RulE: Knowledge Graph Reasoning with Rule Embedding

Anonymous ACL submission

Abstract

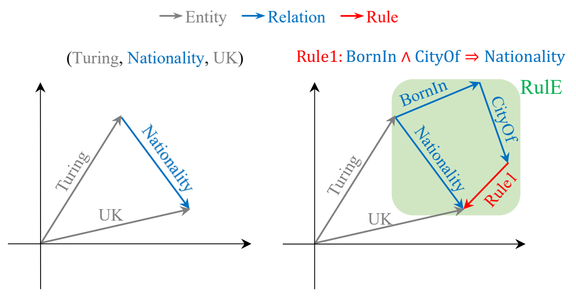
Knowledge graph reasoning is an important problem for knowledge graphs. In this paper, we propose a novel and principled framework called **RulE** (stands for Rule Embedding) to effectively leverage logical rules to enhance KG reasoning. Unlike knowledge graph embedding methods, RulE learns rule embeddings from existing triplets and first-order rules by jointly representing **entities**, **relations** and **logical rules** in a unified embedding space. Based on the learned rule embeddings, a confidence score can be calculated for each rule, reflecting its consistency with the observed triplets. This allows us to perform logical rule inference in a soft way, thus alleviating the brittleness of logic. On the other hand, RulE injects prior logical rule information into the embedding space, enriching and regularizing the entity/relation embeddings. This makes KGE alone perform better too. RulE is conceptually simple and empirically effective. We conduct extensive experiments to verify each component of RulE. Results on multiple benchmarks reveal that our model outperforms the majority of existing embedding-based and rule-based approaches.

1 Introduction

Knowledge graphs (KGs) usually store millions of real-world facts and are used in a variety of applications (Wang et al., 2018; Bordes et al., 2014; Xiong et al., 2017). Examples of knowledge graphs include Freebase (Bollacker et al., 2008), WordNet (Miller, 1995) and YAGO (Suchanek et al., 2007). They represent entities as nodes and relations among entities as edges. Each edge encodes a fact in the form of a triplet (head entity, relation, tail entity). However, KGs are usually highly incomplete, making their downstream tasks more challenging. Knowledge graph reasoning, which predicts missing facts by reasoning on existing facts, has thus become a popular research area in artificial intelligence.

There are two prominent lines of work in this area: *knowledge graph embedding (KGE)* and *rule-based KG reasoning*. Knowledge graph embedding (KGE) methods such as TransE (Bordes et al., 2013), RotatE (Sun et al., 2019) and BoxE (Abboud et al., 2020) embed entities and relations into a latent space and compute the score for each triplet to quantify its plausibility. KGE is efficient and robust to noise. However, it only uses zeroth-order (propositional) logic to encode existing facts (e.g., "Alice is Bob's wife.") without explicitly leveraging first-order (predicate) logic. First-order logic uses the universal quantifier to represent **generally applicable logical rules**. For instance, "$\forall x, y\colon x$ is y's wife $\rightarrow$ y is x's husband". Those rules are **not specific to particular entities** (e.g., Alice and Bob) but are generally applicable to all entities. The other line of work, rule-based KG reasoning, in contrast, explicitly applies logic rules to infer new facts (Galárraga et al., 2013, 2015; Yi et al., 2018; Sadeghian et al., 2019; Qu et al., 2020). Unlike KGE, logical rules can achieve interpretable reasoning and generalize to new entities. However, the brittleness of logical rules greatly harms prediction performance. Consider the logical rule $(x, \text{works in}, y) \rightarrow (x, \text{ lives in}, y)$ as an example. It is mostly correct. Yet, if somebody works in New York but actually lives in New Jersey, the rule can still only infer the wrong fact in an absolute way.

Considering that the aforementioned two lines of work can complement each other, addressing each other's weaknesses with their own merits, it becomes imperative to study how to integrate logical rules with KGE methods in a principled manner. If we view this integration in a broader context, embedding-based reasoning can be seen as a neural method, while rule-based reasoning can be seen as a symbolic method. Neural-symbolic learning has also been a focus of artificial intelligence re-



(a) Traditional KGE (b) Our RulE

Figure 1: (a) Traditional KGE methods embed entities and relations as low-dimensional vectors only using existing triplets by defining operations between entities and relations (e.g., translation); (b) Our RulE associates each rule with an embedding and additionally defines mathematical operations between relations and logical rules (e.g., multi-step translation) to leverage first-order **logical rules**.

search in recent years (Parisotto et al., 2017; Yi et al., 2018; Manhaeve et al., 2018; Xu et al., 2018; Hitzler, 2022).

In the KG domain, such efforts exist too. Some works combine logical rules and KGE by using rules to infer new facts as additional training data for KGE (Guo et al., 2016, 2018) or directly convert some rules into regularization terms for specific KGE models (Ding et al., 2018; Guo et al., 2020). However, they both leverage logical rules merely to enhance KGE training without actually using logical rules to perform reasoning. In this way, they might lose the important information contained in explicit rules, leading to empirically worse performance than state-of-the-art methods.

To address the aforementioned limitations, we propose a simple and principled framework called *RulE*, which aims to learn rule embeddings by jointly representing entities, relations and logical rules in a unified space. As illustrated in Figure 1, given a KG and logical rules, RulE assigns an embedding to each entity, relation and rule, and defines respective mathematical operators between entities and relations (traditional KGE part) as well as between relations and rules (RulE part). It is important to note that we cannot define operators between entities and rules because rules are not specific to particular entities. By jointly optimizing entity, relation and rule embeddings in the same space, RulE allows injecting prior logical rule information to enrich and regularize the embedding space. Our experiments reveal that this joint embedding can boost KGE methods themselves. Additionally, based on the relation and rule embeddings, RulE is able to give a confidence score to each rule, similar to how KGE gives each triplet a confidence score. This confidence score reflects how consistent a rule is with the existing facts, and enables performing logical rule inference in a soft way by softly controlling the contribution of each rule, which alleviates the brittleness of logic.

We evaluate RulE on benchmark link prediction tasks and show superior performance. Experimental results reveal that our model outperforms the majority of existing embedding-based and rule-based methods. We also conduct extensive ablation studies to demonstrate the effectiveness of each component of RulE. All the empirical results verify that RulE is a simple and effective framework for neural-symbolic KG reasoning.

2 Preliminaries

A KG consists of a set of triplets $\mathcal{K} = \{(\mathrm{h}, \mathrm{r}, \mathrm{t}) \mid \mathrm{h}, \mathrm{t} \in \mathcal{E}, \mathrm{r} \in \mathcal{R}\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ denotes the set of entities and $\mathcal{R}$ the set of relations. For a testing triplet $(\mathrm{h}, \mathrm{r}, \mathrm{t})$, we define a query as $q = (\mathrm{h}, \mathrm{r}, ?)$. The knowledge graph reasoning (link prediction) task is to infer the missing entity t based on the existing facts and rules.

2.1 Embedding-based reasoning

Knowledge graph embedding (KGE) represents entities and relations as *embeddings* in a continuous space. It calculates a score for each triplet based on these embeddings via a scoring function. The embeddings are trained so that facts observed in the KG have higher scores than those not observed. The learning goal here is to maximize the scores of positive facts (existing triplets) and minimize those of sampled negative samples.

RotatE (Sun et al., 2019) is a representative KGE method with competitive performance on common benchmark datasets. It maps entities in a complex space and defines relations as element-wise rotations in each two-dimensional complex plane. Each entity and each relation is associated with a complex vector, i.e., $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{C}^k$, where the modulus of each element in $\boldsymbol{r}$ is fixed to 1 (multiplying a complex number with a unitary complex number is equivalent to a 2D rotation). If a triplet $(\mathrm{h}, \mathrm{r}, \mathrm{t})$ holds, it is expected that $\boldsymbol{t} \approx \boldsymbol{h} \circ \boldsymbol{r}$ in the complex space, where $\circ$ denotes the Hadamard (element-wise) product. Formally, the distance function of RotatE is defined as:

$$d(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = \| \boldsymbol{h} \circ \boldsymbol{r} - \boldsymbol{t} \| . \quad (1)$$

2.2 Rule-based reasoning

Logical rules are usually expressed as first-order logic formulae, e.g., $\forall x, y, z\colon (x, \mathrm{r}_1, y) \wedge (y, \mathrm{r}_2, z) \rightarrow (x, \mathrm{r}_3, z)$, or $\mathrm{r}_1(x, y) \wedge \mathrm{r}_2(y, z) \rightarrow \mathrm{r}_3(x, z)$ for brevity. The left-hand side of the implication "$\rightarrow$" is called *rule body* or premise, and the right-hand side is *rule head* or conclusion. Logical rules are often restricted to be closed, which form chains. For a chain rule, successive relations share intermediate entities (e.g., y), and the rule head's and rule body's head/tail entity are the same. Chain rules include common logical rules in KG such as symmetry, inversion, composition, hierarchy, and intersection rules. These rules play an important role in KG reasoning. The length of a rule is the number of atoms (relations) that exist in its rule body. A *grounding* of a rule is obtained by substituting all variables x, y, z with specific entities. If all triplets in the body of a grounding rule exist in the KG, we get a *support* of this rule. Those rules that have nonzero support are called *activated* rules. When inferring a query $(\mathrm{h}, \mathrm{r}, ?)$, rule-based reasoning enumerates relation paths between head h and each candidate tail, and uses activated rules to infer the answer. See Appendix B for illustrative examples.

3 Method

This section introduces our proposed model RulE. RulE is a principled framework to combine KG embedding with logical rules by learning rule embeddings. As illustrated in Figure 2, the training process of RulE consists of three key components. Consider a KG containing triplets and a set of logical rules automatically extracted or predefined by experts. They are: 1) **Joint entity/relation/rule embedding**. We model the relationship between entities and relations as well as the relationship between relations and logical rules to jointly train entity, relation and rule embeddings in a continuous space, as demonstrated in Figure 1. 2) **Soft rule reasoning**. With the rule and relation embeddings, we calculate a confidence score for each rule which is used as the weight of activated rules to output a grounding rule score. 3) Finally, we **integrate** the KGE score calculated from the entity and relation embeddings trained in the first stage and the grounding rule score obtained in the second stage to reason unknown triplets.

3.1 Joint entity/relation/rule embedding

Given a triplet $(\mathrm{h}, \mathrm{r}, \mathrm{t}) \in \mathcal{K}$ and a rule $\mathrm{R} \in \mathcal{L}$, we use $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}, \boldsymbol{R} \in \mathbb{C}^k$ to represent their embeddings, respectively, where k is the dimension of the complex space (following RotatE). Similar to KGE, which encodes the plausibility of each triplet with a scoring function, RulE additionally defines a scoring function for logical rules. Based on the two scoring functions, it jointly learns entity, relation and rule embeddings in the same space by maximizing the plausibility of existing triplets $\mathcal{K}$ (zeroth-order logic) and logical rules $\mathcal{L}$ (first-order logic). The following describes in detail how to model the triplets and logical rules together.

Modeling the relationship between entities and relations To model triplets, we take RotatE (Sun et al., 2019) due to its simplicity and competitive performance. Its loss function with negative sampling is defined as:

$$L_t(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = -\log \sigma(\gamma_t - d(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})) - \sum_{(\boldsymbol{h}', \boldsymbol{r}, \boldsymbol{t}') \in \mathbb{N}} \frac{1}{|\mathbb{N}|} \log \sigma(d(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) - \gamma_t), \tag{2}$$

where γ_t is a fixed triplet margin, $d(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$ is the distance function defined in Equation (1), and $\mathbb{N}$ is the set of negative samples constructed by replacing either the head entity or the tail entity with a random entity using a self-adversarial negative sampling approach. Note that RulE is not restricted to particular KGE models. The RotatE can be replaced with other models, such as TransE (Bordes et al., 2013) and ComplEx (Trouillon et al., 2016), too.

Modeling the relationship between relations and logical rules A universal first-order logical rule is some rule that universally holds for all entities. Therefore, we cannot relate such a rule to specific entities. Instead, it is a higher-level concept related only to the relations it is composed of. Our modeling strategy is as follows. For a logical rule $\mathrm{R}\colon \mathrm{r}_1 \wedge \mathrm{r}_2 \wedge \ldots \wedge \mathrm{r}_l \rightarrow \mathrm{r}_{l+1}$, we expect that $\boldsymbol{r}_{l+1} \approx (\boldsymbol{r}_1 \circ \boldsymbol{r}_2 \circ \ldots \circ \boldsymbol{r}_l) \circ \boldsymbol{R}$. Because the modulus of each element in $\boldsymbol{r}$ is restricted to 1, the multiple rotations in the complex plane are equivalent to the summation of the corresponding angles. We define $g(\boldsymbol{r})$ to return the angle vector of relation $\boldsymbol{r}$ (taking the angle for each element of $\boldsymbol{r}$). Note that the definition of Hadamard product in Equation 1 is equivalent to the term $g(\boldsymbol{r})$ as defined in Equation 3. More interpretations are provided in

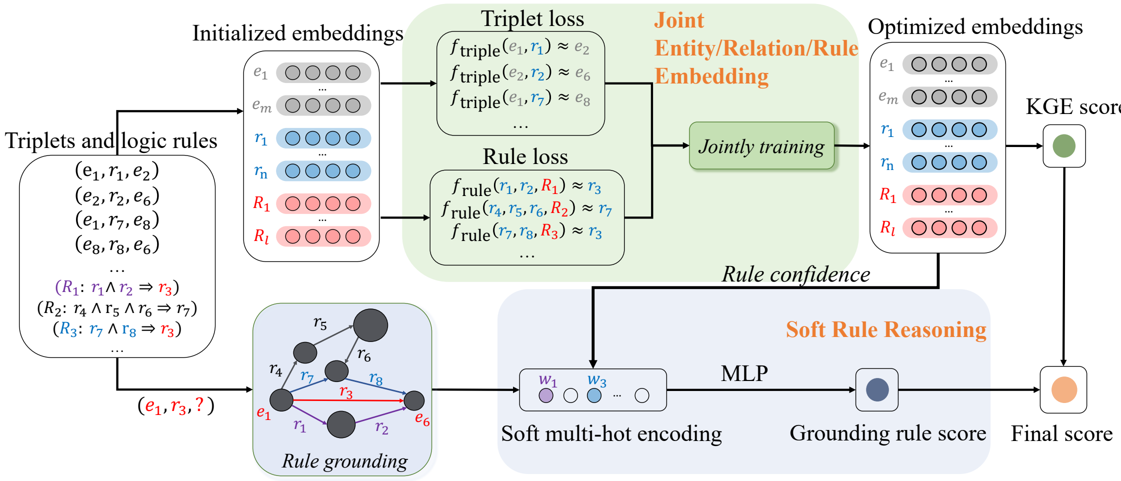


Figure 2: Architecture of RulE. It consists of three components. 1) We first model the relationship between entities and relations as well as the relationship between relations and logical rules to learn **joint entity, relation and rule embedding** in the same continuous space. With the learned rule embeddings ($\boldsymbol{R}$) and relation embeddings ($\boldsymbol{r}$), RulE can output a weight (w) as the confidence score of each rule. 2) In the **soft rule reasoning** stage, we construct a soft multi-hot encoding $\boldsymbol{v}$ based on rule confidences. Specifically, for triplet (e_1, r_3, e_6), only R_1 and R_3 can infer the fact with the grounding paths $e_1 \to r_1 \to r_2 \to e_6$ and $e_1 \to r_7 \to r_8 \to e_6$ (highlighted with purple and blue). Thus, the value of $\boldsymbol{v}_1$ is w_1, $\boldsymbol{v}_3$ is w_3 and others (unactivated rules) are 0. Then the constructed soft multi-hot encoding passes an MLP to output the grounding rule score. 3) Finally, RulE **integrates** the KGE score calculated from the entity and relation embeddings trained in the first stage and the grounding rule score obtained in the second stage to reason unknown triplets.

Appendix H. Then, the distance function is formulated as follows:

$$d_r(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{l+1}, \boldsymbol{R}) = \| \sum_{i=1}^{l} g(\boldsymbol{r}_i) + g(\boldsymbol{R}) - g(\boldsymbol{r}_{l+1}) \| . \tag{3}$$

We also employ negative sampling, the same as when modeling triplets. At this time, it replaces a relation (either in rule body or rule head) with a random relation. The loss function for logical rules is defined as:

$$L_r(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{l+1}, \boldsymbol{R}) = -\log \sigma(\gamma_r - d_r) - \sum_{(\boldsymbol{r}'_1, \ldots, \boldsymbol{r}'_{l+1}, \boldsymbol{R}) \in \mathbb{M}} \frac{1}{|\mathbb{M}|} \log \sigma(d'_r - \gamma_r), \tag{4}$$

where γ_r is a fixed rule margin and $\mathbb{M}$ is the set of negative rule samples.

Note that the above strategy is not the only possible way. For example, when considering the relation order of logical rules (e.g., sister's mother is different from mother's sister), we design a variant of RulE using position-aware sum, which shows slightly improved performance on some datasets. See Appendix G. Nevertheless, we find that Equation (3) is simple and good enough, thus keep it as the default choice.

Joint training Given a KG containing triplets $\mathcal{K}$ and logical rules $\mathcal{L}$, we jointly optimize the two loss functions (2) and (4) to get the final entity, relation and rule embeddings:

$$L = \sum_{(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) \in \mathcal{K}} L_t(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) + \alpha \sum_{(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_l, \boldsymbol{R}) \in \mathcal{L}} L_r(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{l+1}, \boldsymbol{R}), \tag{5}$$

where α is a hyperparameter to balance the two losses. Note that the two losses act as each other's regularization terms. The rule loss (4) cannot be optimized alone, otherwise there always exist $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{l+1}, \boldsymbol{R})$s that can perfectly minimize the loss, leading to meaningless embeddings. However, when jointly optimizing it with the triplet loss, the embeddings will be regularized, and rules more consistent with the triplets tend to have lower losses (by being more easily optimized). On the other hand, the rule loss also provides a regularization to the triplet (KGE) loss by adding additional constraints that relations should satisfy. This additional information enhances the KGE training, leading to entity/relation embeddings more consistent with prior rules.

3.2 Soft rule reasoning

As shown in Figure 2, during soft rule reasoning, we use the joint relation and rule embeddings to compute the confidence score of each rule. Similar to how KGE gives a triplet score, the confidence score of a logical rule $\mathrm{R}_i: \mathrm{r}_{i_1} \wedge \mathrm{r}_{i_2} \wedge ... \wedge \mathrm{r}_{i_l} \rightarrow \mathrm{r}_{i_{l+1}}$ is calculated by:

$$w_i = \gamma_r - d(\boldsymbol{r}_{i_1}, \ldots, \boldsymbol{r}_{i_{l+1}}, \boldsymbol{R}_i), \quad (6)$$

where $d(\boldsymbol{r}_{i_1}, \ldots, \boldsymbol{r}_{i_l+1}, \boldsymbol{R}_i)$ is defined in Equation (3).

To predict a triplet, we perform rule grounding by finding all paths connecting the head and tail that can activate some rule. Often a triplet can have several different rules activated, each with different number of supports (activated paths). An example is shown in Figure 2. The triplet (e_1, r_3, e_6) can be predicted by rule R_1 and R_3 with the grounding paths $e_1 \rightarrow r_1 \rightarrow r_2 \rightarrow e_6$ and $e_1 \rightarrow r_7 \rightarrow r_8 \rightarrow e_6$. In this case, a straightforward way is to use the maximum (i.e., $\max(w_1, w_3)$) or summation (i.e., $w_1 + w_3$) of the confidences of those activated rules as the grounding rule score of the triplet.

However, the above way will lose the *dependency* among different rules. For example, consider the following two rules: parent_of(x, y) $\rightarrow$ mother_of(x, y) and sister_of$(x, z) \wedge$aunt_of$(z, y) \rightarrow$ mother_of(x, y). We know that they individually are both not reliable, because a parent can also be a father, and an aunt's sister can be another aunt. However, when these two rules are activated together, one can almost surely infer the "mother" relation. In practice, those rules extracted automatically may contain a lot of redundancy or noise. Compared to the naive aggregation approach (such as summation or maximum), we choose to use an MLP to model the **complex interdependencies** among rules.

Specifically, let us still consider the example in Figure 2. We construct a soft multi-hot encoding $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{L}|}$ such that $\boldsymbol{v}_i$ is the product of the confidence of R_i and the number of grounding paths activating R_i (# of supports). Formally, $\boldsymbol{v}_i = w_i \times |\mathcal{P}(\mathrm{h}, \mathrm{r}, \mathrm{t}, \mathrm{R}_i)|$ for $i \in \{1, \ldots, \mathcal{L}\}$, where $\mathcal{P}(\mathrm{h}, \mathrm{r}, \mathrm{t}, \mathrm{R}_i)$ is the set of supports of the rule R_i applying to the current triplet $(\mathrm{h}, \mathrm{r}, \mathrm{t})$. For the candidate e_6 in Figure 2, the value of $\boldsymbol{v}_1$ is $w_1 \times 1$ (grounding path $e_1 \rightarrow r_7 \rightarrow r_8 \rightarrow e_6$ appears one times), $\boldsymbol{v}_3$ is $w_3 \times 1$, and others (unactivated rules) are 0.

With this soft multi-hot encoding $\boldsymbol{v}$, we apply an MLP on $\boldsymbol{v}$ to calculate the grounding rule score:

$$s_g(\mathrm{h}, \mathrm{r}, \mathrm{t}) = \mathrm{MLP}(\boldsymbol{v}). \quad (7)$$

Note that for a query $(\mathrm{h}, \mathrm{r}, ?)$, we will iterate over all candidates t, and the grounding paths for all candidates can be efficiently computed by running BFS. The complexity analysis is presented in Appendix F. Once we have the grounding rule score for all candidate answers, we further use a softmax function to compute the probability of the true answer. Finally, we train the MLP by maximizing the log likelihood of the true answers in the training triplets. Fine-grained implementation details are included in Appendix C.

3.3 Inference

Finally, during inference, we predict any missing fact with a weight-ed sum of the KGE score ($s_t = \gamma_t - d(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$) and the grounding rule score (Equation (7)):

$$s(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = s_t(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) + \beta \cdot s_g(\mathrm{h}, \mathrm{r}, \mathrm{t}'), \quad (8)$$

where β is a hyperparameter balancing the weights of embedding-based and rule-based reasoning.

4 Experiments

In this section, we empirically evaluate RulE on several benchmark KGs and show superior performance to existing embedding-based, rule-based methods and hybrid approaches that combine both. Additionally, we also conduct extensive ablation experiments to verify the effectiveness of each component of RulE. Furthermore, we provide theoretical analysis and case studies in Appendix K to provide further insights and understanding.

4.1 Experiment settings

Datasets We choose six datasets for evaluation: FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018), YAGO3-10 (Mahdisoltani et al., 2014), UMLS, Kinship, and Family (Kok and Domingos, 2007). More details of data split and logical rules used in the experiments are in Appendix I.

Baselines We compare with a comprehensive suite of embedding and rule-based baselines. (1)

[1]Except for YAGO3-10, DistMult, ComplEx and TuckER results are taken from Abboud et al. (2020).

[1]For UMLS and Kinship, [*] means the numbers are taken from Qu et al. (2020); [†] means we rerun the methods with the same evaluation process. For Family, Neural-LP and DRUM results are taken from Sadeghian et al. (2019) and others from our rerun results.

Table 1: Results of reasoning on FB15k-237, WN18RR and YAGO3-10. H@k is in %. [*] means the numbers are taken from the original papers[1]. [†] means we rerun the methods with the same evaluation process. Best results are in **bold** while the seconds are underlined.

	FB15k-237				WN18RR				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE†	0.329	23.0	36.9	52.8	0.222	1.2	39.9	53.0	0.501	40.6	-	67.4
DistMult*	0.241	15.5	26.3	41.9	0.43	39	44	49	0.34	24	38	54
ComplEx*	0.247	15.8	27.5	42.8	0.44	41	46	51	0.36	26	40	55
ConvE*	0.325	23.7	35.6	50.1	0.43	40	44	52	0.44	35	49	62
TuckER*	0.358	**26.6**	39.4	54.4	0.470	44.3	48.2	52.6	0.529	-	-	67.0
RotatE†	0.337	23.9	37.4	53.2	0.476	43.1	49.2	56.2	0.497	40.3	55.2	67.5
PathRank*	0.087	7.4	9.2	11.2	0.189	17.1	20.0	22.5	-	-	-	-
Neural-LP*	0.237	17.3	25.9	36.2	0.435	37.1	43.4	56.6	-	-	-	-
DRUM*	0.343	25.5	37.8	51.6	0.486	42.5	51.3	58.6	-	-	-	-
RNNLogic+ (w/o emb.)*	0.299	21.5	32.8	46.4	0.489	45.3	50.6	56.3	-	-	-	-
RNNLogic+ (w/o emb.)†	0.330	24.3	36.3	50.2	0.502	46.1	52.2	58.5	0.484	41.0	53.8	61.5
NCRL	0.30	20.9	-	47.3	**0.67**	**56.3**	-	**85.0**	0.38	27.4	-	53.6
RNNLogic+ (with emb.)*	0.349	25.8	38.5	53.3	0.513	47.1	53.2	59.7	-	-	-	-
RNNLogic+ (with emb.)†	0.356	26.2	39.3	54.6	0.516	46.9	53.7	60.4	0.499	41.4	55.1	65.8
Naive Combination †	0.350	26.2	38.7	52.8	0.512	46.9	53.1	59.7	0.484	41.0	53.7	61.4
RulE (emb with TransE.)	0.346	25.1	38.5	53.4	0.242	6.7	37.8	52.6	0.510	41.4	57.3	68.2
RulE (emb.)	0.338	24.1	37.6	53.3	0.484	44.3	49.9	56.3	0.530	44.2	58.2	69.0
RulE (rule.)	0.335	24.9	36.9	50.4	0.514	47.3	53.3	59.7	0.481	40.9	53.2	61.0
RulE (emb & rule.)	**0.362**	**26.6**	**40.0**	**55.3**	0.519	47.5	**53.8**	60.5	**0.535**	**44.7**	**58.8**	**69.4**

Embedding-based models: we include TransE (Bordes et al., 2013), DisMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), TuckER (Balažević et al., 2019) and RotatE (Sun et al., 2019). (2) *Rule-based models*: we compare with MLN (Richardson and Domingos, 2006), PathRank (Lao and Cohen, 2010), as well as popular rule learning methods Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), RNNLogic+ (*w/o emb.*) (Qu et al., 2020) and NCRL (Cheng et al., 2023). (3) *Joint KGE and logical rules*: we also compare with baselines that ensemble embedding-based and rule-based method, including RNNLogic+ (*with emb.*) (Qu et al., 2020) and Naive Combination (Meilicke et al., 2021). See more introduction to RNNLogic+ in Appendix D. (4) For our *RulE*, we present results of embedding-based, rule-based and integrated reasoning. The first variant only uses KGE scores obtained from joint entity/relation/rule embedding to reason unknown triplets, denoted by RulE (*emb.*). The second variant only uses the grounding score calculated from soft rule reasoning, denoted by RulE (*rule.*). The last one is the full model combining both, denoted by RulE (*emb & rule.*). Furthermore, to sufficiently verify the effect of rule embedding on different KGE models, we also experiment with a variant of RulE (*emb.*) using TransE (Bordes et al., 2013) as the KGE model, denoted by *emb with TransE.*. We conduct additional experiments on more datasets to compare RulE with the graph-based method NBFNet (Zhu et al., 2021) (see Appendix J.4). Considering the relation order of logical rules, we also design another variant of RulE using position-aware sum (see Appendix G).

Evaluation protocols We follow the setting in RNNLogic (Qu et al., 2020) and evaluate models by Mean Reciprocal Rank (MRR) as well as Hits at N (H@N). For above baselines, we carefully tune the parameters and achieve better results than reported in RNNLogic. To ensure a fair comparison, in the KGE part of RulE, we use the same parameters as those used in TransE and RotatE without further tuning them and rerun RNNLogic+ with the same logical rules as RulE (See Appendix I.3).

Hyperparameter settings By default, we use RotatE (Sun et al., 2019) as our KGE model. We search for parameters according to validation set performance. The ranges of the hyperparameters in the grid search and final adopted values are provided in Appendix I.4.

4.2 Results

The results are shown in Table Tables 1 and 2. We observe that: (1) RulE outperforms both embedding-based and rule-based methods on most datasets, especially on UMLS and Kinship which show significant improvements. This indicates that combining KGE and rule-based methods with rule embedding can take advantage of both and improve the performance of KG reasoning. (2) Compared with loosely composed methods (i.e., RNNLogic+ (*with emb.*) and Naive Combination), RulE (*emb & rule.*) obtains better results on all datasets, demonstrating that it is more beneficial for KG reasoning

Table 2: Results of reasoning on UMLS, Kinship and Family. H@k is in %. [*] means the numbers are taken from Qu et al. (2020); [†] means we rerun the methods with the same evaluation process[2]. Best results are in **bold** while the seconds are underlined.

	UMLS				Kinship				family			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE[†]	0.704	55.4	82.6	92.9	0.300	14.3	35.2	63.7	0.813	67.5	94.6	98.5
DistMult[*]	0.391	25.6	44.5	66.9	0.354	18.9	40.0	75.5	0.680	53.0	78.7	96.6
ComplEx[*]	0.411	27.3	46.8	70.0	0.418	24.2	49.9	81.2	0.930	88.3	97.6	99.1
TuckER[*]	0.732	62.5	81.2	90.9	0.603	46.2	69.8	86.3	-	-	-	-
RotatE[†]	0.802	69.6	89.0	96.3	0.672	53.8	76.4	93.5	0.914	85.3	97.4	99.0
MLN[*]	0.688	58.7	75.5	86.9	0.351	18.9	40.8	70.7	-	-	-	-
PathRank[*]	0.197	14.8	21.4	25.2	0.369	27.2	41.6	67.3	-	-	-	-
Neural-LP[*]	0.483	33.2	56.3	77.5	0.302	16.7	33.9	59.6	0.91	86.0	96.0	99.0
DRUM[*]	0.548	35.8	69.9	85.4	0.334	18.3	37.8	67.5	0.950	91.0	98.0	99.0
RNNLogic+ (w/o emb.)[†]	0.800	70.4	87.8	94.3	0.655	50.4	76.0	94.7	0.974	96.3	98.5	98.6
NCRL	0.78	65.9	-	95.1	0.64	49.0	-	92.9	0.91	85.2	-	**99.3**
RNNLogic+ (with emb.)[†]	0.847	76.7	91.6	96.9	0.714	58.1	81.8	95.4	0.980	97.1	98.9	99.1
Naive Combination[†]	0.856	78.5	91.3	96.3	0.728	60.3	82.1	**95.7**	0.979	97.2	98.5	98.6
RulE (emb with TransE.)	0.748	61.9	85.2	93.3	0.347	20.7	39.8	62.3	0.820	68.9	94.6	98.6
RulE (emb.)	0.807	70.6	89.2	96.3	0.675	53.8	77.1	93.7	0.945	91.0	97.9	99.1
RulE (rule.)	0.827	74.9	88.9	95.5	0.673	52.8	77.5	95.0	0.975	96.7	98.5	98.6
RulE (emb & rule.)	**0.867**	**79.7**	**92.5**	**97.2**	**0.736**	**61.5**	**82.4**	**95.7**	**0.984**	**97.8**	**99.0**	99.1

Table 3: Results of reasoning on FB15k and WN18. H@k is in %. [†] means we rerun the methods with the same evaluation process.

	FB15k		WN18	
	MRR	H@10	MRR	H@10
TransE[†]	0.730	86.4	0.772	92.2
RulE (emb with TransE.)	**0.734**	**86.9**	**0.775**	**95.0**
ComplEx[†]	0.766	88.3	0.898	**95.2**
RulE (emb with ComplEx.)	**0.788**	**89.6**	**0.928**	94.4

to use rule embedding to bridge embedding-based and rule-based approaches than naively combining them. A detailed analysis is as follows.

Embedding logical rules helps KGE We first compare RulE (*emb.*) with RotatE. Note that RulE (*emb.*) and RulE (*emb with TransE.*) only add an additional rule embedding loss to the KGE training and still use KGE scores only for prediction. As presented in Table 1 and 2, RulE (*emb.*) and RulE (*emb with TransE.*) both achieve comparable or higher performance than the corresponding KGE models, especially for RulE (*emb with TransE.*), which obtains 4.4% and 4.7% absolute MRR gain than TransE on UMLS and Kinship. This indicates that by jointly embedding entities/relations/rules into a unified space, RulE can inject logical rule information to enrich and regularize the embedding space and improve the generalization of KGE. This verifies the effectiveness of joint entity/relation/rule embedding.

We also observe that the improvement of RulE (*emb with TransE.*) is more significant than RulE (*emb.*). The reason is probably that RotatE is expressive enough to capture many relational patterns of KG, thus more complex logical rules may be needed. In Table 3, we further use TransE and ComplEx as the KGE model of RulE and test on FB15k and WN18 datasets. They both obtain superior performance to the corresponding KGE models (see Appendix J.1).

Additionally, we find that RulE (*emb with TransE.*) on UMLS and Kinship achieves more improvement than FB15k-237 and WN18RR. The reason is probably that UMLS and Kinship contain more rule-inferrable facts while WN18RR and FB15k-237 consist of more general facts (like the publication year of an album, which is hard to infer via rules). This phenomenon is observed in previous works too (Qu et al., 2020). To verify it, we perform a data analysis in Appendix E.

Soft rule reasoning outperforms hard rule reasoning We compare RulE (*rule.*) with rule mining methods. Note that we rerun RNNLogic+ with the same rules as RulE for fair comparisons. From Table 1 and 2, we can observe that RulE (*rule.*) outperforms existing hard rule reasoning baselines except for WN18RR on NCRL. This demonstrates that soft multi-hot encoding over MLP is more powerful than other ways of performing rule inference.

Comparison with other joint reasoning and rule-enhanced KGE models We also compare with RNNLogic+ (*emb & rule.*) and Naive Combination, which separately trains embedding-based and rule-based methods and then only loosely ensemble them. Although the final inference of RulE (*emb & rule.*) is similar to the above methods (weighted sum over KGE score and grounding rule score), RulE uses rule embedding as a bridge to strengthen KGE and rule reasoning process, by injecting rule information to the KGE embedding space and also extracting rule confidence for soft

Table 4: Ablation study on soft rule reasoning part of RulE. H@k is in %.

	FB15k-237		WN18RR		UMLS		Kinship		Family	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
standard	0.335	50.4	0.514	59.7	0.827	95.5	0.673	95.0	0.975	98.6
sum (w/o MLP)	0.276	42.9	0.390	50.9	0.587	82.0	0.591	90.0	0.877	97.6
max (w/o MLP)	0.256	18.4	0.294	23.4	0.346	23.1	0.373	21.7	0.748	94.9
hard-encoding	0.330	50.2	0.496	45.4	0.791	94.6	0.643	94.0	0.973	96.2

rule reasoning. This demonstrates that the interaction between embedding-based methods and rule-based methods can further enhance each other and the rule embedding serves as the medium. We further study how the hyperparameter β balances both of them. See more details in Appendix J.2.

4.3 Ablation study

This section analyzes whether individual components of the RulE design are useful via ablation experiments. As the usefulness of joint entity/relation/rule embedding has been verified extensively by previous experiments, here we focus on validating the soft rule reasoning part. Specifically, we compare the following RulE versions: (1) *standard*, which is the standard RulE (*rule.*) described in Section 3.2; (2) *hard-encoding*, which only uses hard 1/0 to select activated rules instead of the rule confidence obtained from joint relation/rule embeddings. This is to verify that the confidence scores of logical rules, which are learned through jointly embedding KG and logical rules, help rule-based reasoning; (3) *sum (w/o MLP)* and *max (w/o MLP)*, which replace the MLP layer with sum and max respectively over the weights of all activated rules as the grounding rule score. This is to demonstrate the importance of capturing the complex interdependencies among logical rules.

Ablation Results As presented in Table 4, *standard* achieves better performance than *hard-encoding*, which indicates that using soft multi-hot encoding to perform logical rule inference in a soft way is beneficial to the rule reasoning process. Besides, the performances of *sum (w/o MLP*) and *max (w/o MLP*) versions degrade sharply compared to *standard*, showing that it is important to use an MLP to capture the complex interdependencies among rules.

5 Related work

Embedding-based methods Embedding-based methods aim to learn embeddings for entities and relations and estimate the plausibility of unobserved triplets based on these learned embeddings (Bordes et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Sun et al., 2019; Balažević et al., 2019; Vashishth et al., 2019; Zhang et al., 2020a; Abboud et al., 2020; Ge et al., 2023).

Rule-based methods Learning logical rules for knowledge graph reasoning has also been extensively studied, including Inductive Logic Programming (Quinlan, 1990), Markov Logic Networks (Kok and Domingos, 2005; Beltagy and Mooney, 2014), AMIE (Galárraga et al., 2013), AMIE+ (Galárraga et al., 2015), Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), RNN-Logic (Qu et al., 2020) and other methods (Cheng et al., 2023; Nandi et al., 2023). They almost solely use the learned logical rules for reasoning, which suffer from brittleness and are hardly competitive with embedding-based reasoning in most benchmarks.

Joint KGE and logical rules Some work tries to incorporate logical rules into KGE models. They usually use logical rules to infer new facts as additional training data for KGE (Guo et al., 2016, 2018) or inject rules via regularization terms during training (Wang et al., 2015; Ding et al., 2018). However, they do not really perform reasoning with logical rules.

GNN-based methods Recently, there are some KG reasoning works based on graph neural networks (Schlichtkrull et al., 2018; Teru et al., 2020; Zhang et al., 2020b; Zhu et al., 2021; Li et al., 2023). They exploit neighboring information via message-passing mechanisms. More details of related work and comparison with RNNLogic (Qu et al., 2020) are provided in Appendix A.

6 Conclusion

We propose a simple and principled framework RulE to jointly represent entities, relations and logical rules in a unified embedding space. The incorporation of rule embedding allows injecting rule information to enrich and regularize the embedding space, thus improving the generalization of KGE. Besides, we also demonstrate that with the learned rule embedding, RulE can perform rule inference in a soft way and empirically verify that using an MLP can effectively model the complex interdependencies among rules, thus enhancing rule inference.

7 Limitations

A limitation of RulE is that, similar to prior works which apply logical rules for inference, RulE's soft rule reasoning part needs to enumerate all paths between entity pairs, making it difficult to scale. Another limitation is that currently we only consider chain rules provided as prior knowledge. In the future, we plan to explore more efficient and effective rule reasoning algorithms and consider more complex rules.

References

Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. 2020. Boxe: A box embedding model for knowledge base completion. *Advances in Neural Information Processing Systems*, 33:9649–9661.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Islam Beltagy and Raymond J Mooney. 2014. Efficient markov logic inference for natural language semantics. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Matthias Brocheler, Lilyana Mihalkova, and Lise Getoor. 2012. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*.

Liwei Cai and William Yang Wang. 2017. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*.

Kewei Cheng, Nesreen K Ahmed, and Yizhou Sun. 2023. Neural compositional rule learning for knowledge graph reasoning. *arXiv preprint arXiv:2303.03581*.

William W Cohen, Fan Yang, and Kathryn Rivard Mazaitis. 2017. Tensorlog: Deep learning meets probabilistic dbs. *arXiv preprint arXiv:1707.05390*.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271.

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

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving knowledge graph embedding using simple constraints. *arXiv preprint arXiv:1805.02408*.

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

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422.

Xiou Ge, Yun Cheng Wang, Bin Wang, and C-C Jay Kuo. 2023. Compounding geometric operations for knowledge graph completion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6947–6965.

Shu Guo, Lin Li, Zhen Hui, Lingshuai Meng, Bingnan Ma, Wei Liu, Lihong Wang, Haibin Zhai, and Hong Zhang. 2020. Knowledge graph embedding preserving soft logical regularity. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 425–434.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 192–202.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Pascal Hitzler. 2022. Neuro-symbolic artificial intelligence: The state of the art.

Patrick Hohenecker and Thomas Lukasiewicz. 2020. Ontology reasoning with deep neural networks. *Journal of Artificial Intelligence Research*, 68:503–540.

Stanley Kok and Pedro Domingos. 2005. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448.

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

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Juanhui Li, Harry Shomer, Jiayuan Ding, Yiqi Wang, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. 2023. Are message passing neural networks really helpful for knowledge graph completion? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10696–10711.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31.

Christian Meilicke, Patrick Betz, and Heiner Stuckenschmidt. 2021. Why a naive way to combine symbolic and latent knowledge base completion works surprisingly well. In *3rd Conference on Automated Knowledge Base Construction*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ananjan Nandi, Navdeep Kaur, Parag Singla, et al. 2023. Simple augmentations of logical rules for neuro-symbolic knowledge graph completion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 256–269.

Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. 2017. Neuro-symbolic program synthesis. In *ICLR*.

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

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

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1):107–136.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

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

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

Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR.

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 *International conference on machine learning*, pages 2071–2080. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 417–426.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Twenty-fourth international joint conference on artificial intelligence*.

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

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. 2018. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*.

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

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31.

Yongqi Zhang, Quanming Yao, and Lei Chen. 2020a. Interstellar: searching recurrent architecture for knowledge graph embedding. *Advances in Neural Information Processing Systems*, 33:10030–10040.

Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. 2020b. Efficient probabilistic logic reasoning with graph neural networks. *arXiv preprint arXiv:2001.11850*.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490.

A Related work

Embedding-based methods Embedding-based methods aim to learn embeddings for entities and relations and estimate the plausibility of unobserved triplets based on these learned embeddings (Bordes et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Cai and Wang, 2017; Sun et al., 2019; Balažević et al., 2019; Vashishth et al., 2019; Zhang et al., 2020a; Abboud et al., 2020; Ge et al., 2023). Much prior work in this regard views a relation as some operation or mapping function between entities. Most notably, TransE (Bordes et al., 2013) defines a relation as a translation operation between some head entity and tail entity. It is effective in modelling inverse and composition rules. DistMult (Yang et al., 2014) uses a bilinear mapping function to model symmetric patterns. RotatE (Sun et al., 2019) uses rotation operation in complex space to capture symmetry/antisymmetry, inversion and composition rules. CompoundE (Ge et al., 2023) leverages translation, rotation, and scaling operations to create relation-dependent compound operations on head and/or tail entities. BoxE (Abboud et al., 2020) models relations as boxes and entities as points to capture symmetry/anti-symmetry, inversion, hierarchy and intersection patterns but not composition rules. These approaches learn representations solely based on triplets (zeroth-order logic) contained in the given KG. In contrast, our approach is able to embody more complex first-order logical rules in the embedding space by jointly modeling entities, relations and logical rules in a unified framework.

Rule-based methods Learning logical rules for knowledge graph reasoning has also been extensively studied. As one of the early efforts, Quinlan (1990) uses Inductive Logic Programming (ILP) to derive logical rules (hypothesis) from all the training samples in a KG. Markov Logic Networks (MLNs) (Kok and Domingos, 2005; Brocheler et al., 2012; Beltagy and Mooney, 2014) define the joint distribution of given variables (observed facts) and hidden variables (missing facts) such that missing facts can be inferred in the probabilistic graphical model. AMIE (Galárraga et al., 2013) and AMIE+ (Galárraga et al., 2015) first enumerate possible rules and then learn a scalar weight for each rule to encode its quality. Neural-LP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) mine rules by simultaneously learning logic rules and their weights based on TensorLog (Cohen et al., 2017). RNNLogic (Qu et al., 2020) simultaneously trains a rule generator and reasoning predictor to generate high-quality logical rules. Nandi et al. (2023) propose three augmentations aimed at enhancing the rule set's coverage in RNNLogic-based models. NCRL (Cheng et al., 2023) infers rule head by recursively merging atomic compositions in rule body. Except for RNNLogic, the above methods solely use the learned logical rules for reasoning, which suffer from brittleness and are hardly competitive with embedding-based reasoning in most benchmarks. Although RNNLogic considers the effect of KGE during inference, it pretrains KGE *separately* from logical rule learning without jointly modeling KGE and logical rules in the same space. Most existing works focus on mining rules from observed triplets. In contrast, we focus on the setting where rules are already given (either mined from KG or provided as prior knowledge) and the task is to leverage the rules for better inference. Thus, in principle, our framework can be combined with any rule mining model to improve their rule usage.

Joint KGE and logical rules Some recent work tries to incorporate logical rules into KGE models to improve the generalization performance of KGE reasoning. KALE (Guo et al., 2016) and RUGE (Guo et al., 2018) use logical rules to infer new facts as additional training data for KGE. Several other works inject rules via regularization terms during training, including Wang et al. (2015) and Ding et al. (2018). These methods leverage logical rules only to enhance KGE training and do not really perform reasoning with logical rules. Although Meilicke et al. (2021) combines symbolic and embedding-based methods, it only loosely ensembles the rankings generated by embedding-based and symbolic methods. In contrast, our method jointly learns entity/relation/rule embeddings in a unified space, which is shown to enhance KGE itself. With the learned rule embedding, RulE can also perform logical rule inference in a soft way, improving the rule-based reasoning process. Moreover, the combination of both further advance the performance.

GNN-based methods Recently, there are some KG reasoning works based on graph neural networks (Schlichtkrull et al., 2018; Teru et al., 2020; Zhang et al., 2020b; Zhu et al., 2021; Li et al., 2023). They exploit neighboring information via message-passing mechanisms, which are empirically powerful and can be applied to the inductive setting. However, they usually suffer from high complexity. Furthermore, these methods cannot leverage prior/domain knowledge presented as logical rules, its interpretability is built on path-explanation of the predictions.

B Example of rule-based reasoning

The length of a rule is the number of atoms (relations) that exist in its rule body. One example of a length-2 rule is:

$$\text{born\_in}(x, y) \wedge \text{city\_of}(y, z) \rightarrow \text{nationality}(x, z), \tag{9}$$

of which $\text{born\_in}(\cdot) \wedge \text{city\_of}(\cdot)$ is the rule body and $\text{nationality}(\cdot)$ is the rule head. A *grounding* of a rule is obtained by substituting all variables x, y, z with specific entities. For example, if we replace x, y, z with Bill Gates, Seattle, US respectively, we get a grounding:

$$\text{born\_in}(\text{Bill Gates}, \text{Seattle}) \wedge \text{city\_of}(\text{Seattle}, \text{US}) \rightarrow \text{nationality}(\text{Bill Gates}, \text{US}) \tag{10}$$

If all triplets in the body of a grounding rule exist in the KG, we get a *support* of this rule. Those rules that have nonzero support are called *activated* rules. When inferring a query $(\mathrm{h}, \mathrm{r}, ?)$, rule-based reasoning enumerates relation paths between head h and each candidate tail, and uses activated rules to infer the answer. For example, if we want to infer $\text{nationality}(\text{Bill Gates}, ?)$, given the logical rule (9) as well as the existing triplets $\text{born\_in}(\text{Bill Gates}, \text{Seattle})$ and $\text{city\_of}(\text{Seattle}, \text{US})$, the answer US can be inferred.

C Fine-grained implementation details

This section introduces the fine-grained implementation details. Recall the soft reasoning process: we use the joint relation and rule embeddings to compute a *scalar* as the confidence score of each rule, then construct a soft multi-hot encoding with the confidence, and finally pass the MLP layer to output the grounding rule score. In other words, we obtain the grounding rule score by using a multi-hot encoding vector to activate an MLP. However, in practice, we can use a fine-grained way, i.e., use multiple multi-hot encoding vectors rather than only one.

Specifically, recall that $\boldsymbol{R}, \boldsymbol{r} \in \mathbb{C}^k$ are the embeddings of logical rules and relations, respectively. To prevent confusion, we use $\boldsymbol{v}[i]$ to denote the i-th elements of vector $\boldsymbol{v}$. With the optimized relation and rule embeddings, we can compute the confidence vector of a logical rule $\mathrm{R}_i : \mathrm{r}_{i_1} \wedge \mathrm{r}_{i_2} \wedge ... \wedge \mathrm{r}_{i_l} \rightarrow \mathrm{r}_{i_{l+1}}$ as:

$$\boldsymbol{c}_i = \frac{\gamma_r}{k} - (\sum_{j=1}^{l} \boldsymbol{r}_{i_j} + \boldsymbol{R}_i - \boldsymbol{r}_{i_{l+1}})^p, \tag{11}$$

where p is a hyperparameter, usually the same as the norm defined in Equation (3) , γ_r is the fixed rule margin defined in Equation (4). Note that $\boldsymbol{c}_i$ is a k-dimensional vector, slightly different from the definition in Section 3.2. Each element of $\boldsymbol{c}_i$ represents a way of encoding the confidence of rule R_i. Given the confidence vector $\boldsymbol{c}_i$, we can further construct k multi-hot encoding vectors. Each multi-hot encoding vector activates the MLP to output a grounding score. Further, the mean of all the grounding scores is computed as the grounding rule score s_g of a triplet.

Let us consider the example (e_1, r_3, e_6) in Figure 2. We construct k soft multi-hot encoding vectors $\{\boldsymbol{v}_j \in \mathbb{R}^{|\mathcal{L}|}, j = 1, \ldots, k\}$ such that $\boldsymbol{v}_j[i]$ is the product of of the confidence of R_i and the number of grounding paths activating R_i. Formally, $\boldsymbol{v}_j[i] = \boldsymbol{c}_i[j] \times |\mathcal{P}(\mathrm{h}, \mathrm{r}, \mathrm{t}, \mathrm{R}_i)|$ for $i \in \{1, \ldots, \mathcal{L}\}$, where $\mathcal{P}(\mathrm{h}, \mathrm{r}, \mathrm{t}, \mathrm{R}_i)$ is the set of supports of the rule R_i applying to the current triplet $(\mathrm{h}, \mathrm{r}, \mathrm{t})$. For the candidate e_6 in Figure 2, the value of multi-hot encoding vector $\boldsymbol{v}_j[1]$ is $\boldsymbol{c}_1[j] \times 1$, $\boldsymbol{v}_j[3]$ is $\boldsymbol{c}_3[j] \times 1$, and others are 0 (i.e., $\boldsymbol{v}_j[k] = 0, k = 2, 4, \ldots, \mathcal{L}$).

With these soft multi-hot encoding vectors, we apply an MLP to output the grounding rule score:

$$s_g = \frac{1}{k} \sum_{j=1}^{k} \mathrm{MLP}(\boldsymbol{v}_j). \tag{12}$$

Note that the MLP used by different soft multi-hot encodings is the same. Once we have the grounding rule score for all candidate answers, we further use a softmax function to compute the probability of the true answer. Finally, we optimize the MLP and grounding-stage rule embedding by maximizing the log likelihood of the true answers based on these training triplets.

D Introduction of RNNLogic+

RNNLogic (Qu et al., 2020) aims to learn logical rules from knowledge graphs, which simultaneously trains a rule generator as well as a reasoning predictor. The former is used to generate rules while the latter learns the confidence of generated rules. Because RulE is designed to leverage the rules for better inference, to compare with it, we only focus on the reasoning predictor RNNLogic+, which is a more powerful predictor than RNNLogic. The details are described in this section.

Given a KG containing a set of triplets and logical rules, RNNlogic+ associates each logical rule with a grounding-stage rule embedding $\boldsymbol{R}^{(g)}$ (different from the joint rule embedding in RulE), for a query $(\mathrm{h}, \mathrm{r}, ?)$, it grounds logical rules into the KG, finding different candidate answers. For each candidate answer t', RNNLogic+ aggregates all the rule embeddings of those activated rules, each weighted by the number of paths activating this rule (# supports). Then an MLP is further used to project the aggregated embedding to the grounding rule score $s_r(\mathrm{h}, \mathrm{r}, \mathrm{t}')$:

$$s_r = \mathrm{MLP}\left(\mathrm{AGG}(\{\boldsymbol{R}_i^{(g)}, |\mathcal{P}(\mathrm{h}, \mathrm{R}_i, \mathrm{t}')|\}_{\mathrm{R}_i \in \mathcal{L}})\right) \tag{13}$$

where LN is the layer normalization operation, AGG is the PNA aggregator (Corso et al., 2020), $\mathcal{L}$ is the set of generated high-quality logical rules, and $\mathcal{P}(\mathrm{h}, \mathrm{R}_i, \mathrm{t}')$ is the set of supports of the rule R_i which starts from h and ends at t'. Once RNNLogic+ computes the score of each candidate answer, it can use a softmax function to compute the probability of the true answer. Finally, the predictor can be optimized by maximizing the log likelihood of the true answers based on training triplets. In essence, when replacing the PNA aggregator with sum aggregation, it is equivalent to using hard multi-hot encoding to activate an MLP (i.e., only using hard 1/0 to select activated rules). However, RulE additionally employs the confidence scores of rules as soft multi-hot encoding.

During inference, there are two variants of models:

- RNNLogic+ (*w/o emb.*): This variant only uses the logical rules for knowledge graph reasoning. Specifically, we calculate the score s_r of each candidate answer defined in Equation (13).

- RNNLogic+ (*with emb.*): It uses RotatE (Sun et al., 2019) to *pretrain* knowledge graph embeddings models, which is different from RulE in that RulE jointly models KGE and logical rules in the same space to learn entity, relation and logical rule embeddings. During inference, it linearly combines the grounding rule score and KGE score as the final prediction score, i.e.,

$$s(\mathrm{h}, \mathrm{r}, \mathrm{t}') = s_r(\mathrm{h}, \mathrm{r}, \mathrm{t}') + \alpha * \mathrm{KGE}(\mathrm{h}, \mathrm{r}, \mathrm{t}'), \tag{14}$$

where $\mathrm{KGE}(\mathrm{h}, \mathrm{r}, \mathrm{t}')$ is the KGE score calculated with entity and relation embeddings optimized by RotatE alone, and α is a positive hyperparameter weighting the importance of the knowledge graph embedding score.

E Analysis of rule-inferrable indicator

This section analyzes the rule-inferrable of KGs. Naturally, without considering the directions of edges, any rule can be viewed as a cycle by including both the relation path and the target relation itself. To simplify the analysis, we assume that any cycle can be a logical rule, regardless of concrete relations and the correct semantic information. If a relation appears in a rule, it must be an edge consisting of the cycle; on the other hand, if an edge can be a part of a cycle, it must be a participant relation of the rule. Based on the above hypothesis, we define the proportion of edges existing in cycles to evaluate the rule-inferrable of KGs (i.e., the rule-inferrable indicator).

To verify our hypothesis, we conduct simulation experiments with a Family Tree KG (Hohenecker and Lukasiewicz, 2020), an artificially closed-world dataset generated with logical rules. By randomly selecting $N\%$ of triplets to replace with randomly sampled triplets, we evaluate their rule-inferrable indicators. As shown in Table 5, as the randomness increases, the proportion of edges appearing in cycles decreases and are all lower than in the standard Family Tree. These results indicate that the proportion of edges appearing in the rings can empirically measure the rule-inferrable of KGs.

Next, we analyze the rule-inferrable on all datasets, i.e., FB15k-237, WN18RR, YAGO3-10, UMLS, Kinship and Family. The results are included in Table 6. We observe that: UMLS, Kinship and Family reach 100% of 3-membered cycles while YAGO3-10 and WN18RR have a relatively low proportion, especially WN18RR, which is only about 17%. Therefore, we can empirically conclude that compared to those KGs containing more general facts (FB15k-237, WN18RR and YAGO3-10), UMLS, Kinship and Family are more rule-inferrable datasets. Furthermore, the performance

Table 5: Simulation results of family-tree datasets.

	2-membered cycle	3-membered cycle	$\leq$ 3-membered cycle
standard Family Tree	0.941	0.996	1.000
random5%	0.850	0.958	0.960
random10%	0.766	0.931	0.934
random15%	0.684	0.912	0.915
random20%	0.611	0.898	0.901
random25%	0.542	0.895	0.898
random30%	0.479	0.887	0.891

improvement of the RulE (*emb with TransE.*) is more significant, which is consistent with the observation in our experiments (See Table 2).

F Complexity analysis

This section analyzes the complexity of RulE. We use d to denote hidden dimension and $\mathcal{E}$ is the set of relations (edges).

During training, for the joint entity/relation/rule embedding stage, the amortized time of a single triplet or a logical rule is $O(d)$ due to linear operations. For the soft reasoning part, considering a query $(h, r, ?)$, RulE performs a BFS search from h to find all candidates and compute their grounding rule scores. We group triplets with the same h, r together, where each group contains $|\mathcal{V}|$. For each group, we only need to use an MLP to get predictions, which takes $O(|\mathcal{E}|d^2)$ time. Thus, the amortized time for a single triplet is $O(\frac{|\mathcal{E}|d^2}{|\mathcal{V}|})$.

During inference, we compute the final score with a weighted sum of the KGE score and the grounding rule score. Thus each triplets takes $O(\frac{|\mathcal{E}|d^2}{|\mathcal{V}|} + d)$ time.

The inference time of RulE and RNNLogic+ on different datasets is presented in Table 7. We can see that RulE has similar inference time to RNNLogic+.

G A variant of RulE with position-aware sum

In this section, considering the relation order of rules, we design a variant of RulE using position-aware sum and evaluate the variant based on TransE and RotatE.

It is obvious that 2D rotations and translations are commutative—they cannot model the non-commutative property of composition rules, which is crucial for correctly expressing the relation order of a rule. Take sister_of$(x, y)\wedge$mother_of$(y, z) \rightarrow$ aunt_of(x, z) as an example. If we permute the relations in rule body, e.g., change (sister_of $\wedge$ mother_of) to (mother_of $\wedge$ sister_of), the rule is no longer correct. However, the above model will output the same score since $(\boldsymbol{r}_1 \circ \boldsymbol{r}_2) = (\boldsymbol{r}_2 \circ \boldsymbol{r}_1)$ and $(\boldsymbol{r}_1 + \boldsymbol{r}_2) = (\boldsymbol{r}_2 + \boldsymbol{r}_1)$.

Therefore, to respect the relation order of logical rules, we use position-aware sum to model the relationship between logical rules and relations. Recall that $\boldsymbol{r} \in \mathbb{C}^k$ is the embedding of relation and $g(\boldsymbol{r})$ is to return the angle vector of relation $\boldsymbol{r}$. For each logical rule R: $\mathrm{r}_1 \wedge \mathrm{r}_2 \wedge \ldots \wedge \mathrm{r}_l \rightarrow \mathrm{r}_{l+1}$, we associate it with a rule embedding $\boldsymbol{R} = [\boldsymbol{R}^1, \boldsymbol{R}^2, ..., \boldsymbol{R}^l], \boldsymbol{R} \in \mathbb{C}^{kl}$, where l is the length of the logical rule and $[\cdot, \cdot]$ is concatenation operation. Based on the above definitions, we can formulate the distance function as:

$$d(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{l+1}, \boldsymbol{R}) = \| \sum_{j=1}^{l} \Big(g(\boldsymbol{r}_k) \cdot g(\boldsymbol{R}^k) \Big) - g(\boldsymbol{r}_{l+1}) \|, \tag{15}$$

where $\cdot$ is an element-wise product. Then we use Equation (4) to further define the loss function of logical rules.

Experimental results with TransE and RotatE are displayed in Table 8. RulE (*emb_o.*) is the new version that uses position-aware sum. From the results, we can see that RulE (*emb_o.*) almost obtains superior performance to the corresponding KGE models, again empirically demonstrating that jointly representing entity, relation and rule embeddings can improve the generalization of KGE. Moreover, the performance of RulE (*emb_o.*) is comparable with RulE (*emb.*) in FB15k-237 and WN18RR. It also increases a lot in UMLS and Kinship, especially Kinship, which outperforms RulE (*emb with TransE.*) with a 2.9% improvement in MRR. The reason is probably that relation order plays an important role in modeling logical rules for rule-inferrable datasets (e.g., UMLS and Kinship).

Table 6: The cycle proportion of edges on all datasets.

	2-membered cycle	3-membered cycle	$\leq$ 3-membered cycle
FB15k-237	0.344	0.856	0.877
WN18RR	0.389	0.177	0.452
YAGO3-10	0.569	0.179	0.698
UMLS	0.676	1.00	1.00
Kinship	0.998	1.00	1.00
Family	0.997	0.954	1.00

Table 7: Inference time (in minutes) of RulE and RNNLogic+ on all datasets.

Inference time	FB15k-237	WN18RR	YAGO3-10	UMLS	Kinship	Family
RulE	3.70	3.10	4.50	0.50	0.75	0.60
RNNLogic+	4.10	3.25	4.88	0.70	0.90	1.13

H Different representations of entity-relation loss and relation-rule loss

The entity-relation loss is defined in terms of the Hadamard product, while the relation-rule loss is defined in terms of $g(r)$. Essentially, the two representations are equivalent. We utilize distinct representations for the sake of convenience and to maintain consistency with the model's implementation. Following the RotatE (Sun et al., 2019) paper, the entity-relation loss (i.e., $t \approx h \circ r$) is defined in terms of the Hadamard product, which is equivalent to rotating the entity-vector with a relation-angle in 2D complex space. For relation-rule loss, if a logical rule R : $r_1 \wedge r_2 \wedge ... \wedge r_l \rightarrow r_{l+1}$ holds, we expect that $r_{l+1} \approx (r_1 \circ r_2 \circ ... \circ r_l) \circ R$. As RotatE restricts the modulus of each r's dimension to be 1, the multiple rotations in the complex plane are equivalent to the summation of the corresponding angles (with the modulus unchanged), making it convenient to use the summation of angles in implementation. Therefore, we do not maintain modulus for r and R (since they are all 1) in our implementation, but only maintain their angular vectors, denoted by $g(r)$ and $g(R)$. To keep consistency with our implementation, it is beneficial to define the function $g(r)$ as the angle vector of relation r and directly formulate the distance function in terms of angle vectors.

I Experiment setup

I.1 Data statistics

The detailed statistics of six datasets for evaluation are provided in Table 9. FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018) and YAGO3-10 are subsets of three large-scale knowledge graphs, FreeBase (Bollacker et al., 2008) and WordNet (Miller, 1995) and YAGO3 (Mahdisoltani et al., 2014). UMLS, Kinship and Family (Kok and Domingos, 2007) are three benchmark datasets for statistical relational learning. For FB15k-237, WN18RR and YAGO3-10, we use the standard split. For Kinship and UMLS, we follow the data split from RNNLogic (Qu et al., 2020) (i.e., split the dataset into train/validation/test with a ratio 3 : 2 : 5) and report the results of some baselines taken from RNNLogic. For Family, we follow the split used by DRUM (Sadeghian et al., 2019). To ensure a fair comparison, we use RNNLogic to mine logical rules and rerun the reasoning predictor of RNNLogic+ with the same logical rules. Here, we consider chain rules, covering common logical rules in KG such as symmetry, composition, hierarchy rules, etc. Because inverse relations are required to apply rules, we preprocess the KGs to add inverse links. More introduction is included in Appendix I.2.

I.2 Data process

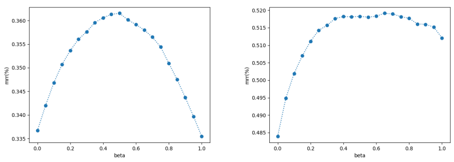
Most rules mined by rule mining systems are not chain rules. They usually need to be transformed into chain rules by inversing some relations. Considering $\mathrm{r}_1(x,y) \wedge \mathrm{r}_2(x,z) \rightarrow \mathrm{r}_3(y,z)$ as an example, with replacing $\mathrm{r}_1(x,y)$ with $\mathrm{r}_1^{-1}(y,x)$, the rule can be converted into chain rule $\mathrm{r}_1(y,x)^{-1} \wedge \mathrm{r}_2(x,z) \rightarrow \mathrm{r}_3(y,z)$. Based on the above, for data processing, we need to add a inverse version triplet $(\mathrm{t}, \mathrm{r}^{-1}, \mathrm{h})$ for each triplet $(\mathrm{h}, \mathrm{r}, \mathrm{t})$, representing the inverse relationship r^{-1} between entity t and entity h.

Table 8: Results of reasoning on FB15k-237, WN18RR, UMLS and Kinship. H@k is in %.

	FB15k-237				**WN18RR**				**UMLS**				**Kinship**			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.329	23.0	36.9	52.8	0.222	1.2	39.9	53.0	0.704	55.4	82.6	92.9	0.300	14.3	35.2	63.7
RulE (emb with TransE.)	0.346	25.1	38.5	53.4	0.242	6.7	37.8	52.6	0.748	61.8	85.1	93.4	0.347	20.7	39.8	62.3
RulE (emb_o with TransE.)	0.336	24.2	37.2	52.2	0.220	3.3	37.2	50.9	0.765	66.9	82.9	92.4	0.376	22.7	42.4	70.0
RotatE	0.337	23.9	37.4	53.2	0.476	43.1	49.2	56.2	0.802	69.6	89.0	96.3	0.672	53.8	76.4	93.5
RulE (emb with RotatE.)	0.337	24.0	37.5	52.9	0.484	44.3	49.9	56.3	0.807	70.6	89.2	96.3	0.675	53.8	77.1	93.7
RulE (emb_o with RotatE.)	0.338	24.1	37.6	53.3	0.484	44.1	50.0	56.7	0.809	71.6	88.3	96.2	0.676	53.8	77.2	93.9

Table 9: Statistics of six datasets.

Dataset	#Entities	#Relations	#Train	#Validation	#Test	#Rules	# length of rules
FB15k-237	14,541	237	272,115	17,535	20,466	131,883	≤ 3
WN18RR	40,943	11	86,835	3,034	3,134	7,386	≤ 5
YAGO3-10	123,182	37	1,079,040	5,000	5,000	7,351	≤ 2
UMLS	135	46	1,959	1,306	3,264	18,400	≤ 3
Kinship	104	25	3,206	2,137	5,343	10,000	≤ 3
Family	3007	12	23,483	2,038	2,835	2,400	≤ 3



(a) FB15k-237 (b) WN18RR

Figure 3: (a) and (b) show the MRR results of RulE with varying β on FB15k-237 and WN18RR.

I.3 Evaluation protocol

During evaluation, for each test triplet (h, r, t), we build two queries $(h, r, ?)$ and $(t, r^{-1}, ?)$ with answer t and h. For each query, we compute the KGE score and grounding rule score (Equation 7) for each candidate entity. As KGE scores and rule scores are scattered over different value ranges, we need to normalize the score before we compute the aggregated score. We map the grounding rule score to $[min, max]$ such that min and max are the minimum and maximum of KGE scores, i.e., map to the range of KGE scores. Then RulE weighted sums over both scores (i.e., $\beta * s_g + (1-\beta) * s_{t_{norm}}$). Once we have the final score for all candidate answers, consider the situation that many entities might be assigned the same score. Following RNNLogic (Qu et al., 2020), we first random shuffles of those entities which receive the same score and then compute the expectation of evaluation metric over them.

I.4 Hyperparameter optimization

We search for parameters according to validation set performance. For above baselines, we carefully tune the parameters and achieve better results than reported in RNNLogic (Qu et al., 2020). To ensure a fair comparison, in the KGE part of RulE, we use the same parameters as those used in TransE and RotatE without further tuning them. When comparing RulE (*rule.*) with RNNLogic+ (*w/o emb.*), we use the same logical rules mined from RNNLogic (Qu et al., 2020). Note that the reported results for TransE and RotatE are indeed based on their best parameter settings, where we carefully tuned their parameters such that our reported results for TransE and RotatE are even higher than those reported in RNNLogic (Qu et al., 2020). However, in the KGE part of RulE, we use the same parameters as those used in TransE and RotatE without further tuning them. So the truth is, we did not adopt TransE/RotatE settings tuned on RulE for TransE/RotatE, but on the contrary, adopt TransE/RotatE settings tuned on themselves for RulE. This should bring disadvantages to RulE, yet we still observe improved performance.

The hyperparameters are tuned by the grid search, The range is set as follows: embedding dimension $k \in \{500, 1000, 2000\}$, batch size of triplets and rules $b \in \{256, 512, 1024\}$, the weight balancing two losses (L_t and L_r) $\alpha \in \{0.5, 1, 2, 3, 4, 5\}$, triplet margin and rule margin $\gamma_t, \gamma_r \in [0 : 30 : 1]$ and the weight balancing embedding-based and rule-based reasoning $\beta \in [0 : 0.05 : 1]$. The optimal parameter con-

Table 10: Hyperparameter configurations of RulE on different datasets.

	Hyperparameter	FB15k-237	WN18RR	YAGO3-10	UMLS	Kinship	Family
Joint embedding	k	1000	500	500	2000	2000	2000
	bt	1024	512	1024	256	256	256
	br	128	256	256	256	256	256
	γ_t	9	6	24	6	6	6
	γ_r	9	2	24	8	5	1
	lr	0.00005	0.00005	0.005	0.0001	0.0001	0.0001
	adv	1.0	0.5	1.0	0.25	0.25	1.0
	λ	0	0.1	0	0	0.1	1.0
	α	3	0.5	10	1	3.0	1.0
Soft rule reasoning	lr	0.005	0.005	0.01	0.0001	0.0005	0.0001
	gb	32	32	16	16	32	32
	β	0.50	0.60	0.10	0.20	0.35	0.35

Table 11: Comparison NBFNet with RulE.

MRR	FB15k-237	WN18RR	UMLS	Kinship	family
NBFNet	0.415	0.551	0.922	0.635	0.990
RulE	0.362	0.519	0.867	0.736	0.984

figurations for different datasets for RulE (*emb & rule.*) can be found in Table 10, including embedding dimension k, batch size of triplets bt, batch size of rules br, fix margin of triplets γ_t, fix margin of triplets γ_r, learning rate lr, self-adversarial sampling temperature adv, regularization coefficient λ, the weight balancing the importance of rules in joint loss function (Equation 5) α, batch size in soft rule reasoning gb and the weight of inference process (Equation 8) β. Note that we use RotatE as the KGE model.

J Experiment details

J.1 Embedding logical rules helps KGE

This section discusses the effectiveness of rule embedding on KGE. As shown in Table 12, the two variants using TransE and ComplEx as KGE models are denoted by RulE (*emb with TransE.*) and RulE (*emb with ComplEx.*), respectively. They both obtain superior performance to the corresponding KGE models.

We also further compare with other rule-enhance KGE models. In the experiment setup, RulE (*emb with TransE.*) uses the same logical rules as KALE (Guo et al., 2016); RulE (*emb with ComplEx.*) uses the same logical rules as ComplEx-NNE-AER (Ding et al., 2018). The comparison shows that RulE (*emb with TransE.*) yields more accurate results than KALE. For RulE (*emb with ComplEx.*), although it does not outperform ComplEx-NNE+AER (probably because it additional injects the regularization terms on entities but RulE does not), compared to RUGE, RulE (*emb with ComplEx.*) also obtains 2% improvement in MRR on FB15k as well as comparable results on WN18.

For a fair comparison, RulE (*emb. TransE*) applies the same logical rules as KALE; RulE (*emb. ComplEx*) uses the same logical rules as ComplEx-NNE-AER.

J.2 Sensitivity analysis of beta

To analyze how the hyperparameter β balances the weights of embedding-based and rule-based reasoning (defined in Equation (8)), we conduct experiments for RulE under varying β. Figure 3(a) and 3(b) show the results on Fb15k-237 and WN18RR.

With the increase of β, the performance of RulE first improves and then drops on both datasets. This is because the information captured by logical rules and knowledge graph embedding is complementary, thus combining embedding-based and rule-based methods can enhance knowledge graph reasoning. Moreover, the trend of β for the performance on the two datasets is different (FB15k-237 tends to drop faster than WN18RR). We think that in WN18RR, information captured by the rule-based method may be more than embedding-based, leading that the rule-based method is more predominant in WN18RR ($\beta = 0.6$).

J.3 More results of ablation study

More results of ablation study are presented in Table 13 and 14.

J.4 Comparison NBFNet with RulE

We follow the results of FB15k-237 and WN18RR reported in NBFNet and conduct additional experiments on UMLS, Kinship and family datasets. The results (MRR) are shown in Table 11:

Table 12: Results of reasoning on FB15k and WN18. H@k is in %. [*] means the numbers are taken from (Guo et al., 2018) and (Ding et al., 2018). [†] means we rerun the methods with the same evaluation process.

	FB15k				WN18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE[†]	0.730	64.6	79.2	86.4	0.772	**70.5**	80.8	92.2
KALE[*]	0.523	38.3	61.6	76.2	0.662	-	85.5	93.0
RulE (*emb with TransE.*)	**0.734**	**65.0**	**79.9**	**86.9**	**0.775**	67.2	**86.2**	**95.0**
ComplEx[†]	0.766	69.7	81.3	88.3	0.898	85.4	92.6	**95.2**
RUGE[*]	0.768	70.3	81.5	86.5	0.943	-	-	94.4
ComplEx-NNE+AER[*]	**0.803**	**76.1**	83.1	87.4	**0.943**	**94.0**	**94.5**	94.8
RulE (*emb with ComplEx.*)	0.788	72.4	**83.3**	**89.6**	0.928	91.9	93.5	94.4

Table 13: Ablation results on FB15k-23 and WN18RR datasets. H@k is in %.

	FB15k-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
standard	0.335	24.9	36.9	50.4	0.514	47.3	53.3	59.7
sum (w/o MLP)	0.276	19.8	30.2	42.9	0.390	32.7	41.9	50.9
max (w/o MLP)	0.256	18.4	27.7	39.7	0.294	23.4	31.5	41.4
hard-encoding	0.330	24.3	36.3	50.2	0.496	45.4	51.5	57.7

Table 14: Ablation results on UMLS, Kinship and Family datasets. H@k is in %.

	UMLS				Kinship				Family			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
standard	0.827	74.9	88.9	95.5	0.673	52.8	77.5	95.0	0.975	96.7	98.5	98.6
sum (w/o MLP)	0.587	46.1	65.7	82.0	0.591	44.3	67.4	90.0	0.877	81.2	92.9	97.6
max (w/o MLP)	0.346	23.1	36.4	58.7	0.372	21.8	40.7	74.7	0.748	63.9	82.7	94.9
hard-encoding	0.791	69.5	86.7	94.6	0.643	49.1	74.5	94.0	0.973	96.2	98.4	98.6

NFBNet has better results than RulE on FB15k-237, WN18RR and UMLS. However, RulE achieves comparable or higher performance than NBFNet on Kinship and family, especially on Kinship, where RulE obtains about 10% absolute MRR gain. This might be explained by that Kinship and family contain more rule-inferrable facts while WN18RR and FB15k-237 consist of more general facts (a more detailed discussion is given in Appendix E). This indicates that our method RulE is more favorable for knowledge graphs where rules play an important role, which is expected as it leverages rules explicitly. Another advantage of RulE is the ability to use prior/domain knowledge, while GNN-based methods cannot leverage prior/domain knowledge presented as logical rules. Moreover, RulE is more interpretable on rule-level than GNN methods, which is still valuable in certain domains. Although NBFNet is also interpretable, RulE's interpretability is on rule level while that of NBFNet is on path level. For example, when the KG system desires high interpretability (such as those in medical applications), each inferred knowledge must be accompanied with which exact rules are responsible for the inference, otherwise the doctors are hard to trust it. In contrast, GNN methods (such as NBFNet) are only interpretable on path-level instead of rule-level. Take "Alice is Bob's mother" as an example, GNN methods might tell us the path "Alice is David's mother" and "David is Bob's brother" is activated during the inference, while our RulE can not only tell us that this path is activated, but also the rule $\forall x, y, z$: mother$(x, y) \wedge$ brother$(y, z) \rightarrow$ mother(x, z) is responsible behind the prediction.

In summary, although NBFNet demonstrates state-of-the-art performance on many KGs, we still believe a hybrid method that can explicitly model and leverage logical rules is desired and worth studying.

K Theoretical analysis and case studies

As mentioned in the main body, the rule embeddings are not only used to regularize the embedding learning. On the other hand, with the rule embeddings, RulE can compute the confidence score for each logic rule, which enhances the original hard rule-based reasoning process through soft rule confidence. Additionally, combining the jointly trained KGE and the confidence-enhanced rule-based reasoning, we arrive at a final neural-symbolic model achieving superior performance on many datasets.

Consider the rule $r_1(x,y) \wedge r_2(y,z) \rightarrow r_3(x,z)$ as an example, where x, y, z represent specific entities. Given three facts, we obtain $y = x \circ r_1$; $z = y \circ r_2$; $z = x \circ r_3$. Combining these equations, we deduce $r_1 \circ r_2 = r_3$. However, those mined rules may not be confidently correct. Thus, we assign a residual embedding as a rule embedding to each logical rule, i.e., $r_1 \circ r_2 \circ R = r_3$. By adding additional constraints that relations should satisfy, rule loss provides a regularization to the triplet (KGE) loss, improving the generalization of KGE. Meanwhile, with the relation and rule embeddings, RulE can further give a confidence score to each rule, which reflects how consistent a rule is with the existing facts and enables performing the rule inference process in a soft way. This provides an explanation of why RulE is better than naive combination methods.

We further provide some case studies illustrating the confidence scores of logical rules learned by RulE on the family dataset.

$$
\begin{aligned}
&(1)\ \text{brother}(x,y) \wedge \text{brother}(z,y) \wedge \text{mother}(t,z) \\
&\qquad\qquad\qquad\qquad \rightarrow \text{son}(x,t) \quad 0.932 \\
&(2)\ \text{brother}(y,x) \wedge \text{brother}(y,z) \wedge \text{father}(t,z) \\
&\qquad\qquad\qquad\qquad \rightarrow \text{son}(x,t) \quad 0.798 \\
&(3)\ \text{mother}(x,y) \wedge \text{brother}(z,y) \\
&\qquad\qquad\qquad\qquad \rightarrow \text{mother}(x,z) \quad 0.834 \\
&(4)\ \text{wife}(x,y) \wedge \text{son}(z,y) \rightarrow \text{mother}(x,z) \quad 0.589
\end{aligned}
$$

Ideally, rules with higher success probability should yield higher confidence scores. For instance, rule (1) has a higher confidence score than rule (2) because the x in rule (2) could also be the daughter of t, while the x in rule (1) must be male because x is y's brother. Our RulE successfully learns them out. Another example is rule (3) and rule (4). They both infer x is z's mother, but rule (4) is less confident because x can also be z's stepmother.