Logical Neural Networks for Knowledge Base Completion with Embeddings & Rules

Prithviraj Sen[*] Amazon
Breno W. S. R. de Carvalho IBM Research
Ibrahim Abdelaziz IBM Research
Pavan Kapanipathi IBM Research
Salim Roukos IBM Research
Alexander Gray IBM Research

Abstract

Knowledge base completion (KBC) has benefitted greatly by learning explainable rules in an human-interpretable dialect such as first-order logic. Rule-based KBC has so far, mainly focussed on learning one of two types of rules: *conjunction-of-disjunctions* and *disjunction-of-conjunctions*. We qualitatively show, via examples, that one of these has an advantage over the other when it comes to achieving high quality KBC. To the best of our knowledge, we are the first to propose learning *both* kinds of rules within a common framework. To this end, we propose to utilize *logical neural networks* (LNN) (Riegel et al., 2020), a powerful neuro-symbolic AI framework that can express both kinds of rules and learn these end-to-end using gradient-based optimization. Our in-depth experiments show that our LNN-based approach to learning rules for KBC leads to roughly 10% relative improvements, if not more, over SotA rule-based KBC methods. Moreover, by showing how to combine our proposed methods with knowledge graph embeddings we further achieve additional 7.5% relative improvement.

1 Introduction

Knowledge bases (KB), e.g. Freebase (Bollacker et al., 2008), are inherently incomplete thus techniques have been proposed to identify facts missing from a KB. Knowledge base completion (KBC) can be broadly classified into *knowledge graph embeddings* (KGE) and *rule-based knowledge base completion* (rule-based KBC). While KGE has achieved highly accurate KBC by learning a low dimensional hyperspace, rule-based KBC offers superior explainability by learning rules in a human-interpretable dialect such as first-order logic (FOL).

In this work, we take a closer look at rule-based KBC. A knowledge base consists of *facts* denoted by triples of the form $\langle h, r, t\rangle$ where tail t and head h denote *entities* while r denotes a *relation*. Figure 1 shows a KB containing two facts $\langle$Al Pacino, bornIn, New York City$\rangle$, $\langle$New York City, partOf, U. S.$\rangle$ denoted by (solid) directed edges. The goal of rule-based KBC is to *infer* missing facts using FOL rules such as the one shown on top of Figure 1 that combines a person P's birth city C and the country in which C is present in to ascertain P's citizenship. By substituting P/Al Pacino, C/New York City, and N/U. S. into the rule we can infer that Al Pacino is a U. S. citizen which is a fact missing from the KB (denoted by a dashed edge in Figure 1).

$$\forall P, N \exists L : \text{citizenOf}(P, N) \leftarrow \text{bornIn}(P, C) \wedge \text{partOf}(C, N)$$

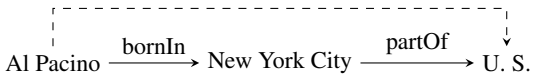

Figure 1: An FOL rule that infers a missing fact.

Most approaches that learn FOL rules for KBC fall into one of two categories: *edge-based* (EB) and *path-based* (PB). We compare their relative strengths via an example. NeuralLP (Yang et al., 2017) is perhaps the oldest example of an EB KBC model. Intuitively, EB approaches learn rules by breaking paths connecting the tail and head of an existing fact into their constituent edges. For instance, to predict missing facts for R_0 in $\mathcal{G}_{\text{train}}$ (Figure 2 (a)), EB learns that in the two paths connecting head and tail of $\langle v_1, R_0, u_1\rangle$, viz. $u_1 \xrightarrow{R_1} w_1 \xrightarrow{R_3} v_1$ and $u_1 \xrightarrow{R_2} w_2 \xrightarrow{R_4} v_1$, the first edge is annotated with either relation R_1 or R_2, and the second edge is annotated with either R_3 or R_4. In FOL syntax, this is expressed as:

$$\forall U, V \exists W : R_0(U, V) \leftarrow \{R_1(U, W) \vee R_2(U, W)\} \wedge \{R_3(W, V) \vee R_4(W, V)\}$$

This kind of rule is called *conjunction-of-disjunctions* since the *conjunction* operator ($\wedge$)

[*] Work done while first author was at IBM Research.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3863 - 3875
December 7-11, 2022

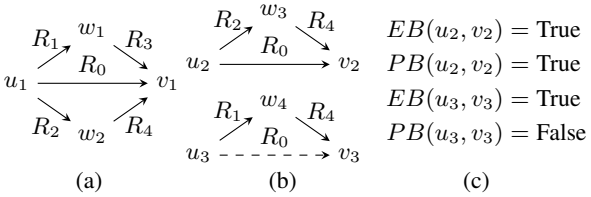

Figure 2: Edge-based (EB) vs. path-based (PB) KBC: (a) $\mathcal{G}_{\text{train}}$, (b) $\mathcal{G}_{\text{test}}$, and (c) Evaluation of rules on $\mathcal{G}_{\text{test}}$.

with *and* semantics is outside and the *disjunction* operators ($\vee$) with *or* semantics are inside. In contrast to EB, recently rule-based KBC has opted for a *path-based* approach (PB) where paths are kept intact. Given the paths connecting u_1 and v_1, PB learns that in a missing R_0 fact the tail and head must either be connected by a path with an R_1 edge followed by an R_3 edge, or an R_2 edge followed by an R_4 edge which is expressed in FOL as:

$$\forall U, V \exists W : R_0(U, V) \leftarrow \{R_1(U, W) \wedge R_3(W, V)\} \vee \{R_2(U, W) \wedge R_4(W, V)\}$$

In contrast to the previous rule, this kind of rule is called *disjunction-of-conjunctions* since the *disjunction* operator ($\vee$) is outside while the *conjunction* operators ($\wedge$) are inside. Both EB and PB's learned rules correctly predict the missing fact $\langle u_2, R_0, v_2 \rangle$ (Figure 2 (b)) which is connected via $u_2 \xrightarrow{R_2} w_3 \xrightarrow{R_4} v_2$ (Figure 2 (c)). However, EB's rule also claims $\langle u_3, R_0, v_3 \rangle$ (Figure 2 (b)), connected by $u_3 \xrightarrow{R_1} w_4 \xrightarrow{R_4} v_3$, is a missing fact while PB's rule predicts *False* in this case correctly indicating that it is a non-fact (indicated by a dashed edge). EB's claim is especially disconcerting given that u_3 and v_3's connecting path contains an R_1 edge followed by an R_4 edge which is a relation sequence that was never observed in $\mathcal{G}_{\text{train}}$!

Previous rule-based KBC approaches have more often than not restricted themselves to learning only one kind of rule, e.g., Yang et al. (2017) learns conjunction-of-disjunctions while MINERVA (Das et al., 2018) and RNNLogic (Qu et al., 2021) learn disjunction-of-conjunctions. In this paper, we propose to utilize *Logical Neural Networks* (LNN) (Riegel et al., 2020), a recently proposed framework for Neuro-symbolic AI (NeSy) that extends Boolean logic to the continuous domain while harboring close ties to FOL semantics thus enabling learning fully explainable rules for KBC via gradient-based optimization. More importantly, LNN's representation power surpasses that of simpler NeSy variants used in EB rule-based KBC approaches such as NeuralLP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) allowing learning of *both* conjunction-of-disjunctions and disjunction-of-conjunctions. We propose an enhanced LNN framework that improves its applicability to KBC and propose LNN extensions of EB and PB rule-based KBC within the same NeSy framework that can then be compared on KBC benchmarks fairly to determine whether the kind of rule has any effect on KBC quality (as implied by our example). Lastly, since LNNs extend logic to the continuous-valued domain, it is relatively simple to combine our proposals with KGE to achieve further improvements in KBC quality.

- We propose rule-based KBC with logical neural networks to learn explainable rules end-to-end.
- We propose new LNN extensions, EB-LNN and PB-LNN that learn conjunction-of-disjunctions and disjunction-of-conjunctions, respectively.
- We also show how to combine our LNN extensions with KGE to further improve KBC quality.
- Comprehensive experiments against 6 baselines on 4 benchmarks show that:
  - EB-LNN leads to $\sim 10\%$ relative improvement over EB-KBC baselines on average.
  - PB-LNN leads to $> 10\%$ relative improvement over PB-KBC baselines on average.
  - PB-LNN leads to $\sim 40\%$ relative improvement over EB-LNN on avg., illustrating the advantage of disjunction-of-conjunctions.
  - By combining with KGE CP-N3 (Lacroix et al., 2018), PB-LNN achieves further relative improvement of 7.5% on avg.

In the next section, we introduce notation and define the KBC task. Section 3 proposes our LNN extensions highlighting its advantages over other NeSy frameworks. In Section 4, we apply LNN to KBC. In Section 5, we present experiments comparing our models against rule-based KBC baselines before concluding with Section 6.

2 Notation and Problem Formulation

2.1 Notation

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{R}, \mathcal{E} \rangle$ denote a knowledge graph (KG) comprising entities $\mathcal{V}$, relations $\mathcal{R}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$. Each edge denotes a *fact* and is given by a triple $\langle h, r, t \rangle$ where $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$. Most KGs are incomplete, in other words, there exist facts $\langle h, r, t \rangle \notin \mathcal{E}$ but hold true in the

real world. In such a case, we would like to predict facts missing from the KG before using it for downstream applications. Two forms of knowledge base completion (KBC) are popular: tail entity prediction $\langle h, r, ?\rangle$, and relation prediction $\langle h, ?, t\rangle$ where '?' indicates what needs to be predicted, with the former being relatively more natural. For instance, the task in Figure 1 $\langle$Al Pacino, citizenOf, ?$\rangle$ can be equivalently stated in natural language as "What is Al Pachino's citizenship?". Following previous work (Yang et al., 2017; Sadeghian et al., 2019; Das et al., 2018; Lin et al., 2018; Qu et al., 2021), we focus on predicting tails.

2.2 Problem Formulation

Given scoring function f, one way to answer query $\langle h, r, ?\rangle$ is to return the top-ranked entity from $\mathcal{V}$:

$$\text{argmax}_{t\in\mathcal{V}} f(h, r, t)$$

In this work, we focus on learning rules in first-order logic to score triples with, which has the additional advantage of of being human-interpretable. First-order logic consists of the three propositional operators $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation), besides $\exists$ (existential) and $\forall$ (for every) operators. In particular, rule-based KBC has a rich history of utilizing *chain* rules (also called open path rules) of the form:

$$\forall X_0, X_m\ \exists X_1, \ldots X_{m-1}:$$
$$r_0(X_0, X_m) \leftarrow r_1(X_0, X_1) \wedge \ldots r_m(X_{m-1}, X_m)$$

where $r_i \in \mathcal{R}, \forall i = 0, \ldots m$, and $X_0, \ldots X_m$ denote logical constants that can take values from $\mathcal{V}$. Essentially, the above rule states that $r_0(X_0, X_m)$ is a missing fact if $r_i(X_{i-1}, X_i)$, or equivalently $\langle X_{i-1}, r_i, X_i\rangle$, holds true $\forall i = 1, \ldots m$, which in turn holds true if the path $X_0 \xrightarrow{r_1} X_1 \ldots \xrightarrow{r_i} \ldots \xrightarrow{r_m} X_m$ exists in the KG. Given such a rule, it is straightforward to devise a scoring function that can be used for tail prediction, i.e., $f(h, r_0, t) =$

$$\begin{cases} 1 & \text{if } h \xrightarrow{r_1} \ldots \xrightarrow{r_i} \ldots \xrightarrow{r_m} t \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

However, since logical operators are not differentiable, learning such a rule is fraught with challenges . While inductive logic programming has for long, attempted to learn rules from data, they have trouble handling noisy nature of real-world data and may not scale to real-world problem sizes. To overcome these issues, we propose to utilize logical neural networks (Riegel et al., 2020), a framework that extends Boolean logic to the continuous domain and is thus differentiable, which in turn, implies that it can utilize gradient-based optimization to learn rules from real-world data.

3 Enhanced Logical Neural Networks

While previous work has utilized neuro-symbolic AI to learn rules for KBC, these have relied on simplistic, parameter-free alternatives to logical operators and harbor tenuous connections to Boolean logic's semantics. LNNs improve upon both aspects. Let us take the specific example of logical conjunction $\wedge$, a core operator in propositional logic. Since $\wedge$ is not differentiable and thus cannot be used to learn rules via gradient-based optimization, NeuralLP (Yang et al., 2017) replaces it with *product* t-norm:

$$\text{Prod}(x, y) = xy,\ \forall x, y \in [0, 1]$$

$\text{Prod}(x, y)$ is depicted in Figure 3 (left) whose output smoothly interpolates from 0 to 1 as the inputs x and y increase. This is distinct from $\wedge$ whose truth table is defined to return `true` (1) *only* when *both* its inputs are `true` (1), and `false` (0) otherwise. In contrast, LNN conjunction is defined as:

$$\begin{aligned} \otimes_{\mathbf{w},\beta.\alpha}(\mathbf{x}) = \quad & \texttt{relu}_1(1, \beta - \mathbf{w}^\top(\mathbf{1} - \mathbf{x})) \\ \text{subject to:} \quad & \beta\mathbf{1} - \alpha\mathbf{w} \leq (1-\alpha)\mathbf{1} \\ & \beta - (1-\alpha)\mathbf{1}^\top\mathbf{w} \geq \alpha \\ & \mathbf{w} \geq \mathbf{0} \end{aligned} \tag{1}$$

where $\mathbf{x} \in [0, 1]^n$ denotes n-ary input vector, $\texttt{relu}_1(\cdot)$ (Krizhevsky, 2010) denotes $\max(0, \min(1, \cdot))$ (clamped version of `relu` (Nair and Hinton, 2010)), $\mathbf{w}$ and β denote parameters, $\mathbf{0}$ ($\mathbf{1}$) denote a vector of 0s (1s), and α denotes a hyperparameter. A major advantage of $\otimes$ over t-norms is the inclusion of parameters $\mathbf{w}, \beta$ that can be learned thus improving fit to data. The other advantage of $\otimes$ is its close connections to $\wedge$'s semantics which is achieved via constraints. Intuitively, the constraints in Equation 1 ensure that $\otimes(\mathbf{x}) \in [0, 1-\alpha]$ if *any* of its inputs are $\leq 1-\alpha$ (guaranteed by the first constraint in Equation 1), and $\otimes(\mathbf{x}) \in [\alpha, 1]$ if *all* of its inputs are $\geq \alpha$ (guaranteed by the second constraint in Equation 1). The first and second properties are direct translations into continuous space of $\wedge$ returning 0 if any of its inputs are 0 and 1 if all inputs are 1. Figure 3 (right)

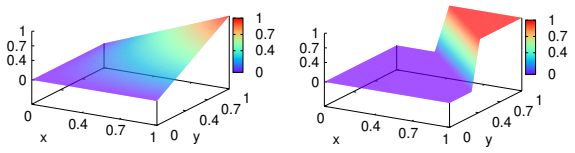

Figure 3: Differentiable conjunctions: product t-norm (left), and $\otimes$ with $\alpha = 0.7$ (right)

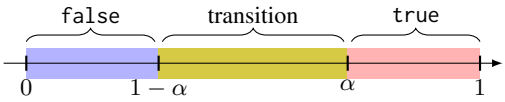

Figure 4: Interpreting $\otimes$ ($\alpha \geq \frac{1}{2}$).

shows a binary $\otimes$ learned from $\wedge$'s truth table with $\alpha = 0.7$ whose output remains $\leq 1 - 0.7 = 0.3$ (in purple) before transitioning to ≥ 0.7 (in red) as inputs increase.

Intuitively, $\otimes$ divides $[0, 1]$ into three regions: $[0, 1-\alpha]$ representing `false`, $(1-\alpha, \alpha)$ denoting a transitionary state, and $[\alpha, 1]$ representing `true` (see Figure 4). From the interpretability perspective, we would like the transitionary state to be as small as possible since it is unclear whether $\otimes$ is `true` or `false` when its output is in this range. One way to affect the size of the transitionary state is by adjusting α which acts as a tunable knob. A higher (lower) α corresponds to a larger (smaller) transitionary state with $\alpha = 1/2$ representing the ideal case where the size reduces to 0 and every output of $\otimes$ can be interpreted either as `true` or `false`. Unfortunately, the following proposition shows that this may not be possible for KBC:

Proposition 3.1. *$\otimes_{\mathbf{w},\beta,\alpha}$ is infeasible if $\alpha < \frac{n}{n+1}$ where n denotes the number of inputs.*

A system of equations is said to be *feasible* if there exists at least one solution for it, otherwise it is *infeasible*. The proof of Proposition 3.1 appears in Appendix A and involves manipulating the constraints in Equation 1. It implies that unless we set α above a certain lower bound, we cannot learn $\mathbf{w}, \beta$ since no such parameter setting exists. On the other hand, even if we set $\alpha \geq n/(n+1)$ to satisfy the lower bound, then the size of the transition state cannot be reduced below $(1 - \frac{n}{n+1}, \frac{n}{n+1})$. This has stark implications for KBC since some of the models we introduce in the next Section entail feeding all relations appearing in the KG to an LNN operator, such as $\otimes$, to learn which relations from $\mathcal{R}$ are useful for predicting missing facts. In this case

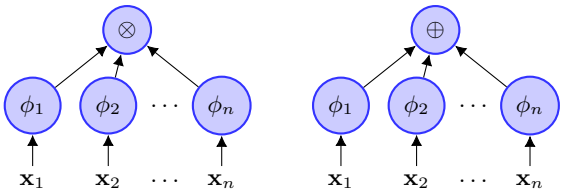

Figure 5: Composing with ϕ: Leads to more interpretable learned operators $\otimes$ (left) and $\oplus$ (right) if $\exists i = 1, \ldots n : |\mathbf{x}_i| > 1$.

however, as $|\mathcal{R}|$ increases[1] so will the arity of the LNN operator which in turn means $n/(n+1) \to 1$ (where $n = |\mathcal{R}|$) thus resulting in a transition state covering the whole range $(0, 1)$ and the learned LNN operator being rendered uninterpretable since it would (almost) never return `true` or `false`.

Our main enhancement to the LNN framework is to enable it to handle high-arity inputs. To this end, we propose to exploit the fact that LNNs being neural networks, allows composition of different operators. Instead of burdening operators such as $\otimes$ to learn from high arity inputs, we use the following operator:

$$\phi_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \text{ subject to } \mathbf{w} \geq \mathbf{0}, \mathbf{w}^\top \mathbf{1} = 1$$

where $\mathbf{x}$ denotes (a vector of) inputs and $\mathbf{w}$ denotes learnable parameters. Since the constraints involved in ϕ are much simpler, it can handle any number of inputs. Instead of feeding a high-arity $\mathbf{x}$ directly to $\otimes$, we can first feed it to ϕ and then feed its output to $\otimes$ which being a single number allows us to set α so that the $\otimes$ can have a much smaller transition state. Figure 5 pictorially depicts the composition where $\oplus$ denotes LNN's disjunction operator which is an extension of classical Boolean logic's *disjunction* operator $\vee$ to continuous space:

$$\oplus_{\mathbf{w},\beta,\alpha}(\mathbf{x}) = 1 - \otimes_{\mathbf{w},\beta,\alpha}(1 - \mathbf{x})$$

where the same constraints from Equation 1 apply since $\oplus$ is defined in terms of $\otimes$.

4 KBC with Logical Neural Networks

In this section, we introduce scoring functions that can perform tail prediction using LNN operators developed so far. All our scoring functions rely on path counts. Let $p = h \xrightarrow{r_1} \ldots \xrightarrow{r_i} \ldots \xrightarrow{r_m} t$ connecting $h, t \in \mathcal{V}$. We refer to the sequence of relations $r_1, \ldots r_m$ as its *relation path*. Furthermore, given an m-length relation path $\mathbf{r} = r_1, \ldots r_m$,

[1] e.g., for the FB15K-237 benchmark $|\mathcal{R}| = 237$.

$\mathcal{P}_{\mathbf{r}}(u, v)$ denotes the *set* of all paths connecting h to t in $\mathcal{G}$ via relation path $\mathbf{r}$.

4.1 Edge-based Logical Neural Networks

Our first scoring function is a translation of the conjunction-of-disjunction approach from Section 1 where we replace the logical operators with LNN operators. Score for triple $\langle h, r, t\rangle$, EB-LNN(h, r, t) is defined as:

$$\sum_{\substack{\mathbf{r}= \\ r_1,\ldots,r_m}} \sum_{\substack{p\in \\ \mathcal{P}_{\mathbf{r}}(h,t)}} \otimes_{\mathbf{w},\beta,\alpha} \left(\phi_{\mathbf{w}_1}(\mathbf{e}_{r_1}), \ldots \phi_{\mathbf{w}_m}(\mathbf{e}_{r_m})\right)$$

where $\mathbf{e}_r \in \{0, 1\}^{|\mathcal{R}|}$ is a one-hot encoding whose r^{th} entry is 1 with 0s everywhere else. The above scoring function exhaustively enumerates all relation paths $\mathbf{r}$ upto length m and assigns $\langle h, r, t\rangle$ a score by counting the number of paths in $\mathcal{P}_{\mathbf{r}}(h, t)$ Essentially, we have replaced the logical conjunction with $\otimes$ and disjunctions with m ϕ operators into the conjunction-of-disjunctions rule. Each ϕ operator has its own set of weights, and combined with the $\otimes$ operator's weights and bias, this means we have a total of $1+m+m|\mathcal{R}|$ parameters that can all be learned via gradient-based optimization. By learning a relation-specific scoring function such as the above, we can learn to predict missing facts for relation $r \in \mathcal{R}$.

4.2 Path-based Logical Neural Networks

Our second scoring function maintains integrity of paths and considers the disjunction-of-conjunctions rule introduced in Section 1. However, here we make a simplification. Note that, unlike EB-LNN, we do not need to learn the conjunctions in the disjunction-of-conjunctions since these are already given to us by the KG $\mathcal{G}$. Each relation path $\mathbf{r}$ is a conjunction and the count $|\mathcal{P}_{\mathbf{r}}(h, t)|$ is essentially all the information we need to compute $\langle h, r, t\rangle$'s score. However, since each relation path is an input to the outer disjunction, using $\oplus$ may compromise its interpretability as there are $|\mathcal{R}|^m$ of these where m denotes the maximum path length. Instead, we express PB-LNN(h, r, t) using a ϕ operator:

$$\sum_{\mathbf{r}=r_1,\ldots,r_m} |\mathcal{P}_{\mathbf{r}}(h, t)| \phi_{\mathbf{w}}(\mathbf{e}_{\mathbf{r}})$$

where $\mathbf{e}_{\mathbf{r}} \in \{0, 1\}^{|\mathcal{R}|^m}$ is a one-hot encoding with 1 in its $\mathbf{r}^{\text{th}}$ position and 0 everywhere else. The total number of parameters is given by $|\mathcal{R}|^m$ which can be learned end-to-end using gradient-based optimization. By learning a relation-specific scoring function such as the above, we can learn to predict missing facts for relation $r \in \mathcal{R}$.

4.3 Combining with Graph Embeddings

One of the drawbacks of EB or PB-LNN is that they treat all paths in $\mathcal{P}_{\mathbf{r}}(h, t)$ equally. This is where utilizing KGE, which embed $\mathcal{V}$ and $\mathcal{R}$ into a low-dimensional hyperspace, may help. KGEs score *more prevalent* relation paths *higher* than less frequent ones. Since LNNs already extend Boolean logic to continuous-valued domain, it is particularly simple for us to incorporate KGE. Our goal is to bias the learning process so that relation paths with larger KGE scores are assigned larger weights. Let $\sigma(p)$ denote the KGE score for path p, $PB - LNN(h, r, t)$ is then given by:

$$\sum_{\mathbf{r}=r_1,\ldots r_m} \phi_{\mathbf{w}}(\mathbf{e}_{\mathbf{r}}) \sum_{p\in\mathcal{P}_{\mathbf{r}}(u,v)} \sigma(p)$$

EB-LNN can also be modified similarly. Note that, there are at least 2 kinds of KGEs available in the literature: 1) that rely on a similarity measure of the triple $\langle h, r, t\rangle$, e.g., CP-N3 (Lacroix et al., 2018), and 2) distance measure used to contrast t's embedding with some function of h and r's embeddings, e.g., RotatE (Sun et al., 2019). For brevity, we only describe the use of similarity based KGE here. Appendix D describes utilizing distance-based KGE.

4.4 Training Logical Neural Networks

While LNN operators are amenable to gradient-based learning, we still need to address how to achieve *constrained* optimization to learn LNN parameters. Fortunately, there exist approaches that can convert any system of linear constraints (including equalities and inequalities) into a sequence of differentiable operations such that we can sample parameters directly from the feasible set (Frerix et al., 2020) which is what we use to train all our LNNs. We also refer the interested reader to Sen et al. (2022) and Riegel et al. (2020) which describe additional LNN training algorithms.

Among various possible training algorithms, based on extensive experimentation, we have found the following scheme to perform reliably. In each iteration, we sample uniformly at random a mini-batch of positive triples B^+ from $\langle h, r, t\rangle \in \mathcal{E}$ and corrupted, negative triples B^- from $\langle h, r, t\rangle \notin \mathcal{E}, \forall h, t \in \mathcal{V}$, such that $|B^+| = |B^-|$ to minimize

	Train	Valid	Test	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{V}\rvert$
Kinship	8544	1068	1074	50	104
Kinship (Qu et al.)	3206	2137	5343	50	104
UMLS	5216	652	661	92	135
UMLS (Qu et al.)	1959	1306	3264	92	135
WN18RR	86835	3034	3134	11	40943
FB15K-237	272155	17535	20466	237	14541

Table 1: Dataset Split Statistics. We also experiment with Qu et al. (2021)'s splits for Kinship and UMLS.

the following margin ranking loss:

$$\sum_{\langle h,r,t\rangle\in B^+} \sum_{\langle h',r,t'\rangle\in B^-} \max\{0, s(h',t')-s(h,t)+\gamma\}$$

where γ denotes the margin hyperparameter and $s()$ denotes one of EB-LNN or PB-LNN.

5 Experimental Results

In this section, we compare the proposed methods with other rule-based KBC approaches. Recall that, in Section 2.2 we pose KBC as a ranking task. Thus, we adopt ranking metrics to compare the quality of the learned rules (Section 5.1). To illustrate interpretability, we also list rule examples in Section 5.2. For brevity, we next describe the salient points of our experimental setup next, while Appendix C provides a full details.

Datasets, Metrics and Baselines: Popular KBC benchmarks include Kinship & UMLS (Kok and Domingos, 2007), WN18RR (Dettmers et al., 2018), and FB15K-237 (Toutanova and Chen, 2015). In addition to their standard splits, Qu et al. (2021) define their own splits for Kinship and UMLS which we also report results on. Table 1 provides dataset statistics. To evaluate the efficacy of learned rules, we compute *filtered* ranks (Bordes et al., 2013) and report mean reciprocal rank (MRR) and Hits@K (H@K) for $K = \{3, 10\}$. To address the case where distinct tails are ranked the same, which may mislead the evaluation (Sun et al., 2020), we average across assignable ranks:

$$g(t) = \frac{1}{m+1} \sum_{r=n}^{n+m} \tilde{g}(r)$$

where tail t is ranked at rank n along with m other tails, $\tilde{g}(r)$ is $1/r$ if metric is MRR and $\delta(r \leq K)$ if metric is Hits@K (δ denotes Dirac delta function).

Baselines: We include a variety of methods:

- EB baselines include NeuralLP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019). Both of these rely on recurrent neural networks (RNN) and variations thereof, to learn conjunction-of-disjunctions.
- PB baselines include RNNLogic (rules only) (Qu et al., 2021), MINERVA (Das et al., 2018) and MultiHopKG (Lin et al., 2018) all of which learn disjunction-of-conjunctions. While RNNLogic relies on RNNs, the latter two utilize reinforcement learning.
- Conditional theorem provers (CTP) (Minervini et al., 2020), a scalable version of neural theorem provers (Rocktäschel and Riedel, 2017).
- KGE combined with rule-based KBC which includes RNNLogic with RotatE (Sun et al., 2019) denoted RNNLogic (w/ embd.).

Please see Appendix B for more details underlying all of these approaches for KBC, including an extended discussion on other closely related works.

Implementation: Following previous work (Yang et al., 2017), we introduce inverse triples, i.e., for each $\langle h, r, t\rangle \in \mathcal{E}$ we add a new triple $\langle t, r^{-1}, h\rangle$ where r^{-1} denotes a new, *inverse* relation. We use Adagrad (Duchi et al., 2011) to train our LNNs and tune hyperparameters on the validation set with step size $\in \{0.1, 1.0\}$, margin $\gamma \in \{0.1, 0.5, 1.0, 2.0\}$, $\alpha \in [5/6, 1)$ and batch size 8. We learn rules up to length 4 for FB15K-237, 5 for WN18RR, and 3 for Kinship and UMLS. We combine our rule-learning approach with pre-trained CP-N3 embeddings (Lacroix et al., 2018) of dimension $4K$ for FB15K-237, $3K$ for WN18RR, and $8K$ for Kinship and UMLS.

5.1 Quantitative Results and Discussion

Tables 2, 3 and 5 list results for all methods. We first compare rule-based KBC methods (RULES), followed by combinations with KGE (W/ EMBD.).

EB-LNN vs. EB baselines: On 3 out of 4 benchmarks, EB-LNN outperforms NeuralLP with respect to MRR (Table 2). A possible reason for this is due to NeuralLP's reliance on simpler operators (e.g., product t-norm) whereas EB-LNN utilizes LNN's parameterized operators that achieve an improved fit and have similar semantics as Boolean logic's (Section 3). Also note that, NeuralLP relies on a recurrent neural network (RNN) to learn its rules whereas EB-LNN uses a handful of LNN operators, one $\otimes$ and few ϕ, resulting in a much simpler architecture. EB-LNN also leads to better MRR in 3 out of 4 datasets compared to DRUM, which replaces the RNN with a bidirectional LSTM

		Kinship			UMLS			WN18RR			FB15K-237		
		MRR	H@10	H@3	MRR	H@10	H@3	MRR	H@10	H@3	MRR	H@10	H@3
EMBD.	CP-N3	88.9	98.7	94.8	93.3	99.7	98.9	47.0*	54.0*	–	36.0*	54.0*	–
	Complex-N3	89.2	98.6	94.7	95.9	99.7	98.9	48.0*	57.0*	–	37.0*	56.0*	–
	RotatE	75.5	97.3	86.5	86.1	99.5	97.0	47.6*	57.1*	49.2*	33.8*	53.3*	37.5*
	Complex	83.7	98.4	91.6	94.5	99.7	98.0	44.0	49.6	44.9	31.8	49.1	34.7
RULES	NeuralLP	48.8	89.1	63.0	55.3	93.0	75.4	33.7	50.2	43.3	25.8	48.5	31.4
	DRUM	40.0	86.1	48.2	61.8	97.9	91.2	34.8	52.1	44.6	25.8	**49.1**	31.5
	CTP	70.3	93.9	79.7	80.1	97.0	91.0	–	–	–	–	–	–
	RNNLogic	64.5	91.1	72.9	71.0	91.1	82.1	45.5*	53.1*	47.5*	28.8*	44.5*	31.5*
	EB-LNN (**Ours**)	39.1	68.7	43.6	78.7	95.1	88.9	36.6	49.2	39.2	28.0	43.5	30.4
	PB-LNN (**Ours**)	**81.9**	**98.4**	**89.3**	**90.0**	**99.4**	**98.3**	**47.3**	**55.5**	**49.7**	**30.7**	47.0	**34.2**
W/ EMBD.	RNNLogic w/ RotatE	–	–	–	–	–	–	48.3*	55.8*	49.7*	34.4*	**53.0***	38.0*
	PB-LNN (**Ours**) w/ CP-N3	**91.1**	**99.2**	**96.5**	**94.5**	**100**	**99.2**	**48.5**	**56.1**	**50.2**	**35.1**	**53.0**	**39.1**

Table 2: Results of LNN-based methods in comparison to other state-of-the-art approaches on standard splits. Bold font denotes best within each category of KBC approaches. * denotes results copied from original papers. CTP did not scale to larger datasets. Lacroix et al. (2018) do not report H@3 for ComplEx-N3 and CP-N3 on WN18RR and FB15K-237. Qu et al. (2021) do not report RNNLogic's results on Kinship and UMLS with RotatE.

	WN18RR			FB15K-237		
	MRR	H@10	H@3	MRR	H@10	H@3
MINERVA*	44.8	51.3	45.6	29.3	45.6	32.9
MultiHopKG*	47.2	54.2	–	40.7	56.4	–
RNNLogic*	47.8	55.3	50.3	37.7	54.9	41.2
RNNLogic w/ RotatE*	50.6	**59.2**	52.3	44.3	**64.0**	48.9
PB-LNN (**Ours**)	51.6	58.4	53.5	40.0	57.4	44.4
PB-LNN w/ CP-N3 (**Ours**)	**51.9**	59.0	**53.9**	**44.8**	63.6	**49.5**

Table 3: Results on direct triples (excluding inverses).

(Hochreiter and Schmidhuber, 1997). In relative terms, EB-LNN's improvement over NeuralLP's (DRUM's) MRR on Kinship, UMLS, WN18RR and FB15K-237 are -19% (-2%), 42% (27%), 8% (5%) and 8% (8%) resulting in 10% (10%) average relative improvement, respectively.

PB-LNN vs. PB baselines: Table 2 compares PB-LNN with RNNLogic (RULES), while Table 3 reports their results only on direct triples including MINERVA's (Das et al., 2018) and MultiHopKG's (Lin et al., 2018) as well. We exclude inverse triples in Table 3 since Das et al. and Lin et al. also exclude these from their evaluation. PB-LNN outperforms all PB baselines in most cases. This is clear indication that the LNN framework, along with our enhancements, is up to the task of learning rules for KBC. These results are even more impressive considering that RNNLogic, MINERVA and MultiHopKG rely on fairly sophisticated architectures such as RNNs and neural reinforcement learning with reward shaping. Across all datasets, PB-LNN leads to 11%, 25% and 4% average relative improvements in MRR over RNNLogic's, MINERVA's and MultiHopKG's, respectively.

PB vs. EB rule-based KBC: This is one of the main questions we aim to answer. We note that, PB-LNN outperforms EB-LNN (RULES) in all cases (Table 2). A possible reason is the confusion EB methods suffer from as illustrated in the example from Section 1. Having said that, the margin of difference separating PB-LNN and EB-LNN varies with the dataset. For instance, on FB15K-237 EB-LNN's MRR comes close to PB-LNN's. PB-LNN leads to 40% average relative improvement in MRR over EB-LNN's across datasets, and is the best performing among all rule-based KBC methods. Only on FB15K-237, PB-LNN is outperformed by NeuralLP and DRUM with respect to Hits@10. However note that, since Hits@10 is a "lenient" metric (one only needs to rank the correct tail entity within the top 10) this is unlikely to matter in a practical setting where PB-LNN's superior MRR and Hits@3 should hold it in good stead.

KGE with Rule-based KBC: We combine PB-LNN (our best LNN-based approach) with CP-N3 embeddings (Lacroix et al., 2018) (Table 2 EMBD. lists results of various KGEs for reference). Across datasets, PB-LNN w/ CP-N3 leads to average relative improvements of 7% over PB-LNN's (RULES) MRR. In comparison to RNNLogic w/ RotatE, PB-LNN w/ CP-N3 shows small but consistent improvements in both Table 2 (W/ EMBD.) and in

FB15K-237	1) person_language$(P, L) \leftarrow$ nationality$(P, N) \wedge$ spoken_in(L, N) 2) film_language$(F, L) \leftarrow$ film_country$(F, C) \wedge$ spoken_in(L, C) 3) tv_program_language$(P, L) \leftarrow$ country_of_tv_program$(P, N) \wedge$ official_language(N, L) 4) burial_place$(P, L) \leftarrow$ nationality$(P, N) \wedge$ located_in(L, N) 5) country_of_tv_program$(P, N) \leftarrow$ tv_program_actor$(P, A) \wedge$ born_in$(A, L) \wedge$ located_in(L, N) 6) film_release_region$(F, R) \leftarrow$ film_crew$(F, P) \wedge$ marriage_location$(P, L) \wedge$ located_in(L, R) 7) marriage_location$(P, L) \leftarrow$ celebrity_friends$(P, F) \wedge$ marriage_location$(F, L') \wedge$ location_adjoins(L', L)
WN18RR	8) domain_topic$(X, Y) \leftarrow$ hypernym$(X, U) \wedge$ hypernym$(U, V) \wedge$ hypernym$(W, V) \wedge$ hypernym$(Z, W) \wedge$ domain_topic(Z, Y) 9) derivation$(X, Y) \leftarrow$ hypernym$(X, U) \wedge$ hypernym$(V, U) \wedge$ derivation$(W, V) \wedge$ hypernym$(W, Z) \wedge$ hypernym(Y, Z) 10) hypernym$(X, Y) \leftarrow$ member_meronym$(U, X) \wedge$ hypernym$(U, V) \wedge$ hypernym$(W, V) \wedge$ member_meronym$(W, Z) \wedge$ hypernym(Z, Y)

Table 4: Examples of PB-LNN's learned rules.

Table 3 (results on direct triples only). Qu et al. (2021) do not evaluate RNNLogic on Kinship and UMLS's standard splits, so we cannot report these in Table 2. Table 5 reports results on Qu et al.'s Kinship and UMLS splits which contain significantly fewer triples in the training set (Table 1). PB-LNN w/ CP-N3 shows significant improvements over RNNLogic w/ RotatE illustrating that PB-LNN can learn effective rules even if training data is scarce.

Besides the above experiments, we also compare against conditional theorem provers (CTP) (Minervini et al., 2020) which falls outside the EB vs. PB nomenclature adopted in this work, relying on theorem proving and soft unification instead. On the smaller datasets Kinship and UMLS, PB-LNN outperforms CTP by a wide margin (Table 2) and despite being an improvement over neural theorem provers (Rocktäschel and Riedel, 2017) in terms of scalability, we were unable to scale CTP to the larger WN18RR and FB15K-237 benchmarks.

5.2 Qualitative Analysis and Discusssion

Learned rules can be extracted from EB-LNN by combining the various ϕ operators using Algorithm 1 proposed in Yang et al. (2017). Extracting learned rules from PB-LNN is even simpler and can be achieved by sorting the relation paths in descending order of $w_{\mathbf{r}}$. Table 4 presents some of the top ranked rules learned by PB-LNN.

Rules for FB15K-237: Rule 1 in Table 4 describes a learned rule that infers the language a person speaks by exploiting knowledge of the language spoken in her/his country of nationality. Shown in terms of relation paths: $P(\text{person}) \xrightarrow{\text{nationality}} N(\text{nation}) \xleftarrow{\text{spoken\_in}} L(\text{language})$. Similarly, Rule 2 (3) uses the country of a film (TV program) to ascertain its language. Rules 5, 6 and 7 are longer rules with 3 relations each in their body. Rule 5 infers a TV program's country by first exploiting knowledge of one of its actor's birth place and then determining which country the birth place belongs to. In terms of relation path: $P(\text{program}) \xrightarrow{\text{tv\_program\_actor}} A(\text{actor}) \xrightarrow{\text{born\_in}} L(\text{birth place}) \xrightarrow{\text{located\_in}} N(\text{nation})$. Rule 6 uses a film crew member's marriage location instead to ascertain the same. Rule 7 infers the marriage location of a celebrity by exploiting knowledge of where their friend got married.

Recursive Rules for WN18RR: WN18RR, being a hierarchical KG, is quite sparse which calls for learning longer rules. Notably, rules shown in Table 4 include the same relation on both sides of $\leftarrow$ denoting recursion. Learning recursive rules is considered a relatively difficult task with only a handful of previous works having tackled it, e.g., (Evans and Grefenstette, 2018), so it is interesting to find that these are useful for predicting missing facts from sparse KGs such as WN18RR.

5.3 Running Time

Table 6 reports training times for WN18RR and FB15K-237. PB-LNN is more efficient than NeuralLP and slower than RNNLogic. The bottleneck, shared by all methods in Table 6, is due to exhaustive enumeration of all possible paths.

6 Conclusion

Our goal is to lend structure to the area of KBC. An ideal approach 1) should be interpretable, 2) can accurately identify missing facts, and 3) can be further improved using successful KBC techniques (e.g., KGE). We categorized previous works into *edge-based* or *path-based* rule-based KBC, and illustrating via carefully constructed examples how one of these may lead to rules better suited for KBC.

	Kinship			UMLS		
	MRR	H@10	H@3	MRR	H@10	H@3
CP-N3	60.2	92.2	70.0	76.6	95.0	85.6
ComplEx-N3	60.5*	92.1*	71.0*	79.1*	95.7*	87.3*
RotatE	65.1*	93.2*	75.5*	74.4*	93.9*	82.2*
Complex	54.4	89.0	63.8	74.0	92.5	81.1
RNNLogic	63.9	92.4	73.1	74.5	92.4	83.3
PB-LNN (**Ours**)	**67.4**	**95.0**	**77.0**	**81.5**	**96.6**	**91.8**
RNNLogic w/ RotatE	**72.2***	94.9*	81.4*	84.2*	96.5*	89.1*
PB-LNN w/ CP-N3 (**Ours**)	**72.2**	**96.9**	**81.6**	**87.6**	**98.0**	**94.7**

Table 5: Results on RNNLogic's splits for Kinship and UMLS. * denotes results copied from Qu et al. (2021).

While previous rule-based KBC has utilized neuro-symbolic AI (NeSy), we proposed the use of logical neural networks, a particularly powerful NeSy framework, that is not only differentiable but also harbors strong connections to Boolean logic's semantics resulting in improved interpretability. We further *enhanced* the LNN framework to improve its handling of high-arity inputs that is useful for learning KBC rules. Our exhaustive experiments confirm that 1) EB-LNN and PB-LNN outperform their respective counterparts available in the literature, 2) PB-LNN is the more accurate of the two, 3) Combining with KGE leads to further improvements in KBC quality, and 4) Learned rules can be easily interpreted.

Riegel et al. (2020)'s original proposal of LNNs mostly evaluated the framework on synthetic tasks with limited size datasets. Besides our work in this paper, Jiang et al. (2021) and Chaudhury et al. (2021) have recently applied LNNs to learning rules for short-text entity linking and helping an agent solve text-world games (Adolphs and Hofmann, 2020). Since knowledge bases may benefit diverse downstream applications including but not limited to question-answering (Kapanipathi et al., 2021), semantic search (Berant et al., 2013), and dialogue generation (He et al., 2017), we expect that in future more efforts will be devoted to adapting and utilizing expressive NeSy frameworks, such as LNNs, to solve real-world applications. Other avenues of future work include learning KBC rules more efficiently and combining EB and PB into one unified, approach for improved KBC.

7 Limitations

This is an instance of an NLP application with close ties to Machine Learning, in that, we are less dependent on the language employed in the Knowledge Graph (KG). But we acknowledge that all of the benchmark KGs employed in our experiments are in English. Another concern would be the scalability of rule-based KBC. As indicated in Table 6, while our approach suffices for the KGs used in these experiments, deploying our solution to larger KGs would invariably require more GPU resources and also benefit from further improvements to the scalability of the learning technique.

	PB-LNN	DRUM	NeuralLP	RNNLogic
WN18RR	2554	2406	3258	2280
FB15K-237	33552	55398	36732	24660

Table 6: Per-epoch training time (in seconds).

8 Ethics Statement

All co-authors and contributors to this work commit to EMNLP and ACL's code of ethics. Knowledge graphs are related to social networks, and while techniques proposed here might, in theory, be used to discover missing facts from a social network which could in turn, lead to invasion of privacy, we explicitly acknowledge that it is neither our goal nor our intent to violate anyone's right to privacy.

9 Acknowledgements

This work was completed prior to the first author's joining of Amazon, and while he was employed at IBM Research. We gratefully acknowledge feedback from anonymous reviewers that undoubtedly resulted in improvements to this work.

References

L. Adolphs and T. Hofmann. 2020. Ledeepchef deep reinforcement learning agent for families of text-based games. In *AAAI*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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

Subhajit Chaudhury, Prithviraj Sen, Masaki Ono, Daiki Kimura, Michiaki Tatsubori, and Asim Munawar. 2021. Neuro-symbolic approaches for text-based policy learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3073–3078, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *JAIR*.

Thomas Frerix, Matthias Nießner, and Daniel Cremers. 2020. Homogeneous linear inequality constraints for neural network activations. In *CVPR Workshops*.

Lise Getoor and Ben Taskar. 2007. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776, Vancouver, Canada. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Hang Jiang, Sairam Gurajada, Qiuhao Lu, Sumit Neelam, Lucian Popa, Prithviraj Sen, Yunyao Li, and Alexander Gray. 2021. LNN-EL: A neuro-symbolic approach to short-text entity linking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 775–787, Online. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

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

A. Krizhevsky. 2010. Convolutional deep belief networks on CIFAR-10. Unpublished Manuscript.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, pages 2863–2872. PMLR.

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

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

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

Stephen Muggleton. 1996. Learning from positive data. In *Worshop on ILP*.

Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

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

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. 2020. Logical neural networks. *CoRR*.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *NeurIPS*.

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

Prithviraj Sen, Breno W. S. R. de Carvalho, Ryan Riegel, and Alexander Gray. 2022. Neuro-symbolic inductive logic programming with logical neural networks. In *AAAI*.

Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. 2020. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In *International Conference on Probabilistic Graphical Models*.

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

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. 2020. A re-evaluation of knowledge graph completion methods. In *ACL*.

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.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NeurIPS*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *NeurIPS*.

A Proof for Proposition 3.1

Proof. A simple lower for α is given by observing that $\otimes_{\mathbf{w},\beta,\alpha}$'s output when it is `false`, i.e., $[0, 1-\alpha]$, should lie to the left of $[\alpha, 1]$, which is its output when $\otimes_{\mathbf{w},\beta,\alpha}$ is `true`, on the number line:

$$1-\alpha \leq \alpha \Rightarrow 1 \leq 2\alpha \Rightarrow \frac{1}{2} \leq \alpha$$

Deriving a tighter bound that depends on input $\mathbf{x}$ requires analyzing the constraints in Equation 1 (in the main body). Beginning with the second constraint:

$$\begin{aligned} & \beta - (1-\alpha)\mathbf{1}^\top \mathbf{w} \geq \alpha \\ \text{(rearranging)} \Rightarrow\ & \beta \geq \alpha + (1-\alpha)\mathbf{1}^\top \mathbf{w} \end{aligned} \tag{2}$$

We next consider implications from Equation 1's first constraint (main body):

$$\begin{aligned} & \beta\mathbf{1} - \alpha\mathbf{w} \leq (1-\alpha)\mathbf{1} \\ \text{(multiply by } \mathbf{1}^\top) \Rightarrow\ & n\beta - \alpha\mathbf{1}^\top\mathbf{w} \leq n(1-\alpha) \\ \text{(rearranging)} \Rightarrow\ & \beta \leq 1 - \alpha + \frac{\alpha}{n}\mathbf{1}^\top\mathbf{w} \end{aligned}$$

The last equation combined with Equation 2 implies that:

$$\begin{aligned} & \alpha + (1-\alpha)\mathbf{1}^\top\mathbf{w} \leq \beta \leq 1 - \alpha + \frac{\alpha}{n}\mathbf{1}^\top\mathbf{w} \\ \Rightarrow\ & \alpha + (1-\alpha)\mathbf{1}^\top\mathbf{w} \leq 1 - \alpha + \frac{\alpha}{n}\mathbf{1}^\top\mathbf{w} \\ \Rightarrow\ & 2\alpha - 1 \leq (\alpha + \frac{\alpha}{n} - 1)\mathbf{1}^\top\mathbf{w} \end{aligned}$$

We have already shown that $\alpha \geq \frac{1}{2}$ which implies that the above LHS in the above equation is ≥ 0. Moreover, Equation 1 (main body) also enforces that $\mathbf{w} \geq 0$. These observations imply that the first term on RHS is also ≥ 0:

$$\begin{aligned} & \alpha + \frac{\alpha}{n} - 1 \geq 0 \\ \text{(rearranging)} \Rightarrow\ & \alpha \geq \frac{n}{n+1} \end{aligned}$$

Thus, if $\alpha < \frac{n}{n+1}$ then $\otimes_{\mathbf{w},\beta,\alpha}$ will not have any solution which proves the Proposition. □

B Related Work

Our work lies at the intersection of Knowledge Base Completion and Rule-learning approaches. Within KBC, knowledge graph embeddings is another prevalent technique.

KBC has gained a lot of interest due to their ability to handle the incompleteness of knowledge bases. KGEs map entities and relations to a low-dimensional vector space to infer new facts. These techniques use neighborhood structure of an entity or relation to learn their corresponding embedding. Beginning with TransE (Bordes et al., 2013), KGEs can now use complex vector spaces such as ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2019), and QuatE (Zhang et al., 2019).

While KGE performance has improved significantly over time, rule-based KBC has gained attention due to its inherent ability to be generate interpretable rules (Yang et al., 2017; Sadeghian et al., 2019; Rocktäschel and Riedel, 2017; Qu et al., 2021). The core ideas in rule learning can be categorized into two groups based on their mechanism to select relations for rules. While *Edge-based (EB)*

methods break paths into its constituent edges, e.g. NeuralLP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), *Path-based (PB)* maintains integrity of paths; e.g., MINERVA (Das et al., 2018), RNN-Logic (Qu et al., 2021). Recent trends in both these types of rule learning approaches has shown significant increase in complexity for performance gains over their simpler precursors. Among the first to learn rules for KBC, NeuralLP (Yang et al., 2017) uses a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) as its rule generator. DRUM (Sadeghian et al., 2019) improves upon NeuralLP by learning multiple such rules obtained using a bi-directional LSTM for more steps. MINERVA (Das et al., 2018) on the other hand, proposes to learn the relation sequences appearing in paths connecting source to destination vertices using neural reinforcement learning. RNNLogic (Qu et al., 2021) is the latest to adopt a path-based approach for KBC that consists of two modules, a rule-generator for suggesting high quality paths and a reasoning predictor that uses said paths to predict missing information. RNNLogic employs expectation-maximization for training where the E-step identifies useful paths per data instance (edge in the KG) by sampling from an intractable posterior while the M-step uses the per-instance useful paths to update the overall set of paths. Both DRUM and RNNLogic represent a significant increase in complexity of their respective approaches compared to NeuralLP and MINERVA.

Unlike these approaches, we propose to utilize *Logical Neural Networks* (LNN) (Riegel et al., 2020); a simple yet powerful neuro-symbolic approach (NeSy) which guarantees interpretability by extending Boolean logic to the real-valued domain. Rules learned with LNNs are eminently interpretable and furthermore, we achieve state-of-the-art KBC quality across multiple datasets by combining with KGE while preserving interpretability.

While NeSy enables us to learn rules end-to-end via gradient-based optimization, it is not the only technique available for learning rules. Inductive logic programming (Muggleton, 1996) has for long attempted to learn rules from real-world data but neither scales to real-world problem sizes nor can it handle noise inherent in real-world data. Another area of work is statistical relational learning (Getoor and Taskar, 2007) that include Markov logic networks (Richardson and Domingos, 2006) however, previous work (Qu et al., 2021) has shown that these neither lead to accurate KBC rules nor do they scale to large KBC benchmarks we experiment with in the main body. Interestingly, our proposed addition to the LNN framework, the ϕ operator, is identical to the *gating* node in sum-product networks (Shao et al., 2020) where it functions similar to a disjunction operator. While LNN has a disjunction operator, its interpretability can be compromised when faced with high-arity inputs in applications such as KBC thus its addition enhances the LNN framework.

C Experimental Setup

C.1 Datasets

To evaluate our approach, we experiment on standard KBC benchmarks, viz. Unified Medical Language System (UMLS) (Kok and Domingos, 2007), Kinship (Kok and Domingos, 2007), WN18RR (Dettmers et al., 2018), and FB15K-237 (Toutanova and Chen, 2015).

- Unified Medical Language System (UMLS) (Kok and Domingos, 2007): models relations among biological concepts including drugs, diseases, and treatments.
- Kinship (Kok and Domingos, 2007): comprises relations among members of Central Australian native tribe.
- WN18RR (Dettmers et al., 2018): a dataset derived from WordNet (Miller, 1995) which is a popular linguistic knowledge base comprising on relations such as synonyms, hypernyms, and hyponyms between words.
- FB15K-237 (Toutanova and Chen, 2015): is derived from Freebase (Bollacker et al., 2008), a knowledge graph with encyclopedic information.

Table 1 (in the main body) provides dataset statistics. We use standard train/validation/test splits for all datasets. Note that, RNNLogic defined its own splits for Kinship and UMLS, we report results on these too for a fair comparison.

C.2 Coherent Metrics and Fair KBC Evaluation

In the recent past, there has been intense criticism on the metrics that are used to evaluate KBC techniques. Specifically, for the KBC task where answering $\langle h, r, ?\rangle$ requires the method to assign probability to answers t, techniques, mostly in the rule-learning category, assigned same probability

scores across multiple answers. The evaluation in most cases considered the minimum rank of the correct answer leading to overly optimistic and unfair evaluation (Sun et al., 2020). This has resulted in more accurate definition of KBC metrics proposed by (Sun et al., 2020) which is used by RNNLogic (Qu et al., 2021); the current state-of-the-art KBC approach. Specifically, Sun et al. (2020) proposed to compute the expectation over all candidate entities with the same score as the correct answer.

Given an unseen query $\langle h, r, ?\rangle$ we compute *filtered* ranks (Bordes et al., 2013) for destination vertices after removing the destinations that form edges with h and r present in the train, test and validation sets. Based on Sun et al. (2020)'s suggestions, our definitions of mean reciprocal rank (MRR) and Hits@K (H@K) satisfy two properties: 1) They assign a larger value to destinations ranked lower, and 2) If destinations $t_1, \ldots, t_m$ share the same rank n then each of them is assigned an average of ranks $n, \ldots, n+m-1$:

$$\text{MRR}(t_i) = \frac{1}{m} \sum_{r=n}^{n+m-1} \frac{1}{r}$$

$$\text{H@K}(t_i) = \frac{1}{m} \sum_{r=n}^{n+m-1} \delta(r \leq K)$$

where $\delta()$ denotes the Dirac delta function. We include inverse triples and report averages across the test set.

D Similarity-based and Distance-based Knowledge Graph Embeddings

Let $\sigma(p)$ denote the score of a path p assigned using a KGE. We describe $\sigma(p)$ for both similarity-based (e.g., CP-N3 Lacroix et al. 2018) and distance-based (e.g., RotatE Sun et al. 2019) KGE:

$$\text{similarity-based: } \sigma(p) = \sum_{\langle h,r,t\rangle \in p} \frac{1}{1 + \exp\{-\text{sim}(h,r,t)\}}$$

$$\text{distance-based: } \sigma(p) = \sum_{\langle h,r,t\rangle \in p} \frac{\exp\{2(\delta - d(h,r,t))\} - 1}{\exp\{2(\delta - d(h,r,t))\} + 1}$$

where δ denotes the margin parameter used by the underlying distance-based KGE to convert distances into similarities. For both of these, we break the path into a series of edges, use the underlying KGE to compute similarity $\text{sim}()$ or distance $d()$ for each triple (as the case may be) and aggregate across all triples in the path. Based on extensive experimentation, we recommend `sigmoid` and `tanh` as the non-linear activation for similarity-based and distance-based KGE, respectively. See main body as to how one may incorporate $\sigma(p)$ into proposed LNN scoring functions for KBC (e.g. PB-LNN).