End-to-end Rule Learning from Knowledge Graphs by Ensembling Expert Logical Rules

Anonymous ACL submission

Abstract

Increasing attention has been paid to learning logical rules automatically on knowledge graphs to explain why a missing fact is inferred. Previous approaches focus on directly learning logical rules from numerous instances, overlooking expert rules that are commonly present in practice. Therefore, we examine the problem of incremental rule learning, which aims to learn new rules via ensembling expert logical rules on knowledge graphs. The challenge of rule learning upon expert rules lies in how to preserve the reasoning semantics of expert rules. We present a framework to allow existing end-to-end rule learning approaches to integrate expert logical rules without losing their logical entailments. In more details, we introduce the notion of *complete onehop-transformed set of rules* to integrate rules into neural networks for single-step reasoning. To preserve all logical entailments of expert rules, we develop an algorithm based on reasoning path extraction and optimized by backward reasoning to compute a complete onehop-transformed set of rules. Experimental results on four benchmark datasets demonstrate that the incorporation of expert rules significantly enhances the performance of link prediction on knowledge graphs.

1 Introduction

Knowledge graphs (KGs) have gained extensive application across various real-world domains, including question answering (Mitra and Baral, 2016), recommendation (Lyu et al., 2020) and information retrieval (Xiong et al., 2017a). KGs is composed of a set of *facts* (also called *triples*) of the form (h, r, t), where h is the *head* entity, r the *relation* and t the *tail* entity. Nevertheless, even the largest KGs remain incomplete due to the data collection process is laborious and error-prone. Thus, judging the existence of new facts from existing facts is a crucial task in KG, which is commonly referred to as *knowledge graph completion* (KGC).

Rules Σ :

R_1 : aunt(X, Y) ← sister(X, V_1) ∧ aunt(V_1, Y).

R_2 : aunt(X, Y) ← sister(X, V_1) ∧ son(Y, V_1).

R_3 : sister(X, Y) ← sister(X, V_1) ∧ sister(V_1, Y).

R_4 : sister(X, Y) ← sister(X, V_2) ∧ brother(Y, X).

Ex 1

KG_1: { sister(Mary, Alice), son(Tom, Alice), sister(Diana, Mary) }

aunt(Mary, Tom) ← sister(Mary, Alice) ∧ son(Tom, Alice) *application of* R_2

aunt(Diana, Tom) ← aunt(Mary, Tom) ∧ sister(Diana, Mary) *application of* R_1

Ex 2

KG_2: { sister(Mary, Alice), sister(Alice, Jane), sister(Jane, Diana) }

sister(Mary, Jane) ← sister(Mary, Alice) ∧ sister(Alice, Jane) *application of* R_3

sister(Mary, Diana) ← sister(Mary, Jane) ∧ sister(Jane, Diana) *application of* R_3

Figure 1: Two examples from the dataset Family.

Logical rules play a pivotal role in KGC and can explain why a missing fact is inferred. Early rule learning methods studied in the field of Inductive Logic Programming (ILP) (Quinlan, 1990; Muggleton, 1995; Schüller and Benz, 2018; Wu et al., 2022), where logical rules are learnt from both positive and negative facts by a generate-and-test manner. Recently, neural logic programming approaches (Yang et al., 2017a; Sadeghian et al., 2019; Yang and Song, 2020; Qu et al., 2021; Cheng et al., 2022; Wang et al., 2024), are proposed to learn logical rules. The end-to-end neural learning of logical rules uses tensor operators to learn continuous parameters from triples in a KG, and extract logical rules from the learned parameters to constitute explanations for reasoning. Compared with traditional ILP methods, neural methods are better at learning rules from imperfect data.

However, most existing approaches have focused on directly learning logical rules from existing triples in KG, ignoring the existence of background rules. In practice, there tends to be a set of domain-specific expert rules. Incremental learning is a methodology of machine learning where the trained model can learn new information to extend the existing knowledge, which has seen extensive appli-

cation in fields such as widely applied in natural language processing (Wang et al., 2023) and image classification (Liu et al., 2025). Incremental learning is particularly beneficial in scenarios where the learning system must dynamically evolve to incorporate novel information or adapt to changing requirements, while maintaining the reasoning capabilities of the original system. Therefore, we examine the problem of incremental rule learning, which aims to learn rules via ensembling expert logical rules on knowledge graphs.

Figure 1 shows examples from the dataset Family. There exist four expert rules. A rule R' is said to *depend on* a rule R if the head predicate of R is a predicate in the body of R'. In Figure 1, R_1 depends on R_2, R_2 depends on R_3 or R_4. The triple "aunt(Diana, Tom)" derived from KG_1 in Figure 1 must be applied with R_2 followed by R_1. There also exist some self-dependent rules, namely R_1, R_3 and R_4. The triple "sister(Mary, Diana)" derived from KG_2 in Figure 1 must be applied with R_3 twice. Due to rule dependencies, utilizing existing rules to derive new triples from existing triples typically requires multi-step reasoning. However, existing neural-based rule learning models typically employ single-step reasoning for rules. Assuming that expert rules are correct, it follows that all their logical entailments are also correct. Thus, the challenging of incremental rule learning beyond expert rules lies in how to enable single-step reasoning without losing logical entailments of expert rules.

To tackle this challenge, we present a framework to allow existing end-to-end rule learning approaches to incrementally learn rules beyond expert logical rules. To facilitate the integration of rules into neural networks for single-step reasoning, we introduce the notion of *complete onehop-transformed set of rules*, which preserves all logical entailments of expert rules by single-step reasoning. Moreover, we propose the formal definition of *reasoning path* to obtain a complete onehop-transformed set of rules. Considering the computational expense associated with deriving reasoning paths, we have developed an optimization algorithm based on backward reasoning. Subsequently, we propose an end-to-end rule learning model to learn rules based on the computed complete onehop-transformed set of rules, by building upon any existing rule learning approach.

We conduct empirical evaluations on four benchmark datasets. Experimental results indicate that the integration of expert rules significantly improves the performance of link prediction on knowledge graphs. Additionally, comparative experiments have demonstrated the efficacy of the complete onehop-transformed set of rules.

2 Related Work

Rule-based methods aim at building effective rule-based systems for KGC. Learning logical rules was previously studied in the field of Inductive Logic Programming (Quinlan, 1990; Muggleton, 1995; Schüller and Benz, 2018; Wu et al., 2022), where logical rules are learnt by a generate-and-test manner. Classical ILP methods such as FOIL (Quinlan and Cameron-Jones, 1995) and QuickFOIL (Zeng et al., 2014) cannot be directly applicable to KGs because there exist no negative examples and the data size is large. Recent ILP methods like AMIE+ (Galárraga et al., 2015) and AnyBURL (Meilicke et al., 2019) treat triples outside a KG as negative examples and can efficiently learn rules through different search algorithms.

More recently, there is an emerging interest in exploiting neural-based methods for rule learning. There exist some end-to-end neural approximate methods that learn continuous parameters based on the Tensorlog (Cohen et al., 2020) operators, such as NeuralLP (Yang et al., 2017a), DRUM (Sadeghian et al., 2019) and NLIL (Yang et al., 2017b). NeuralLP (Yang et al., 2017a) is the first neural approximate method using Tensorlog operators to learn chain-like rules. DRUM (Sadeghian et al., 2019) tackles the limitation of learning meaningless rules for NeuralLP by introducing the identity relation and bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to prune the potential incorrect rule bodies. Besides, FaithfulRE (Wang et al., 2024) is proposed to study faithfulness guarantees in the context of DRUM. In addition to applying Tensorlog, there are several methods that extend the generate-and-test manner to learn logical rules by neural models, including RNNLogic (Qu et al., 2021), RLogic (Cheng et al., 2022) and NCLR (Cheng et al., 2023). Besides, other neural approximate methods such as NTP (Rocktäschel and Riedel, 2017) and CTP (Minervini et al., 2020) learn logical rules based on neural theorem provers.

Compared with ILP methods, neural-based methods are better at learning rules from imperfect data. However, all these methods learn rule-based systems from structural knowledge alone, ignoring the

existence of known expert logical rules.

3 Preliminaries

3.1 Basic concepts

An *atom* α is a basic first-order logic formula of the form $p(t_1, \cdots, t_k)$ where p is *predicate* with arity k, and the t_i are *terms* that denote either variables or constants. Given an atom α, pred(α),vars(α), consts(α) and terms(α) denote respectively, its predicate, its set of variables, of constants and of terms, which can be naturally extended to a set of atoms. A fact is an atom without variables.

Given a set of variables V and a set of terms T, a *substitution* σ of V by T is a mapping from V to T. It is called *ground* if it maps to only constants. Given an atom α, $\sigma(\alpha)$ denotes the atom obtained from α by replacing each variable x in α with $\sigma(x)$. This naturally extends to sets of atoms.

A *plain datalog rule* (simply a *rule*) R is a formula $r(\vec{x}) \leftarrow \exists \vec{y} : \phi(\vec{x}, \vec{y})$, where $\phi(\vec{x}, \vec{y})$ is a conjunction of atoms on $\vec{x}$ and $\vec{y}$, $\vec{x}$ and $\vec{y}$ are sets of variables, r denotes the predicate of the atom inferred by the rule. The part of R at the left (resp. right) of $\leftarrow$ is called the *head* (resp. *body*) of R. By H_R and B_R we denote the atom in the head of R and the set of atoms in the body of R, respectively. The number of atoms in B_R is called the *length* of R. A fact γ can be regarded as a rule R s.t. $B_R = \emptyset$. Let R and R' be two rules. R *subsumes* R' if there exists a substitution σ s.t. $\sigma(H_R) = H_{R'}$ and $\sigma(B_R) \subseteq B_{R'}$.

Let R be a rule, Σ a set of rules, Δ a set of atoms. Δ is said to be a *model* of R if for any substitution σ, if $\sigma(B_R) \subseteq \Delta$, then $\sigma(H_R) \in \Delta$. Δ is said to be a *model* of Σ if for any $R \in \Sigma$, Δ is the model of R. Rules considered in this paper are plain datalog rules, thus there exists a unique least model of Σ.

Let Γ be a set of facts. Let $\Delta_0 = \Gamma$, $\Delta_t = \Delta_{t-1} \cup \{\sigma(H_R) | \sigma(B_R) \subseteq \Delta_{t-1}, R \in \Sigma$ and σ is a ground substitution$\}$. When $\Delta_{t+1} = \Delta_t$, Δ_t is the *fixpoint* of Γ w.r.t. Σ. Δ_t is the unique least model of $\Sigma \cup \Gamma$. A fact γ is said to be *entailed by* $\Gamma \cup \Sigma$ if any model of $\Gamma \cup \Sigma$ contains γ; i.e., $\gamma \in \Delta_t$, denoted by $\Gamma \cup \Sigma \models \gamma$.

3.2 Knowledge graph

Let $\mathcal{E}$ be a set of entities, $\mathcal{R}$ a set of relations. A *knowledge graph* $\mathcal{G}$ is a subset of $\{(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where h denotes the head entity, r the relation and t the tail entity. Note that (h, r, t) can be formed into a binary fact $r(h, t)$. By r^- we denote the inverse relation of $r \in \mathcal{R}$. Accordingly, the equivalent set of triples for $\mathcal{G}$ composed by inverse relations, namely $\{(t, r^-, h) | (h, r, t) \in \mathcal{G}\}$, is denoted by $\mathcal{G}^-$.

Given a knowledge graph $\mathcal{G}$, let $\mathcal{K} = \mathcal{G} \cup \mathcal{G}^-$, we say a triple (h, r, t) or a fact $r(h, t)$ is *inferred* (resp. *potentially inferred*) from $\mathcal{G}$ if there exists (resp. could be) a set of rules Σ s.t. $\mathcal{K} \cup \Sigma \models r(h, t)$. Given a head query $(?, r, t)$ or a tail query $(h, r, ?)$, the task of *link prediction* aims to find all entities $e \in \mathcal{E}$ such that (e, r, t) for $(?, r, t)$ or (h, r, e) for $(h, r, ?)$ is potentially inferred from $\mathcal{G}$.

4 Approach

4.1 Problem Definition

Traditional rule learning methods on knowledge graph directly learn logical rules from numerous instances. Acknowledging the presence of expert rules, we propose incremental rule learning in this paper. Let $\mathcal{G}$ be a given knowledge graph, Σ be a set of plain datalog rules, $\mathcal{K} = \mathcal{G} \cup \mathcal{G}^-$. Given an unverifed triple (h, r, t), the problem is to calculate the truth degree $P_{r(h,t)}$ of the triple $(h, r, t) \in \mathcal{E} * \mathcal{P} * \mathcal{E}$, where $P_{r(h,t)}$ reflects the degree of whether the triple (h, r, t) can be potentially inferred by the rules learned from $\mathcal{K}$ and Σ.

4.2 Overview of Approach

Assuming that all rules in Σ are correct, it follows that all logical entailments of Σ are also correct. Thus, it is crucial to ensure the preservation of the reasoning semantics of expert rules while learning rules based on expert rules. Considering the dependency among rules mentioned in the example of Figure 1, multi-step reasoning is essential for inferring all triples. To integrate expert rules into neural networks for single-step reasoning, we introduce the notion of *onehop-transformed set of rules* to optimize the architecture of the neural network.

Definition 1. Let Σ, Ω be two sets of rules. Let Γ be a specific set of fact. Ω is called a Γ-*specific onehop-transformed set of rules* from Σ if for any fact $\gamma \notin \Gamma$, $\Gamma \cup \Sigma \models \gamma$ if there is a rule $R \in \Omega$ and a substitution σ for R s.t. $\sigma(B_R) \subseteq \Gamma$ and $\sigma(H_R) = \gamma$. Ω is further called *complete* if for any fact $\gamma \notin \Gamma$, $\Gamma \cup \Sigma \models \gamma$ iff there is a rule $R \in \Omega$ and a substitution σ for R s.t. $\sigma(B_R) \subseteq \Gamma$ and $\sigma(H_R) = \gamma$.

To preserve all logical entailments of the expert rules Σ, we need to calculate a complete $\mathcal{K}$-specific onehop-transformed set Ω from Σ. In view of the

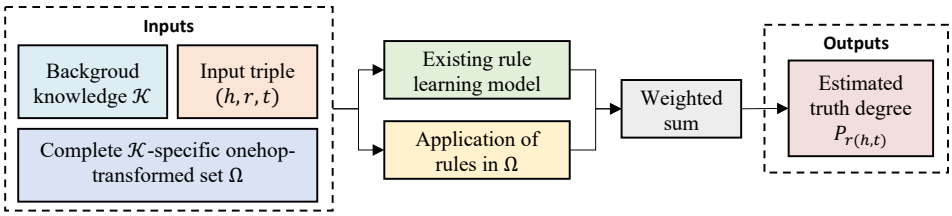

Figure 2: Illustration of the proposed incremental end-to-end rule learning model.

inference efficiency of the rule learning models based on $\mathcal{K}$ and Ω, it is desirable to have as few rules as possible in Ω. Consequently, we proceed to remove any redundant rules that are subsumed by other rules within Ω.

Since rules in Σ are plain datalog rules, all triples entailed by $\mathcal{K} \cup \Sigma$ are in the fixpoint of $\mathcal{K}$ w.r.t. Σ. To obtain the rules for single-step reasoning, we calculate the *reasoning path*, defined as follows, for each fact in the fixpoint of $\mathcal{K}$ w.r.t. Σ. Through substituting constants in reasoning paths, a rule in onehop-transformed set can be derived. In EX 1 of Figure 1, {aunt(Diana, Tom) ← aunt(Mary, Tom) ∧ sister(Diana, Mary), aunt(Diana, Tom) ← sister(Mary, Alice) ∧ son(Tom, Alice) ∧ sister(Diana, Mary)} is a reasoning path for the fact aunt(Diana, Tom) w.r.t. σ and KG_1.

Definition 2. Given a set of rules Σ and a set of facts Γ. $\Delta_{\Gamma,\Sigma}$ is the fixpoint of Γ w.r.t. Σ. For any $\gamma \in \Delta_{\Gamma,\Sigma} \setminus \Gamma$, there exists a set of ground rules $\{\sigma_1(R_1), \cdots, \sigma_m(R_m)\}$ s.t. $R_i \in \Sigma$, σ_i is a substitution for R_i, $\sigma_1(R_1) = \gamma$ and let $P_1 : \gamma \leftarrow \sigma_1(B_{R_1})$, $P_j : \gamma \leftarrow B_{P_{j-1}} \cup \sigma_j(B_{R_j}) \setminus \sigma_i(H_{R_j})$, $B_{P_m} \subseteq \Gamma$, $1 < j \leq m$, $\{P_1, \cdots, P_m\}$ is a *reasoning path* of γ w.r.t. Γ and Σ. P_m is the corresponding ground rule.

As depicted in Definition 2, computing a reasoning path of a given fact requires considering all the substitutions for all rules, leading to high computational cost. Consequently, it is impractical to obtain a complete $\mathcal{K}$-specific onehop-transformed set by calculating reasoning paths for all facts in the fixpoint of $\mathcal{K}$ w.r.t. Σ while excluding the facts already in $\mathcal{K}$. We have discovered that by employing backward reasoning which considers only the rule set without the facts, we can identify the single-step reasoning rules for most new facts. Therefore, we introduce backward reasoning as an optimization algorithm. Ultimately, we are required to compute reasoning paths only for those facts that cannot be inferred through single-step reasoning of the rules obtained via backward reasoning. By substituting constants, we can obtain a complete $\mathcal{K}$-specific onehop-transformed set.

After calculating a complete $\mathcal{K}$-specific onehop-transformed set Ω from Σ, we propose a model for learning rules based on $\mathcal{K}$ and Ω. An overview of the proposed model is illustrated in Figure 2. Intuitively, the model aggregates the rules in Ω along with those from existing rule learning models to obtain the final estimated truth degrees. Though rules in Ω are accurate, it is worth to note that the facts in $\mathcal{K}$ may contain errors (e.g., outdated facts), rendering a direct application of the rules in Ω inappropriate. Moreover, each rule R in Ω with the same predicate in H_R can be conceptualized as an individual classifier. Similarly, rules from existing rule learning model also can be regarded as classifiers. By employing the weighted sum, a widely-used ensemble learning technique, the final estimated truth degree is calculated.

4.3 Rule Completion

Given a set of facts $\mathcal{K}$ and a set of plain datalog rules Σ, the target of rule completion is to calculate a complete $\mathcal{K}$-specific onehop-transformed set Ω of rules from Σ.

Let $\Delta_{\mathcal{K},\Sigma}$ be the fixpoint of $\mathcal{K}$ w.r.t. Σ. There is a reasoning path of each fact $\gamma \in \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}$. Algorithm 1 is developed to extract a reasoning path and substitute constants. The function GetReasoningPath() is designed to extract reasoning paths of δ based on the definition of reasoning path. It adopts a depth-first search strategy to recursively search the reasoning path of δ and obtains the corresponding ground rule R until all facts in Γ are in $\mathcal{K}$. The function LiftRule() is designed to map constants in the corresponding ground rule into variables. During the process of mapping, we need to ensure that the *confidence* of the rule is equal to 1.0. Let $\Gamma = \{\sigma(H_R) | \sigma(B_R) \subseteq \mathcal{K}, \sigma \text{ is a substitution for } R\}$, the *confidence* of a rule R is defined as the ratio of the size of $\Gamma \cap \Delta_{\mathcal{K},\Sigma}$, and the size of Γ .

Theorem 1. Given a set of facts $\mathcal{K}$, a set of rules

Algorithm 1: GetRule($\mathcal{K}$, Σ, $\Delta_{\mathcal{K},\Sigma}$, t, m)

Input: A set of facts $\mathcal{K}$, a set of rules Σ, the fixpoint $\Delta_{\mathcal{K},\Sigma}$ of $\mathcal{K}$ w.r.t. Σ, the minimum number of iterations t, a fact $\gamma \in \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}$, and m.
Output: A rule R_γ.

$\Phi \leftarrow \{\}$;
GetReasoningPath(γ, 0, $\emptyset$);
$R_\gamma \leftarrow \text{argmax}_{R' \in \Phi} |\text{vars}(R')|$;
Function GetReasoningPath(δ, l, Γ):
  if $l = t$ or $|\Phi| \geq m$ **then**
    return;
  for $R \in \Sigma$ s.t. pred(H_R) = r **do**
    for substitution σ s.t. $\sigma(H_R) = \delta$, $\sigma(B_R) \subseteq \Delta_{\mathcal{K},\Sigma}$ **do**
      $\Gamma \leftarrow \Gamma \cup \{\sigma(B_R)\}$;
      if $\forall \alpha \in \Gamma$ *s.t.* $\alpha \in \mathcal{K}$ **then**
        $R' \leftarrow$ LiftRule(Γ);
        $\Phi \leftarrow \Phi \cup \{R'\}$;
      for $\alpha \in \Gamma$ *s.t.* $\alpha \notin \mathcal{K}$ **do**
        GetReasoningPath(α, $l+1$, $\Gamma \setminus \{\alpha\}$);
Function LiftRule(Γ):
  $H_{R'} \leftarrow \gamma$;
  $B_{R'} \leftarrow \Gamma$;
  $R \leftarrow R'$;
  for constant $c \in \text{consts}(R')$ **do**
    $R'' \leftarrow$ mapping(c, R);
    $s \leftarrow$ confidence(R'', $\mathcal{K}$, $\Delta_{\mathcal{K},\Sigma}$) ;
    if $s == 1.0$ **then**
      $R \leftarrow R''$;
  return R
return R_γ

Σ, the fixpoint $\Delta_{\mathcal{K},\Sigma}$ of $\mathcal{K}$ w.r.t. Σ, the minimum number of iterations t , a fact $\gamma \in \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}$, and the maximum number m of inference paths discovered , R_γ = ExtractRule($\mathcal{K}$, Σ, $\Delta_{\mathcal{K},\Sigma}$, t, m) is a rule s.t. there is a substitution σ for R_γ s.t. $\sigma(B_{R_\gamma}) \subseteq \mathcal{K}$ and $\sigma(H_{R_\gamma}) = \gamma$.

Proof. In Algorithm 1, $\gamma \in \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}$, m is the max number of reasoning paths found for γ. The function GetReasoningPath() obtains the corresponding ground rule R until all facts in Γ are in $\mathcal{K}$. $H_R = \gamma$. Then the corresponding ground rule R is lifted into R' through function LiftRule(). Abviously, there is a substitution σ for R' s.t. $\sigma(H_{R'}) = H_R$ and $\sigma(B_{R'}) = B_R$. When the number of reasoning paths found for the missing fact γ is greater than m, the algorithm terminates and returns the rule R_γ with the most variables in Φ. There is a substitution σ for R_γ s.t. $\sigma(B_{R_\gamma}) \subseteq \mathcal{K}$ and $\sigma(H_{R_\gamma}) = \gamma$. □

Due to the computational expense of obtaining reasoning paths and substituting constants, we propose a backward reasoning algorithm in Algorithm 2 to compute a possibly incomplete $\mathcal{K}$-specific onehop-transformed set Ψ from Σ, which is implemented based on the *rewriting* of a rule defined as follows.

Algorithm 2: BackwardReasoning()

Input: A set of rules Σ, a relation r, a max length of L, a set of facts $\mathcal{K}$ and the fixpoint $\Delta_{\mathcal{K},\Sigma}$ of $\mathcal{K}$ w.r.t. Σ.
Output: A rule set Ψ_r.

$R_0 \leftarrow$ "$r(x,y) \leftarrow r(x,y)$";
$\Psi_r \leftarrow \{\}$;
$\mathcal{Q} \leftarrow$ A sorted queue constructed of rules, ordered by the length of rules in ascending order;
$\mathcal{Q}$.add(R_0);
while $\mathcal{Q} \neq \emptyset$ **do**
  $R_c \leftarrow \mathcal{Q}$.poll();
  $\Psi_r \leftarrow \Psi_r \cup \{R_c\}$;
  $\Phi \leftarrow \emptyset$;
  for $R \in \Sigma$ **do**
    for substitution μ that can rewrite R_c w.r.t. R **do**
      $R' \leftarrow$rewriting(R_c, R, μ);
      if $len(R') > L$ **then**
        if R' can be collapsed **then**
          $\Phi \leftarrow \Phi \cup$ Collapse(R');
      else
        $\Phi \leftarrow \Phi \cup \{R'\}$;
  $\Phi \leftarrow$ filterLength(Φ, L) ;
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup (\Phi \setminus (\Psi_r \cup \mathcal{Q}))$
$\Psi_r \leftarrow$ filterConfidence(Ψ_r, $\mathcal{K}$, $\Delta_{\mathcal{K},\Sigma}$);
$\Psi_r \leftarrow$ filterSupport(Ψ_r, $\mathcal{K}$);
Function Collapse(R):
  $\Phi \leftarrow \{\}$;
  for substitution σ that can collapse R **do**
    $R' \leftarrow$ collapsing(R, σ);
    $\Phi \leftarrow \Phi \cup \{R'\}$;
    if R' can be collapsed **then**
      $\Phi \leftarrow \Phi \cup$ Collapse(R');
  return Φ;
return Ψ_r

Definition 3. Given two rules R and R'. A rule R'' is a *rewriting* of R w.r.t. R' if there exists a substitution σ s.t. $\sigma(H_{R'}) \subseteq B_R$, $H_{R''} = H_R$ and $B_{R''} = B_R \cup \sigma(B_{R'}) \setminus \{\sigma(H_{R'})\}$.

Considering the process of backward reasoning may be infinite, we introduce the *collapsing* of the rule under the constraint of the rule length.

Definition 4. Given a rule R. Let α and β be two atoms in B_R. A rule R' is a *collapsing* of R if there exists a substitution σ s.t. $\sigma(\alpha) = \sigma(\beta)$, $H_{R'} = \sigma(H_R)$ and $B_{R'} = \sigma(B_R) \setminus \{\sigma(\alpha)\}$.

For each relation r, Algorithm 2 calculates a rules set Ψ_r s.t. for any $R \in \Psi_r$, pred(H_R) = r. As described in Algorithm 2, we start rewriting rules

Algorithm 3: RemoveRedundant(Φ)

Input: A set of rules Φ.
Output: An irreducible subset of rules Ψ.

```
Ψ ← Φ;
for each R ∈ Ψ do
    for each R' ∈ Ψ do
        if R ≠ R' and R' subsumes R then
            Ψ ← Ψ \ {R};
            break;
return Ψ
```

from "$r(x, y) \leftarrow r(x, y)$" and proceed by traversing rules in Σ. The function rewriting() is crafted to rewrite the rule R_c by the rule R based on the substitution μ. Given the constraint on the length of rules, we introduce the function Collapse() to generate all possible collapsings of a given rule. The function collapsing() is to collapse the rule R based on σ. The function filterLength() is utilized to remove rules whose length is larger than L. In order to ensure the correctness of rewriting rules, the function filterConfidence() is designed to filter rules whose confidence is less than 1.0. Furthermore, the function filterSupport() is used to preserve the rules in Ψ_r where for each R there exist a substitution σ s.t. $B_R \subseteq \mathcal{K}$. For any fact γ s.t. there is a rule $R \in \Psi_r$ and a substitution for R s.t. $\sigma(B_R) \subseteq \mathcal{K}$ and $\sigma(H_R) = \gamma$, $\mathcal{K} \cup \Sigma \models \gamma$. Therefore, $\Psi = \{\Psi_r | r \in \mathcal{R}\}$ is a $\mathcal{K}$-specific onehop-transformed set of rules from Σ. However, Ψ is possibly incomplete. Let $\Gamma_\Psi = \{\sigma(H_R) | R \in \Psi, \sigma(B_R) \subseteq \mathcal{K}, \sigma$ is a substitution for $R\}$. We complete Ψ to Π by invoking Algorithm 1 for each fact $\gamma \in \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}, \gamma \notin \Gamma_\Psi$. There may be redundant rules in Π. We can obtain an irreducible subset of rules Ω by removing the rules in Π that are subsumed by others.

Definition 5. Let Φ be a set of rules. An *irreducible subset of rules* Ψ of Φ is a subset of Φ s.t. for any $R' \in \Psi$, there is no $R \in \Psi$ such that R subsumes R'.

The algorithm is shown in Algorithm 3 and Ω is a complete $\mathcal{K}$-specific onehop-transformed set from Σ as illustrated in Theorem 2.

Theorem 2. Given a set of facts $\mathcal{K}$, a set of rules Σ, the fixpoint $\Delta_{\mathcal{K},\Sigma}$ of $\mathcal{K}$ w.r.t. Σ, the max number m of reasoning paths found for a fact, a $\mathcal{K}$-specific onehop-transformed set Ψ of rules from Σ. Let $\Gamma_\Psi = \{\sigma(H_R) | R \in \Psi, \sigma(B_R) \subseteq \mathcal{K}, \sigma$ is a substitution for $R\}$, $\Phi = \{\text{GetRule}(\mathcal{K}, \Sigma, \Delta_{\mathcal{K},\Sigma}, m) \,|\gamma \in \Delta_{\mathcal{K},\Sigma} \setminus (\mathcal{K} \cup \Gamma_\Psi)\}$, $\Omega = \text{RemoveRedundant}(\Psi \cup \Phi)$. Ω is a complete $\mathcal{K}$-specific onehop-transformed set from Σ.

Proof. Let $\Gamma_\Phi = \{\sigma(H_R) | R \in \Phi, \sigma(B_R) \subseteq \mathcal{K}, \sigma$ is a substitution for $R\}$. Obviously, $\Gamma_\Phi \cup \Gamma_\Psi = \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}$. According to the Algorithm 3, for any rule R in $\Psi \cup \Phi$, there is a rule R' in Ω s.t. there exists a substitution σ for R' s.t. $\sigma(H_{R'}) = H_R$ and $\sigma(B_{R'}) \subseteq B_R$. Let $\Gamma_\Omega = \{\sigma(H_R) | R \in \Omega, \sigma(B_R) \subseteq \mathcal{K}, \sigma$ is a substitution for $R\}$. Obviously, $\Gamma_\Omega = \Gamma_\Phi \cup \Gamma_\Psi = \Delta_{\mathcal{K},\Sigma} \setminus \mathcal{K}$. Then for any fact $\gamma \notin \Gamma$, $\Gamma \cup \Sigma \models \gamma$ iff there is a rule $R \in \Omega$ and a substitution σ for R s.t. $\sigma(B_R) \subseteq \Gamma$ and $\sigma(H_R) = \gamma$. □

4.4 Rule Learning

To learn rules based on expert rules in an end-to-end manner, we propose a neural model based on utilizing traditional end-to-end rule learning methods. As shown in Figure 2, the proposed model mainly includes two modules: existing rule learning model and application of rules in Ω.

For the existing rule learning model, we select three neural-based models: Neural-LP (Yang et al., 2017a), DRUM (Sadeghian et al., 2019), sm-DRUM (Wang et al., 2024), implemented utilizing Tensorlog. For each step, the model calculates the intermediate truth degrees based on estimating predicate weights and predicate selection for chained atoms. All of them generate N rules to calculate the truth degree $M_{r(h,t)}$ of the triple (h, r, t):

$$M_{r(h,t)} = \sum_{j=0}^{N} b_j M_{r(h,t)}^{R_j} \tag{1}$$

where b_j is trainable and $M_{r(x,y)}^{r_j}$ represents the truth degree of (h, q, t) based on j-th rules from traditional end-to-end rule learning methods. b_j is constrained to $[0, 1]$ by a sigmoid layer. Intuitively, the use of the sigmoid function enables the model to learn a positive weight for each rule.

Incremental end-to-end rule learning mines rules based on Ω_r. Application of rules in Ω is to compute all scores $S_{r(h,t)}^{R_i}$, where $R_i \in \Omega_r$. The score $S_{r(h,t)}^{R_i}$ represents the confidence that $r(h, t)$ can be inferred from R_i. $\Gamma_{R_i} = \{\sigma(H_{R_i}) | \sigma(B_{R_i}) \subseteq \mathcal{K}, \sigma$ is a substitution for $R_i\}$.

Then $S_{r(h,t)}^{R_i}$ can be defined as:

$$S_{r(h,t)}^{R_i} = w_i \mathbf{I}(r(h, t) \in \Gamma_{R_i}) \tag{2}$$

where $w_i \in \mathbf{w}$ is a trainable parameter that represents the weight of R_i. $\mathbf{I}(\phi)$ is an indicator function

that returns 1 if ϕ is true or 0 otherwise. The size of Ω_r and $\mathbf{w}$ are both K. w_i is confined to $[0, 1]$ by a sigmoid layer. The background rules Ω_r integrated into rule learning is defined as follows.

$$S_{r(h,t)} = \sum_{R_i \in \Omega_r} S^{R_i}_{r(h,t)} \tag{3}$$

Then the estimated truth probability $P_{r(h,t)}$ is calculated by:

$$P_{r(h,t)} = M_{r(h,t)} + S_{r(h,t)} \tag{4}$$

The incremental model is trained by minimizing the following objective function.

$$\mathcal{L} = - \sum_{(h,r,t) \in \mathcal{K}} \log P_{r(h,t)} \tag{5}$$

Incremental rule learning requires maintaining all logical entailments of the given expert rules, which can be theoretically proved by Theorem 3.

Theorem 3. Let $\mathcal{G}$ be a knowledge graph, $\mathcal{K} = \mathcal{G} \cup \mathcal{G}^-$, Σ a set of existing rules, and Ω a complete $\mathcal{K}$-specific onehop-transformed set from Σ. Suppose $\forall 1 \leq i \leq K$: $w_i = 1$, for an arbitrary triple (h, r, t), $\mathcal{K} \cup \Sigma \models r(h, t)$ iff $S_{r(h,t)} \geq 1$.

Proof. If $\mathcal{K} \cup \Sigma \models r(h, t)$, there exists a rule $R_i \in \Omega_r$ and a substitution σ for R s.t. $\sigma(H_R) = r(h, t)$ and $\sigma(B_R) \subseteq \mathcal{K}$. Then $S^{R_i}_{r(h,t)} = 1$. According to Equation 2, $S_{r(h,t)} \geq 1$. If $S_{r(h,t)} \geq 1$, then there exist a $R_i \in \Omega_r$ s.t. $S^{R_i}_{r(h,t)} = 1$ due to that each $w_i = 1$. Then we can conclude that $r(h, t) \in \Gamma_{R_i}$. Thus, $\mathcal{K} \cup \Sigma \models r(h, t)$. □

5 Experimental Evaluation

5.1 Experimental Setups

5.1.1 Datasets

To compare our incremental rule learning with tradition rule learning for KGC, we used four well-known benchmark datasets for empirical evaluation, including UMLS (Kok and Domingos, 2007), Family (Yang et al., 2017b), CODEX-S (Safavi and Koutra, 2020), NELL-995 (Xiong et al., 2017b). Statistics for these datasets are reported in Table 1.

5.1.2 Evaluation Metrics

Following (Sadeghian et al., 2019), we computed the truth degrees for corrupted triples and computed the rank of the correct answer. Based on the rank, we calculate the Mean Reciprocal Rank(MRR) and Hit@k metrics for empirical evaluation under the filtered setting introduced by (Bordes et al., 2013). Following (Qu et al., 2021), for a query $r(?, t)$, the rank of the correct answer is defined by $i+(j+1)/2$ in our proposed setting, where i is the number of entities with higher truth degrees than the correct answer, and j is the number of entities with the same truth degree as the correct answer.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\Sigma\rvert$	$\lvert\mathcal{G}_{train}\rvert$	$\lvert\mathcal{G}_{valid}\rvert$	$\lvert\mathcal{G}_{test}\rvert$
UMLS	135	46	5327	569	633
Family	3007	12	23484	2038	2835
CODEX-S	2034	42	32888	1827	1828
NELL-995	57016	199	118304	11221	11099

Table 1: Statistics for experimental datasets.

	UMLS	Family	CODEX	NELL-995
max-length	2	2	2	3
$\lvert\Sigma\rvert$	44	36	18	143
$\lvert\Omega\rvert$	303	260	53	245

Table 2: Statistics for expert rules for experiment. Max-length, $|\Sigma|$, $|\Omega|$ represent the max length of extracted rules, the number of original rules and the number of rules completed respectively.

5.1.3 Implementation Details

Due to the lack of the background rules, we utilize an ILP method AnyBURL (Meilicke et al., 2019) to extract rules from KGs. We set a high threshold to ensure the extracted rules are as accurate as possible. Then, we manually screen the extracted rule set to obtain the plain datalog rules for incremental rule learning. Statistics for background rules are reported in Table 2.

Algorithms in Section 4.3 are implemented in Java. And we implement the incremental end-to-end rule learning based on three neural-based methods: NeuralLP, DRUM, smDRUM. The implementation environments of these methods are respectively followed. We implemented the end-to-end rule learning model on an NVIDIA A100 GPU with 40GB RAM. The model was trained by Adam (Kingma and Ba, 2015) with 10 training epochs. An early stopping strategy was applied to maximize the MRR score on the validation set. The initial learning rate was set to 1e-3 and the mini-batch size was set to 64. We applied dropout (Srivastava et al., 2014) by setting the dropout rate to 0.1. The maximum length of rules was set to 3. The number of rules learnt from existing rule learning method was set 1 for NeuralLP, 3 for DRUM and smDRUM. All experimental results are the average of experiments conducted with 5 different random

	UMLS				Family				CODEX-S				NELL-995			
	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	MRR
NeuralLP	66.0	93.5	96.8	0.800	87.5	97.4	98.7	0.926	31.1	45.5	58.9	0.407	36.3	50.0	**62.8**	0.455
Incre-NeuralLP	86.1	94.4	97.0	0.906	97.5	98.7	**99.0**	0.982	**33.6**	45.1	58.2	**0.420**	36.6	49.9	62.1	0.455
DRUM	69.5	94.1	97.4	0.822	89.8	97.6	98.9	0.938	31.0	**45.8**	**59.2**	0.408	36.4	50.0	62.7	0.455
Incre-DRUM	86.1	**94.6**	97.5	0.908	97.9	**98.8**	**99.0**	**0.984**	32.3	45.2	57.8	0.412	**37.2**	**50.4**	62.5	**0.460**
smDRUM	69.0	93.1	**98.1**	0.818	89.5	97.5	98.9	0.937	17.8	29.8	48.0	0.282	28.3	41.6	54.5	0.379
Incre-smDRUM	**86.7**	94.3	97.5	**0.910**	**98.0**	**98.8**	**99.0**	**0.984**	20.9	32.5	49.8	0.308	28.1	40.9	53.8	0.374

Table 3: Comparison results on four benchmark datasets. DRUM, NeuralLP, and smDRUM denote the results trained with the original models. The "Incre-" versions represent the proposed models for incremental rule learning.

seeds. For each metric and dataset, the best results are highlighted in bold.

5.2 Main Empirical Results

We conducted experiments on four datasets for link prediction. As mentioned before, we propose a novel setting to learn rules upon the existing expert rules. We verify the effectiveness of the framework upon three existing rule learning methods.

Table 3 reports the comparison results on four datasets. Results show that for all the three existing rule learning methods, the incremental version significantly outperforms the original version on UMLS, Family, and CODEX-S. For example, Incre-DRUM outperforms DRUM by absolute gains of 16.6, 8.1 and 1.3 in the Hit@1 scores on UMLS, Family and CODEX-S, respectively. These results confirm that incremental rule learning is effective and helps to significantly improve performance in KGC. The results on NELL-995 indicate that incremental rule learning has an improvement for NeuralLP and DRUM, and can also achieve comparable performance for smDRUM.

As previously discussed, we propose to extract a complete $\mathcal{K}$-specific onehop-transformed set Ω from the given rules Σ to maintain all the logical entailments. Thus, to access the impact of rule completion, we conducted further experiments on UMLS and Family, as presented in Table 4. We train the incremental rule learning model using Σ and Ω separately. For the corresponding three existing rule learning methods, we conducted a statistical analysis on the results for both versions. Additionally, Table 4 also compares the outcomes of link prediction when using Σ alone versus Ω alone, with the aggregation method of rules being the maximum value selection. The results indicate that the performance of the model using Ω alone surpasses that of the model using Σ alone, thereby validating the significance of computing a complete $\mathcal{K}$-specific onehop-transformed set. Due to the dependence among rules, considering only single-step reasoning with Σ leads to a degradation in reasoning capability. As shown in Table 4, the performance of incremental rule learning on Ω is superior to that of models on Σ. Furthermore, the results suggest that the performance of incremental rule learning is better than that of the original models, regardless of which version of the rules is employed. This further proves that incremental learning based on existing rules can effectively improve the performance of link prediction.

	UMLS				Family			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
Σ	61.2	62.7	62.7	0.628	79.1	79.1	79.1	0.791
Ω	62.6	63.8	63.8	0.640	82.5	82.5	82.5	0.825
Neural-LP	66.0	93.5	96.8	0.800	87.5	97.4	98.7	0.926
Incre-NeuralLP(Σ)	85.1	94.4	97.0	0.900	96.6	98.6	**99.0**	0.976
Incre-NeuralLP(Ω)	86.1	94.4	97.0	0.906	97.6	98.7	**99.0**	0.982
DRUM	69.5	94.1	97.4	0.822	89.8	97.6	98.9	0.938
Incre-DRUM(Σ)	85.4	**94.8**	97.6	0.905	96.9	98.5	**99.0**	0.977
Incre-DRUM(Ω)	86.1	94.6	97.5	0.908	97.9	**98.8**	**99.0**	0.983
smDRUM	69.0	93.1	**98.1**	0.818	89.5	97.5	98.9	0.937
Incre-smDRUM(Σ)	85.8	94.2	98.0	0.905	96.8	98.4	**99.0**	0.977
Incre-smDRUM(Ω)	**86.7**	94.3	97.5	**0.910**	**98.0**	**98.8**	**99.0**	**0.984**

Table 4: Comparison results on UMLS and family to verify the effectiveness of query rewriting.

Additionally, we analyze the impact of learning rules of different length. The results are exhibited in Appendix A.1. And there is a case study for comparing the learned rules from different NeuralLP models on Family and UMLS in Appendix A.2.

6 Conclusion

In this paper, we have introduced a novel problem, named incremental rule learning, which mines rules upon the existing expert rules. Furthermore we develop an algorithm to compute a complete onehop-transformed set of rules to preserve the reasoning ability of expert rules. Subsequently, an incremental end-to-end model is proposed to learning rules incorporated with expert rules. We conduct experiments on four datasets, demonstrating the effectiveness of incremental rule learning.

7 Limitations

This paper proposes an end-to-end model to learn rules from KG by integrating expert logical rules. The employed ensembling methodology only involves learning a weight for each rule. In light of existing rule learning methods that learn rules directly in the schema level via representation-based model (Cheng et al., 2022, 2023), we will explore embedding the existing rules to further enhance the rule learning capabilities of our model.

References

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

Kewei Cheng, Nesreen K. Ahmed, and Yizhou Sun. 2023. Neural compositional rule learning for knowledge graph reasoning. In *ICLR*.

Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. 2022. Rlogic: Recursive logical rule learning from knowledge graphs. In *KDD*, pages 179–189.

William W. Cohen, Fan Yang, and Kathryn Mazaitis. 2020. Tensorlog: A probabilistic database implemented using deep-learning infrastructure. *J. Artif. Intell. Res.*, 67:285–325.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Stanley Kok and Pedro M. Domingos. 2007. Statistical predicate invention. In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 433–440.

Wenzhuo Liu, Xin-Jian Wu, Fei Zhu, Ming-Ming Yu, Chuang Wang, and Cheng-Lin Liu. 2025. Class incremental learning with self-supervised pre-training and prototype learning. *Pattern Recognit.*, 157:110943.

Xinze Lyu, Guangyao Li, Jiacheng Huang, and Wei Hu. 2020. Rule-guided graph neural networks for recommender systems. In *ISWC (1)*, volume 12506 of *Lecture Notes in Computer Science*, pages 384–401.

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

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. 2020. Learning reasoning strategies in end-to-end differentiable proving. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 6938–6949.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *AAAI*, pages 2779–2785.

Stephen H. Muggleton. 1995. Inverse entailment and progol. *New Gener. Comput.*, 13(3&4):245–286.

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

J. Ross Quinlan. 1990. Learning logical definitions from relations. *Mach. Learn.*, 5:239–266.

J. Ross Quinlan and R. Mike Cameron-Jones. 1995. Induction of logic programs: Foil and related systems. *New Generation Computing*, 13:287–312.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *NIPS*, pages 3788–3800.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. DRUM: end-to-end differentiable rule mining on knowledge graphs. In *NeurIPS*, pages 15321–15331.

Tara Safavi and Danai Koutra. 2020. Codex: A comprehensive knowledge graph completion benchmark. In *EMNLP (1)*, pages 8328–8350.

Peter Schüller and Mishal Benz. 2018. Best-effort inductive logic programming via fine-grained cost-based hypothesis generation - the inspire system at the inductive logic programming competition. *Mach. Learn.*, 107(7):1141–1169.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Xiaxia Wang, David Jaime Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. 2024. Faithful rule extraction for differentiable rule learning models. In *ICLR*.

Yigong Wang, Zhuoyi Wang, Yu Lin, Jinghui Guo, Sadaf Md. Halim, and Latifur Khan. 2023. Dual contrastive learning framework for incremental text classification. In *EMNLP (Findings)*, pages 194–206.

Lianlong Wu, Emanuel Sallinger, Evgeny Sherkhonov, Sahar Vahdati, and Georg Gottlob. 2022. Rule learning over knowledge graphs with genetic logic programming. In *ICDE*, pages 3373–3385.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017a. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*, pages 1271–1279.

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

Fan Yang, Zhilin Yang, and William W. Cohen. 2017a. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pages 2319–2328.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017b. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pages 2319–2328.

Yuan Yang and Le Song. 2020. Learn to explain efficiently via neural logic inductive learning. In *ICLR*.

Qiang Zeng, Jignesh M Patel, and David Page. 2014. Quickfoil: Scalable inductive logic programming. *Proceedings of the VLDB Endowment*, 8(3):197–208.

A Appendix

A.1 Analysis for learning rules of different length

The existing rule-based methods can learn rules with different lengths. To compare the impact of incremental rule learning on learning rules of different lengths, we conducted experiments through setting different length on Family and UMLS. Table 5 illustrates the comparison results. We can observe that the performance of link prediction tasks on both the Family and UMLS datasets remains relatively stable, regardless of the lengths of the rules learned by the neural network models. Due to the smaller scale of these two datasets and the simplicity of the rules, shorter rules are often sufficient to capture the necessary information for effective reasoning.

A.2 Case study on learned rules

In this paper, we learn rules by ensembling expert logical rules. And we first calculate a complete $\mathcal{K}$-specific onehop-transformed set Ω of rules from Σ. A fact inferred through single-step reasoning using a rule in Ω may require multi-step reasoning involving rules in Σ.

Besides, we present a case study for comparing the learned rules from different NeuralLP on Family and UMLS, as shown in Figure 4. "Original" represents the corresponding model is trained without any rules. "Incre(Σ)" and "Incre(Ω)" denote the models are trained by ensembling Σ and Ω respectively. We select the top 5 rules with the highest scores for analysis. It is illustrated that the Incre(Ω) approach has a greater capacity for rule extraction, successfully capturing a larger number of rules. This suggests that incremental rule learning may enhance the performance in identifying and utilizing relevant rules compared to its counterparts. In addition, we observe that the top rule R_1:$aunt(C, A) \leftarrow inv\_brother(B, A), aunt(C, B)$ learned by Original NeuralLP already exists in Σ in Figure 3. However, R_1 was not extracted from Incre(Σ) and Incre(Ω), which reflects that under incremental learning, neural networks tend to learn new rules. Similarly, rule R_2: $aunt(D, A) \leftarrow father(B, A), inv\_brother(C, B), sister(D, C)$ is extracted from Incre(Σ), but not from Incre(Ω), due to that R_2 exists in Ω from Figure 3. Additionally, according to our statistics, all the rules learned from Incre(Ω) that are not displayed in Ω, which demonstrates the effectiveness of rule learning with expert rules on knowledge graphs.

		UMLS				Family			
	length	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
NeuralLP		66.0	93.5	96.8	0.800	87.5	97.4	98.7	0.926
	2	86.2	93.9	96.5	0.904	97.2	98.7	98.9	0.979
Incre-NeuralLP	3	86.1	94.4	97.0	0.906	97.5	98.7	**99.0**	0.982
	4	85.6	94.2	96.9	0.902	97.8	98.8	**99.0**	0.983
DRUM		69.5	94.1	97.4	0.822	89.8	97.6	98.9	0.938
	2	85.6	**95.6**	**98.1**	0.911	97.9	98.8	**99.0**	**0.984**
Incre-DRUM	3	86.1	94.6	97.5	0.908	97.9	98.8	**99.0**	**0.984**
	4	**87.4**	**95.4**	97.7	**0.916**	97.7	98.8	**99.0**	0.983
smDRUM		69.0	93.1	**98.1**	0.818	89.5	97.5	98.9	0.937
	2	87.0	94.1	98.0	0.912	**98.0**	**98.9**	98.9	**0.984**
Incre-smDRUM	3	86.7	94.2	97.5	0.910	**98.0**	98.8	**99.0**	**0.984**
	4	85.8	93.4	97.0	0.902	97.7	98.7	**99.0**	0.983

Table 5: Results against different rule length on UMLS and family.

Σ	aunt(X,Y) :- aunt(X,A), brother(Y,A) aunt(X,Y) :- sister(X,A), mother(A,Y) aunt(X,Y) :- sister(X,A), son(Y,A) son(X,Y) :- brother(X,A), mother(Y,A) son(X,Y) :- brother(X,A), son(A,Y) son(X,Y) :- brother(X,A), father(Y,A) …
Ω	aunt(X,Y) :- mother(V1, Y), sister(X, V1) aunt(X,Y) :- sister(X, V1), son(Y, V1) aunt(X,Y) :- brother(Y, V1), father(V2, V1), sister(X, V2) aunt(X,Y) :- brother(Y, V1), mother(V2, V1), sister(X, V2) aunt(X,Y) :- aunt(X, V1), brother(Y, V1) aunt(X,Y) :- brother(Y, V1), father(V2, Y), sister(X, V2) …

Figure 3: A case study for comparing the learned rules from different rule learning systems on Family.

	Family	UMLS
Original	0.805 aunt(C, A) <- inv_brother(B, A), aunt(C, B) 0.101 aunt(C, A) <- inv_brother(B, A), sister(C, B) 0.046 aunt(D, A) <- inv_brother(B, A), sister(C, B), aunt(D, C) 0.012 aunt(C, A) <- inv_brother(B, A), inv_nephew(C, B)	0.085 uses(B, A) <- produces(B, A) 0.014 uses(C, A) <- produces(B, A), produces(C, B) 0.011 uses(B, A) <- assesses_effect_of(B, A) 0.011 uses(B, A) <- developmental_form_of(B, A)
Increl(Σ)	0.655 aunt(D, A) <- father(B, A), inv_brother(C, B), sister(D, C) 0.306 aunt(C, A) <- father(B, A), sister(C, B) 0.022 aunt(C, A) <- father(B, A), inv_brother(C, B)	0.066 uses(C, A) <- performs(B, A), performs(C, B) 0.052 uses(B, A) <- performs(B, A) 0.029 uses(C, A) <- treats(B, A), performs(C, B) 0.013 uses(C, A) <- performs(B, A), treats(C, B) 0.010 uses(C, A) <- measures(B, A), performs(C, B)
Increl(Ω)	0.188 aunt(C, A) <- father(B, A), sister(C, B) 0.123 aunt(C, A) <- inv_daughter(B, A), sister(C, B) 0.119 aunt(C, A) <- uncle(B, A), sister(C, B) 0.111 aunt(D, A) <- father(B, A), sister(C, B), sister(D, C) 0.107 aunt(C, A) <- inv_nephew(B, A), sister(C, B)	0.080 uses(B, A) <- inv_occurs_in(B, A) 0.040 uses(C, A) <- inv_occurs_in(B, A), inv_occurs_in(C, B) 0.020 uses(B, A) <- treats(B, A) 0.017 uses(C, A) <- treats(B, A), inv_occurs_in(C, B) 0.013 uses(C, A) <- assesses_effect_of(B, A), inv_occurs_in(C, B)

Figure 4: A case study for comparing the learned rules from different NeuralLP on Family and UMLS.