relations. As shown in Figure 2, the relation pair in the rule $FilmLanguage(x, y) \Leftarrow TVLanguage(x, y)$ should have higher similarity

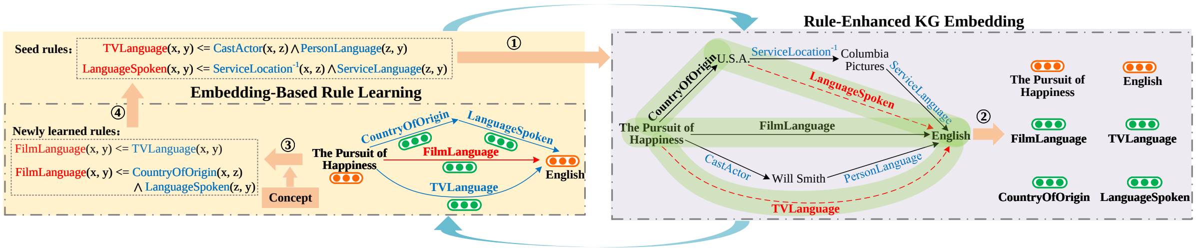

Figure 2: The overall architecture of our developed KG inference model EngineKG in a closed-loop neural-symbolic learning framework. Specific to the rule-enhanced KG embedding module, the green highlighted parts contain the triples and the composed paths via rules, indicating the inputs of the KGE module.

than other relations. Thus, we measure the association between relation pairs as

$$E_r(r_1, r_2) = \|\mathbf{r}_1 - \mathbf{r}_2\| \tag{7}$$

where $\mathbf{r}_1$ and $\mathbf{r}_2$ are the embeddings of relations r_1 and r_2. $E_r(r_1, r_2)$ should be closer to a small value if r_1 and r_2 appear in a length-1 rule at the same time.

With the energy functions specific to the factual triple, the path representation and the relation correlation, the joint loss function for training is designed as follows:

$$L = \sum_{(h,r,t)\in\mathcal{T}} (L_t + \alpha_1 L_p + \alpha_2 L_r) \tag{8}$$

$$L_t = \sum_{(h',r,t')\in\mathcal{T}'} [\gamma_1 + E_t(h,r,t) - E_t(h',r,t')]_+ \tag{9}$$

$$L_p = \sum_{(h',t')\in\mathcal{T}'} [\gamma_2 + E_p(h,t,P) - E_p(h',t',P)]_+ \tag{10}$$

$$L_r = \sum_{r_p\in\mathcal{S}} \sum_{r_n\in\mathcal{S}'} [\gamma_3 + E_r(r,r_p) - E_r(r,r_n)]_+ \tag{11}$$

where L is the whole training loss consisting of three components: the triple-specific loss L_t, the path-specific loss L_p, and the relation correlation-specific loss L_r. α_1 and α_2 are the weights of paths and relation correlation, respectively. γ_1, γ_2 and γ_3 are three margins in each loss function. $[x]_+$ is the function returning the maximum value between 0 and x. $\mathcal{T}$ is the set of triples observed in the KG and $\mathcal{T}'$ is the set of negative samples obtained by random negative sampling. $\mathcal{S}$ is the set of positive relations that are correlated with relation r by length-1 rules and $\mathcal{S}'$ is the set of negative relations beyond $\mathcal{S}$ and relation r.

We employ mini-batch Stochastic Gradient Descent (SGD) algorithm to optimize the joint loss function for learning entity and relation embeddings. The entity and relation embeddings are initialized randomly and constrained to be unit vectors by the additional regularization term with L2 norm.

3.3 Embedding-Based Rule Learning Module

We develop an embedding-based rule learning (Expansion Stroke) to mine high-quality CP rules via conducting the rule searching and the rule pruning efficiently. Remarkably, a path can naturally represent the body of a CP rule. Motivated by this observation, we firstly reuse the paths extracted in section 3.2 and regard these paths as candidate CP rules, which improves the efficiency of rule searching. For instance, given an entity pair (h, t) connected by a relation r and a path $[h \to r_1 \to e_1 \to r_2 \to e_2 \to, \cdots \to e_{n-1} \to r_n \to t]$, it can be deduced as a CP rule $r(x, y) \Leftarrow r_1(x, z_1) \wedge r_2(z_1, z_2) \wedge \cdots \wedge r_n(z_{n-1}, y)$, where x, y and $z_i (i = 1, \cdots, n-1)$ are the variables in the rule, and $r_i (i = 1, \cdots, n)$ is a relation.

To evaluate the plausibility of candidate CP rules efficiently, we develop a novel rule pruning strategy consisting of two components: **Embedding-based Semantic Relevance** and **Concept-based Co-occurrence**. It should be noted that the Concept-based Co-occurrence is available when the KG contains concepts. For the KGs without concepts, employing Embedding-based Semantic Relevance solely is still valid to learn rules.

Embedding-based Semantic Relevance. Intuitively, a candidate rule is plausible if the rule body corresponding to a path p is semantically relevant to the rule head corresponding to the relation r. We focus on the paths and the CP rules with lengths no longer than 2 for the trade-off of efficiency and performance. Based on the KG embeddings learned in our KGE module, we could measure the semantic relevance between the body and the head of a candidate rule by the path embedding and relation embedding as well as the score function as

$$E_{sr}(r, p) = \exp(-\|\mathbf{r} - \mathbf{p}\|) \tag{12}$$

where $\mathbf{p}$ denotes the embedding of the path p.

Dataset	#Relation	#Entity	#Concept	#Train	#Valid	#Test
FB15K	1,345	14,951	89	483,142	50,000	59,071
FB15K237	237	14,505	89	272,115	17,535	20,466
NELL-995	200	75,492	270	123,370	15,000	15,838
DBpedia-242	298	99,744	242	592,654	35,851	30,000

Table 1: Statistics of the experimental datasets.

The embedding-based semantic relevance indicates a global plausibility of a rule from the perspective of relations. Furthermore, a concept-based co-occurrence is proposed to evaluate the local relevance of the arguments in a rule.

Concept-based Co-occurrences. The neighbor arguments in a high-quality CP rule are expected to share as many same concepts as possible. Given a CP rule $Nationality(x, y) \Leftarrow BornIn(x, z) \wedge LocatedIn(z, y)$, the tail argument of relation $BornIn$ and the head argument of relation $LocatedIn$ should share the concept $Location$. Considering there are far fewer concepts than entities, we encode each concept as a one-hot representation to maintain the precise concept features. The concept embedding of the head or tail argument of an atom can be formalized as

$$AC_h(r) = \frac{1}{|C_h(r)|} \sum_{c \in C_h(r)} OH(c) \tag{13}$$

$$AC_t(r) = \frac{1}{|C_t(r)|} \sum_{c \in C_t(r)} OH(c) \tag{14}$$

where $AC_h(r)$ and $C_h(r)$ are the concept embedding and concept set in the head argument of an atom containing relation r while $AC_t(r)$ and $C_t(r)$ are that of in the tail argument. $OH(c)$ denotes the one-hot representation of the concept c.

Specific to a CP rule in the form of $r(x, y) \Leftarrow r_1(x, z_1) \wedge r_2(z_1, z_2) \wedge \cdots \wedge r_n(z_{n-1}, y)$, three types of co-occurrence score functions are designed according to the different positions of the overlapped arguments:

$$E^h_{co}(r, r_1) = sim(AC_h(r), AC_h(r_1)) \tag{15}$$

$$E^t_{co}(r, r_n) = sim(AC_t(r), AC_t(r_n)) \tag{16}$$

$$E^i_{co}(r_i, r_{i+1}) = sim(AC_t(r_i), AC_h(r_{i+1})) \tag{17}$$

where $E^h_{co}(r, r_1)$ and $E^t_{co}(r, r_n)$ respectively denote the co-occurrence similarities specific to the head arguments and the tail arguments between the rule head and the rule body. $E^{in}_{co}(r_i, r_{i+1})$ represents the co-occurrence similarity between the adjacent arguments in the rule body. $sim(x, y)$ represents the cosine distance function for measuring the similarity between x and y.

Then, the whole co-occurrence score function can be achieved by composing all the scores in Eqs. 15-17 as

$$\begin{aligned} E_{co}(r, p) = E^h_{co}(r, r_1) + E^t_{co}(r, r_n) \\ + \sum_{i=1}^{n-1} E^i_{co}(r_i, r_{i+1}) \end{aligned} \tag{18}$$

Consequently, the overall score function for evaluating candidate rules is defined as:

$$E_{cg} = E_{sr}(r, p) + \beta E_{co}(r, p) \tag{19}$$

where β is the weight of the co-occurrence score. We set a threshold and select the candidate rules with the scores calculated by Eq. 19 above the threshold as filtered candidate rules. Afterward, we output the high-quality rules from the filtered candidate rules that satisfy the thresholds of the precise quality criteria namely standard confidence and head coverage defined in Eqs. 2-4. Then, the updated rule set is obtained via fusing the newly learned rules and the previous rule set (Exhaust Stroke) for the KGE module in the next iteration.

3.4 Algorithm Flow and Complexity

It is noteworthy that from the first iteration of EngineKG, our rule learning module could potentially achieve sustainable growth of rules. The entire iteration process will keep running until no fresh rules can be generated. Then, the learned KG embeddings learned in the last iteration are exploited for the KG inference. The Algorithm 1 summarizes the whole closed-loop KG inference procedure of our EngineKG model.

To evaluate the complexity of our EngineKG model, we denote n_e, n_r, n_p, n_c and n_t as the amount of entities, relations, paths, concepts and

Algorithm 1: Training framework of our model EngineKG

Input: $\mathcal{G}$: Training set
$\mathcal{C}_h$, $\mathcal{C}_t$: The head and tail concept set associated with relations
$\mathcal{P}$: The set of paths extracted from $\mathcal{G}$ via PCRA
γ_1, γ_2, γ_3: The margins in loss functions
α_1, α_2: The weights for trade-off
st: The score threshold for coarse-grained evaluation of rules
Max_e: The maximum epochs

Initialize entity embeddings **e** and relation embeddings **r** randomly and encode the concept embeddings from $\mathcal{C}_h$ and $\mathcal{C}_t$ by one-hot representation;
Mine rules by a rule mining tool such as AMIE+;
while *new rules can be learned* **do**
  for *epoch=1,2,...,* Max_e **do**
    Sample a minibatch of triples $\mathcal{T}$ from $\mathcal{G}$;
    Compose the paths between the entity pairs in $\mathcal{T}$ by the logic and data-driven path representation described in section 3.2;
    Generate the set of negative samples $\mathcal{T}'$ by the random negative sampling as in TransE (Bordes et al., 2013);
    Update **e** and **r** by optimizing the loss functions in Eqs. 5-11;
  Generate the initial candidate rules from the paths in $\mathcal{P}$;
  Load the learned entity and relation embeddings **e** and **r**;
  Calculate the coarse-grained evaluation score E_{cg} of each candidate rule according to Eqs. 12-19;
  if E_{cg} *of a rule is smaller than* st **then**
    Eliminate this rule;
  Pick out the candidate rules that satisfy the thresholds of the standard confidence and the head coverage;
  Output the newly learned rules;
  Update the rule set by merging the newly learned rules and the rule set in the last iteration;

triples in a KG. The average length of paths is l_p. The embedding dimension of both entities and relations is represented as d. The embedding dimension of concepts is n_c due to the one-hot encoding applied for concept representations. Our model complexity of parameter sizes is $\mathcal{O}(n_e d + n_r d + n_c^2)$. For each iteration in training, the time complexity of our model is $\mathcal{O}(n_t n_p l_p d)$.

4 Experiments

4.1 Experimental Setup

Datasets. Four datasets containing ontological concepts are employed for our experiments, including FB15K (Bordes et al., 2013), FB15K237 (Toutanova and Chen, 2015), NELL-995 and DBpedia-242. Particularly, NELL-995 here is a re-split of the original dataset (Xiong et al., 2017) into training/validation/test sets. DBpedia-242 is generated from the commonly-used KG DBpedia (Lehmann et al., 2015) to ensure each entity in the dataset has a concept. The statistics of the experimental datasets are listed in Table 1.

Baselines. We compare our model EngineKG with two categories of baselines:

(1) The traditional KGE models depending on triple facts: TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2019), QuatE (Zhang et al., 2019a), HAKE (Zhang et al., 2020) and DualE (Cao et al., 2021).

(2) The models using paths or rules: the path-based model MultiHopKG (Lin et al., 2018), the rule learning-based models RNNLogic (Qu et al., 2021) and RPJE (Niu et al., 2020), and the model combining rules with KG embeddings IterE (Zhang et al., 2019b).

The evaluation results of these baselines are obtained by employing their open-source codes with the suggested hyper-parameters.

Training Details. We implement our model in C++ and on an Intel i9-9900 CPU with a memory of 64G. For a fair comparison, the embedding dimension of all the models is fixed as 100, the batch size is set to 1024 and the number of negative samples is set to 10. Specific to our model, during each iteration, the maximum training epoch is set to 1000, and the standard confidence and the head coverage are selected as 0.7 and 0.1 for better performance. The entity and relation embeddings are initialized randomly. We employ grid search for selecting the best hyper-parameters on the validation dataset.

Models	FB15K					FB15K237				
	MR	MRR	Hits@10	Hits@3	Hits@1	MR	MRR	Hits@10	Hits@3	Hits@1
TransE (Bordes et al., 2013)	117	0.534	0.775	0.646	0.386	228	0.289	0.478	0.326	0.193
ComplEx (Trouillon et al., 2016)	197	0.346	0.593	0.405	0.221	507	0.236	0.406	0.263	0.150
RotatE (Sun et al., 2019)	39	0.612	0.816	0.698	0.488	168	0.317	0.553	0.375	0.231
QuatE (Zhang et al., 2019a)	40	0.765	0.878	0.819	0.693	173	0.312	0.495	0.344	0.222
HAKE (Zhang et al., 2020)	42	0.678	0.839	0.761	0.570	183	0.344	0.542	0.382	0.246
DualE (Cao et al., 2021)	43	0.759	0.882	0.820	0.681	202	0.332	0.522	0.367	0.238
MultiHopKG (Lin et al., 2018)	-	0.670	0.769	0.708	0.612	-	0.385	0.562	0.429	0.298
RNNLogic (Qu et al., 2021)	244	0.496	0.669	0.544	0.405	620	0.280	0.428	0.306	0.205
RPJE (Niu et al., 2020)	40	0.811	0.898	0.832	0.762	207	0.443	0.579	0.479	0.374
IterE (Zhang et al., 2019b)	85	0.577	0.807	0.663	0.451	463	0.210	0.355	0.227	0.139
EngineKG (Ours)	**20**	**0.854**	**0.933**	**0.885**	**0.810**	**121**	**0.555**	**0.707**	**0.590**	**0.479**

Models	DBpedia-242					NELL-995				
	MR	MRR	Hits@10	Hits@3	Hits@1	MR	MRR	Hits@10	Hits@3	Hits@1
TransE (Bordes et al., 2013)	1996	0.256	0.539	0.395	0.075	8650	0.167	0.354	0.219	0.068
ComplEx (Trouillon et al., 2016)	3839	0.196	0.387	0.230	0.104	11772	0.169	0.298	0.185	0.106
RotatE (Sun et al., 2019)	1323	0.308	0.594	0.422	0.143	9620	0.292	0.444	0.325	0.216
QuatE (Zhang et al., 2019a)	1618	0.411	0.612	0.491	0.293	12296	0.281	0.422	0.315	0.207
HAKE (Zhang et al., 2020)	1522	0.379	0.551	0.432	0.283	13211	0.245	0.370	0.283	0.175
DualE (Cao et al., 2021)	1363	0.360	0.592	0.439	0.232	11529	0.292	0.447	0.329	0.214
MultiHopKG (Lin et al., 2018)	-	0.520	0.625	0.530	0.426	-	0.416	0.474	0.345	0.275
RNNLogic (Qu et al., 2021)	7857	0.344	0.514	0.390	0.253	15772	0.335	0.422	0.356	0.290
RPJE (Niu et al., 2020)	1770	0.521	0.576	0.542	0.487	6291	0.360	0.496	0.401	0.288
IterE (Zhang et al., 2019b)	5016	0.190	0.326	0.215	0.120	12998	0.233	0.327	0.246	0.185
EngineKG (Ours)	**1275**	**0.523**	**0.647**	**0.551**	**0.501**	**5243**	**0.454**	**0.506**	**0.407**	**0.293**

Table 2: Link prediction results on four datasets. **Bold** numbers are the best results, and the second best is underlined.

Evaluation Metrics. Take the head entity prediction for an instance, we fill the missing head entity with each entity e in the KG, and score a candidate triple (e, r, t) according to the following energy function together with the path information:

$$E_e(e, r, t, \mathcal{P}) = E_t(e, r, t) + \alpha_1 E_p(e, t, \mathcal{P}) \quad (20)$$

in which we reuse the energy functions in Eq. 5 and Eq. 6, and $\mathcal{P}$ is the path set consisting of all the paths between entities e and t. We rank the scores of the candidate triples in ascending order. Tail entity prediction is similar way.

We employ three frequently-used metrics: (1) Mean rank (MR) and (2) Mean reciprocal rank (MRR) of the triples containing the correct entities. (3) Hits@n is the proportion of the correct triples ranked in the top n. The lower MR, the higher MRR and the higher Hits@n declare the better performance. All the results are "filtered" by wiping out the candidate triples that are already in the KG (Wang et al., 2014).

4.2 Results of Link Prediction

The evaluation results of link prediction are reported in Table 2. **Firstly**, our model EngineKG significantly and consistently outperforms all the state-of-the-art baselines on all the datasets and all the metrics. Compared to best-performing models RotatE and RPJE on MR, EngineKG achieves performance gains of 95.0%/42.4%/3.7%/83.5% compared to RotatE and 100.0%/71.1%/38.8%/20.0% against RPJE on datasets FB15K/FB15K237/DBpedia-242/NELL-995, respectively. Particularly, on FB15K and FB15K237, the difference between the best performing baseline RPJE and our developed model is statistically significant under the paired at the 99% significance level. **Secondly**, our model achieves better performance than the traditional models that utilize triples alone, indicating that EngineKG is capable of taking advantage of extra knowledge including rules and paths as well as concepts, which all benefit to improving the performance of the whole model. **Thirdly**, EngineKG further beats IterE, illustrating the superiority of exploiting both rules and paths for KG inference in a joint logic and data-driven fashion.

4.3 Evaluation on Various Relation Properties

The relations can be classified into four categories: One-to-One (1-1), One-to-Many (1-N), Many-to-

FB15K	Head Entities Prediction				Tail Entities Prediction			
	1-1	1-N	N-1	N-N	1-1	1-N	N-1	N-N
TransE (Bordes et al., 2013)	0.356	0.626	0.172	0.375	0.349	0.146	0.683	0.413
RotatE (Sun et al., 2019)	0.895	0.966	0.602	0.893	0.881	0.613	0.956	0.922
HAKE (Sun et al., 2019)	0.926	0.962	0.174	0.289	0.920	0.682	**0.965**	0.805
DualE (Cao et al., 2021)	0.912	0.967	0.557	0.901	0.915	0.662	0.954	0.926
MultiHopKG (Lin et al., 2018)	-	-	-	-	0.893	0.576	0.921	0.763
RPJE (Niu et al., 2020)	0.942	0.965	0.704	0.916	**0.941**	0.839	0.953	0.933
EngineKG	**0.943**	**0.968**	**0.835**	**0.929**	**0.941**	**0.904**	0.957	**0.949**

FB15K237	Head Entities Prediction				Tail Entities Prediction			
	1-1	1-N	N-1	N-N	1-1	1-N	N-1	N-N
TransE (Bordes et al., 2013)	0.356	0.626	0.172	0.375	0.349	0.146	0.683	0.413
RotatE (Sun et al., 2019)	0.547	0.672	0.186	0.474	0.578	0.140	0.876	0.609
HAKE(Zhang et al., 2020)	0.791	**0.833**	0.098	0.237	0.794	0.372	**0.938**	**0.803**
DualE(Cao et al., 2021)	0.516	0.637	0.153	0.471	0.526	0.135	0.860	0.607
MultiHopKG(Lin et al., 2018)	-	-	-	-	0.417	0.026	0.794	0.457
RPJE(Niu et al., 2020)	0.692	0.476	0.180	0.575	0.669	0.197	0.691	0.685
EngineKG	**0.792**	0.743	**0.629**	**0.651**	**0.807**	**0.399**	0.881	0.757

Table 3: Link prediction results on FB15K and FB15K237 on various relation properties (Hits@10). MultiHopKG could only predict tail entities rather than head entities.

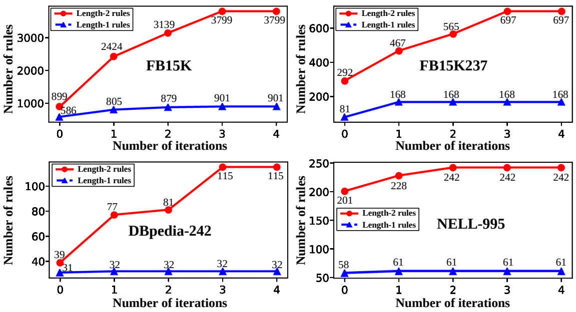

Figure 3: The number of rules over iterations.

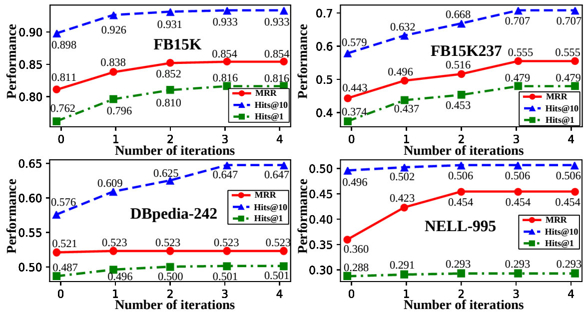

Figure 4: The performance curves of MRR, Hits@10 and Hits@1 over iterations on four datasets.

One (N-1), and Many-to-Many (N-N). We select some well-performing models observed in Table 2 as the baselines in this section. Table 3 exhibits that EngineKG achieves more performance gains on complex relations compared to other baselines. More interestingly, specific to the most challenging tasks (highlighted) namely predicting head entities on N-1 relations and tail entities on 1-N relations, our model consistently and significantly outperforms the outstanding baselines RotatE and RPJE by achieving performance improvements: 38.7%/47.5% on FB15K and 238.2%/185.0% on FB15K237 compared to HAKE while 18.6%/7.75% on FB15K and 249.4%/102.5% on FB15K237 compared to RPJE. These results all demonstrate that the paths and the generated rules enrich the associations between entities and relations, contributing to better performance of KG inference on complex relations.

4.4 Performance Evaluation Over Iterations

We evaluate the performance of our rule learning module on the learning time and the number of rules compared to the excellent rule learning tool AMIE+. For generating high-quality rules, our model takes 6.29s/2.26s/1.55s/10.50s in an iteration on average while AMIE+ takes 79.19s/26.83s/5.35s/105.53s on datasets FB15K/FB15K237/NELL-995/DBpedia, illustrating the higher efficiency of EngineKG. In Figure 3, the amount of rules mined by AMIE+ is shown as that at the initial iteration. Thus, we can discover that the quantity of rules generated in the first iteration and the third iteration is twice and three times the number of rules obtained by AMIE+.

More specifically, Figure 3 exhibits the number of rules and Figure 4 indicates the performance curves on the four datasets over iterations. Notably,

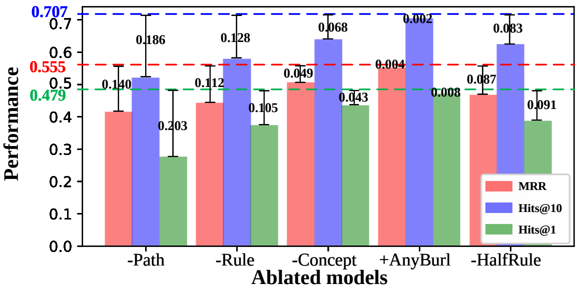

Figure 5: Ablation study on FB15K237. The dash lines indicate the performance of our whole model on MRR (red), Hits@10 (blue) and Hits@1 (green).

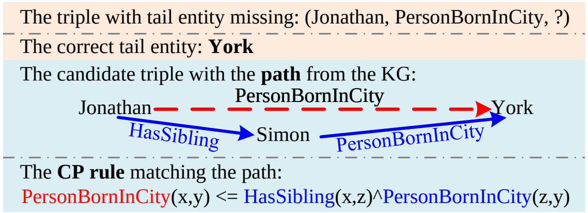

Figure 6: An example of the interpretable tail entity prediction via path and rule on NELL-995.

the number of rules and the performance continue to grow as the iteration goes on and they both converge after three iterations on all the datasets. These results illustrate that: (1) Rule learning and KGE modules in our model indeed complement each other and benefit in not only producing more high-quality rules but also obtaining better inference results. (2) More rules are beneficial to improving the performance of KG inference. (3) The iteration process will gradually converge along with the rule learning.

4.5 Ablation Study

To evaluate each contribution in our whole model EngineKG, we observe the performance on FB15K237 as to the five different ablated settings: (1) Omitting paths (-Path). (2) Omitting rules (-Rule). (3) Omitting concepts (-Concept) by removing concept-based co-occurrences in rule learning. (4) Replacing rule mining tool AMIE+ with AnyBurl (Meilicke et al., 2019) for obtaining the seed rules (+AnyBurl). (5) Employing only half of the seed rules without iteration (-HalfRule). Figure 5 shows that the performance of our whole model is better than that of all the ablated models except for "+AnyBurl", demonstrating that all the components in our designed model are valid and our model is free of any rule mining tool for obtaining the seed rules. Besides, removing paths and rules both have significant impacts on the performance, which suggests the paths and rules in our model play a more vital role in KG inference.

4.6 Case Study

As shown in Figure 6, although the head entity $Jonathan$ and the candidate tail entity $York$ are not linked by any direct relation in the KG, there is an explicit path between them. This path can be represented as the relation $PersonBornInCity$ deduced by a matched CP rule. The standard confidence of the rule shown in Figure 6 is 0.9, which explains that although one's sibling being born in a certain city does not necessarily mean that that person was born in the same city, we can employ this trustworthy rule to explain the reliability of the predicted result obtained by our model. The path and the rule together boost the score of the correct candidate entity $York$ calculated by Eq. 20, and especially provide the interpretability of the result.

5 Conclusion and Future Work

In this paper, we develop a novel closed-loop neural-symbolic learning framework EngineKG for KG inference by jointly rule learning and KGE while exploiting paths and concepts. In the KGE module, both rules and paths are introduced to enhance the semantic associations and interpretability for learning the entity and relation embeddings. In the rule learning module, paths and KG embeddings together with entity concepts are leveraged in the designed rule pruning strategy to generate high-quality rules efficiently and effectively. Extensive experimental results on four datasets illustrate the superiority and effectiveness of our approach compared to some state-of-the-art baselines. In the future, we will investigate combining other semantics such as contextual descriptions of entities, and attempt to apply our model to dynamic KGs.

Acknowledgements

This work was partially supported by Zhejiang Science and Technology Plan Project (No. 2022C01082), the National Natural Science Foundation of China (No. 62072022, 61772054) and the Fundamental Research Funds for the Central Universities.

References

Kurt Bollacker, Georg Gottlob, and Sergio Flesca. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *KDD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Z. Cao, Q. Xu, Z. Yang, X. Cao, and Q. Huang. 2021. Dual quaternion knowledge graph embeddings. In *AAAI*, pages 6894–6902.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2015. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *AAAI*, pages 4816–4823.

Vinh Thinh Ho, Daria Stepanova, Mohamed H. Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. 2018. Rule learning from knowledge graphs guided by embedding models. In *ISWC 2018*, pages 72–90.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *WSDM*, pages 105–113.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP 2011*, pages 529–539.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, S¨oren Auer, and Christian Bizer. 2015. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, pages 3243–3253.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, pages 705–714.

Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime bottom-up rule learning for knowledge graph completion. In *IJCAI*, pages 3137–3143.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*, pages 156–166.

Guanglin Niu, Yongfei Zhang, Bo Li, Peng Cui, Si Liu, Jingyang Li, and Xiaowei Zhang. 2020. Rule-guided compositional representation learning on knowledge graphs. In *AAAI*, pages 2950–2958.

P. G. Omran, K. Wang, and Z. Wang. 2019. An embedding-based approach to rule learning in knowledge graphs. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1–12.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. 2021. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *ICRL*.

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

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, page 2071–2080.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, page 1112–1119.

Wenhan Xiong, Thien Hoang, , and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019a. Quaternion knowledge graph embeddings. In *NeurIPS*, pages 2731–2741.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019b. Iteratively learning embeddings and rules for knowledge graph reasoning. In *WWW*, pages 2366–2377.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *AAAI*, pages 3065–3072.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.