SLOGIC: SUBGRAPH-INFORMED LOGICAL RULE LEARNING FOR KNOWLEDGE GRAPH COMPLETION

Anonymous authors
Paper under double-blind review

ABSTRACT

Logical rule-based methods offer an interpretable approach to knowledge graph completion by capturing compositional relationships in the form of human-readable inference rules. However, current approaches typically treat logical rules as universal, assigning each rule a fixed confidence score that ignores query-specific context. This is a significant limitation, as a rule's importance can vary depending on the query. To address this, we introduce **SLogic** (Subgraph-Informed Logical Rule learning), a novel framework that assigns query-dependent scores to logical rules. The core of SLogic is a scoring function that utilizes the subgraph centered on a query's head entity, allowing the significance of each rule to be assessed dynamically. Extensive experiments on benchmark datasets show that by leveraging local subgraph context, SLogic consistently outperforms state-of-the-art baselines, including both embedding-based and rule-based methods.

1 INTRODUCTION

Knowledge graphs (KGs) capture complex relationships among diverse real-world entities, including commonly used ones such as people, locations, organizations, events, products, and concepts, as well as more specialized types such as genes, diseases, chemicals, publications, and technical terms. They are foundational to many applications, such as recommendation systems and question answering Hildebrandt et al. (2019); Lan & Jiang (2020). For instance, in a movie KG, a recommendation system can suggest a new film to a user by identifying that they have liked other movies directed by the same person or starring a particular actor.

Despite their usefulness, KGs are notoriously incomplete. To address this, the field of Knowledge Graph Completion (KGC) has explored several dominant paradigms. Embedding-based models (e.g., (Bordes et al., 2013; Yang et al., 2014; Dettmers et al., 2018; Trouillon et al., 2016; Sun et al., 2019)) learn vector representations of entities and relations to predict missing links. More recently, Graph Neural Networks (GNNs) based models (Zhang & Yao, 2022; Zhu et al., 2021; 2023) have achieved state-of-the-art performance by capturing intricate topological patterns within the graph's structure. However, the strength of both these approaches is also their primary weakness: their reasoning is opaque. Because they operate on dense, sub-symbolic vectors, they function as "black box" models, making it difficult to understand or trust their predictions in critical applications.

As an alternative, rule-based reasoning offers a transparent and explainable approach to KGC. These methods infer missing links by discovering and applying logical rules (e.g., $bornIn(X, Y) \wedge locatedIn(Y, Z) \rightarrow livesIn(X, Z)$), which are inherently human-readable. This interpretability is a significant advantage, providing not just a prediction, but also the logical path to reach it. While rule-based models offer inductive capabilities and can outperform many embedding-based methods, a common limitation is that they treat rules as universally static, assigning a fixed confidence score to each rule regardless of the context. This overlooks the fact that a rule's relevance can change dramatically depending on the specific query, and therefore, answers to a query should be *context-dependent*.

For a concrete example, consider the KGC query $livesIn(Person, ?)$. One plausible rule, $bornIn(X, Y) \wedge locatedIn(Y, Z) \rightarrow livesIn(X, Z)$, might link a person to their birth location. A second rule, $worksAt(X, Y) \wedge locatedIn(Y, Z) \rightarrow livesIn(X, Z)$, could link them to their place of work. A model relying on static rule confidences, perhaps favoring the globally more common

"birthplace" rule, would fail to distinguish the second rule is more relevant in the context of a CEO of a major tech company because the "$worksAt$" path becomes exceptionally reliable. Our approach uses query-specific subgraphs to dynamically assess the relevance of each potential reasoning path for a given query, capturing the nuance that static models miss.

To capture and utilize the local importance of a rule, we propose a novel hybrid approach that enhances rule-based reasoning with contextual awareness. **Our key innovation is to generate query-specific rule weights by synergizing the strengths of both symbolic rules and neural networks.** We first pre-calculate simple paths to serve as our explicit, interpretable rule base. We then employ a GNN not as the final predictor, but as a powerful context encoder for the query. By integrating the GNN-derived context with the logical paths, our model can dynamically assess which rules are most significant for any given query. This method retains the core explainability of a rule-based system while leveraging the contextual power of GNNs to achieve more accurate and nuanced inferences.

The rest of the paper is organized as follows. Section 2 briefly presents a literature review. Section 3 introduces the terminology, notations, and the research problem. Section 4 explains our proposed method. Section 5 presents the experimental results and Section 6 concludes the work.

2 Related Work

The literature on knowledge graph completion can be broadly categorized into embedding-based approaches (Bordes et al., 2013; Yang et al., 2014; Dettmers et al., 2018; Trouillon et al., 2016; Sun et al., 2019), graph neural network based methods (Zhang & Yao, 2022; Zhu et al., 2021; 2023), and rule-based strategies. As explained in Section 1, rule-based reasoning enhances the explainability of knowledge graph completion. We therefore focus on describing such methods.

Existing rule-based methods can be broadly classified according to how they discover and learn logical rules. A line of work focuses on learning rule confidences by actively searching for paths (groundings) on the graph during the training process. NeuralLP (Yang et al., 2017) and its successor DRUM (Sadeghian et al., 2019) use a differentiable framework inspired by TensorLog (Cohen et al., 2017) to find soft proofs. Similarly, RNNLogic (Qu et al., 2021) is a probabilistic model that treats logic rules as a latent variable and uses an Expectation-Maximization (EM) algorithm to simultaneously train a rule generator and a reasoning predictor. The second paradigm defers the search for rules at inference time, aiming to discover novel patterns not explicitly seen during training. RLogic (Cheng et al., 2022) and NCRL (Cheng et al., 2023) are recent examples that learn a neural function to score the quality of potential rule structures. They perform an exhaustive, brute-force enumeration of all possible rules up to a certain length. All prior efforts rely on rules deemed important at the level of the entire knowledge graph. In contrast, our proposed SLogic framework recalculates rule importance with respect to contextual entities and their surrounding subgraphs.

3 Preliminaries: Knowledge Graphs, Rules, and Completion

Knowledge Graphs, Rules. A knowledge graph (KG) is a structured representation of factual information, formally defined as a directed, multi-relational graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$. Here, $\mathcal{E}$ is a finite set of entities (nodes) and $\mathcal{R}$ is a finite set of relations (edge types). The graph's structure is composed of a set of factual triples $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Each triple (h, r, t) represents a known fact, where $h \in \mathcal{E}$ is a head entity, $t \in \mathcal{E}$ is a tail entity, and $r \in \mathcal{R}$ is the relation that connects them.

A **path** from a node v_i to a node v_j is defined as a sequence $v_i \xrightarrow{r_1} v_{i_1} \cdots \xrightarrow{r_m} v_j$, where each edge is labeled by a relation. **A path is simple** if it does not contain repeated nodes. The corresponding **relational path** is the ordered list of relations along the path, $(r_1, \ldots, r_m)$.

Our proposed method is based on logical rules. A **logical rule** is defined as a Horn clause, where the head is a single atomic formula (the conclusion) and the body is a conjunction of atomic formulas (the premises), i.e., each rule has the form: $r_h(X, Y) \leftarrow r_1(X, Z_1) \wedge r_2(Z_1, Z_2) \wedge \cdots \wedge r_L(Z_{L-1}, Y)$. Here, the rule asserts that the target relation r_h is likely to hold between entities X and Y if there exists an intermediate path connected by a sequence of relations, where the maximum body length L is a predefined hyperparameter. For conciseness, we represent the relational path in the rule body as a single vector, $\mathbf{r}_b = (r_1, r_2, \ldots, r_L)$, which allows us to simplify the rule notation to $r_h \leftarrow \mathbf{r}_b$.

A rule $r_h \leftarrow \mathbf{r}_b$ is locally applicable w.r.t. an entity h, if its body path $\mathbf{r}_b$ can be successfully grounded starting from the head entity h. A **hard negative rule w.r.t. a triplet** (h, r, t) is defined to be a rule that is both *globally high-quality* (high static confidence) and *locally applicable*.

This paper addresses the knowledge graph completion (KGC) problem. Given a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ and a query $\mathbf{q} = (h, r, ?)$, the task is to identify the most plausible answer entities.

4 THE SLOGIC FRAMEWORK

Unlike traditional methods that assign static confidence scores to logical rules, SLogic learns a dynamic, query-dependent scoring function $\phi(h, r, \mathbf{r}_b)$. This function assesses the relevance of a candidate rule $r \leftarrow \mathbf{r}_b$ by considering not only the rule itself but also the rich structural context of the query's head entity, h. In order to calculate a context-aware score for a relation r, we design a novel strategy to generate instance that are enriched by context-aware rules and to train a model.

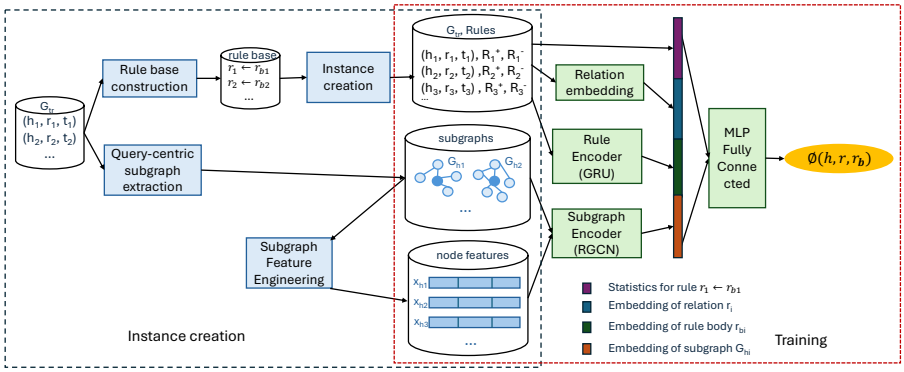


Figure 1: Steps to generate instances enriched by context aware rules and to train SLogic model

Figure 1 shows the major steps to create the rule enriched instances for a knowledge graph $\mathcal{G}_{tr}$ and to train the SLogic model. Sections 4.1 and 4.2 presents these components in detail.

4.1 GENERATION OF INSTANCES ENRICHED BY CONTEXT AWARE RULES

One novel component of this work is to utilize the context-aware rules. We do this by constructing instances that are enriched by context-aware rules.

4.1.1 RULE BASE CONSTRUCTION

The initial step of our method is to mine a comprehensive set of logical rules from the training knowledge graph $\mathcal{G}$. To extract these rules, we iterate through each ground truth triple (h, r_h, t) in the training set. For each triple, we treat it as a positive example of a rule instantiation where r_h is the head relation. We then perform a Depth-First Search (DFS) to find all simple paths connecting the head entity h and the tail entity t within the graph, up to the predefined maximum length L. Each discovered path is converted into its corresponding relational path, which forms a candidate rule body $\mathbf{r}_b$.

After enumerating all candidate rules, we quantify their quality using established statistical metrics. For each rule $r_h \leftarrow \mathbf{r}_b$, we compute its *standard confidence*, defined as the conditional probability $P(r_h|\mathbf{r}_b)$,

$$\text{Confidence}(r_h \leftarrow \mathbf{r}_b) = \frac{\#(\mathbf{r}_b, r_h)}{\#(\mathbf{r}_b)}$$

where $\#(\mathbf{r}_b, r_h)$ is the count of entity pairs connected by both the body path $\mathbf{r}_b$ and the head relation r_h, and $\#(\mathbf{r}_b)$ is the total count of entity pairs connected by the body path.

While standard confidence is widely used, it can be unreliable for rules where the body count $\#(\mathbf{r}_b)$ is low. To address this, we further compute the *Wilson score interval's lower bound* (Wilson, 1927)

for each rule's confidence. This provides a more robust and conservative estimate of a rule's reliability, particularly for less frequent patterns. This score is calculated as

$$\text{Wilson}(p, n, z) = \frac{1}{1 + \frac{z^2}{n}} \left(p + \frac{z^2}{2n} - z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}} \right)$$

where p is the observed standard confidence, n is the body count $\#(\mathbf{r}_b)$, and z is the quantile of the standard normal distribution (typically 1.96 for a 95% confidence interval).

This mining process results in a static global rule base where each rule is associated with a confidence and a Wilson score. This base forms the symbolic foundation that SLogic later enriches with query-specific, contextual information.

4.1.2 Query-Centric Subgraph Extraction

To provide a localized, computationally tractable context for each query, we perform an offline preprocessing step to extract a unique subgraph for every entity in the knowledge graph. This process yields a collection of graph structures that serve as the primary input to the subgraph encoder in SLogic (Figure 1).

To extract subgraphs for each entity $h \in \mathcal{E}$, we extract its local neighborhood by initiating a k-hop Breadth-First Search (BFS) in $\mathcal{G}$. This traversal expands outwards from the central entity h, exploring both its incoming and outgoing connections to capture a rich, bidirectional context. To maintain a uniform structure and mitigate the computational challenges posed by high-degree "hub" nodes, we employ neighbor sampling at each hop of the BFS. Specifically, if an entity has more neighbors than a predefined threshold (α), we randomly pick α neighbors to continue the traversal. This results in a subgraph $\mathcal{G}_h \subset \mathcal{G}$ for entity h, which captures h's most relevant local structure.

Feature engineering. A raw subgraph is insufficient for a GNN to understand the relative importance of its components. We therefore enrich the subgraph by engineering a set of node features, $\mathbf{x}_v$, for each node $v \in \mathcal{G}_h$. These features are designed to encode the structural role of each node relative to the central query entity h. The feature vector $\mathbf{x}_v$ is a concatenation of three components. (i) *Head Entity Indicator:* A two-dimensional binary vector that identifies whether a node is the query head entity h (i.e., $[1, 0]$) or a neighbor (i.e., $[0, 1]$). This anchors the subgraph representation around the query. (ii) *Shortest Path Distance:* The geodesic distance (i.e., minimum number of hops) from node v to the head entity h within the subgraph $\mathcal{G}_h$. This explicitly encodes the structural proximity of each node to the query's origin. (iii) *Global Centrality Score:* The log-scaled degree of node v as calculated from the complete graph $\mathcal{G}$. This feature injects a measure of the node's global importance into the local subgraph context.

These features are purely topological and entirely independent of any node-specific content or identifiers (e.g., entity IDs or pre-trained embeddings). *This design choice enforces the inductive capability of our model*, allowing it to generate meaningful representations for entities and subgraphs not encountered during training.

4.1.3 Instance creation and Negative Sampling

To build the training set from a knowledge graph $\mathcal{G}_{tr}$, for each ground truth triple $(h, r, t) \in \mathcal{G}_{tr}$, we first sample up to k_{pos} unique positive rule bodies that correctly derive t. Then, for each of these positive rules, we pair it with k_{neg} hard negative rules selected using our newly designed sampling strategy described below. This creates a rich set of up to $k_{pos} \times k_{neg}$ training pairs for each original fact (h, r, t). The parameters k_{pos} and k_{neg} are important hyperparameters that control the data diversity and the positive-to-negative ratio in our training objective. Their impact on model performance is thoroughly examined in our analysis (Section 5.2).

For each fact, besides providing the positive rules, we also generate hard negative rules motivated by the use of negative samples in their training process Mikolov et al. (2013). The use of such hard negative rules is to distinguish between correct and incorrect patterns and improves generalization. The selection process of hard negative rules involves two main stages. First, for a given query $(h, r, ?)$, we form a candidate pool by identifying all locally applicable rules for relation r that can be successfully grounded from h, ranking them by their static Wilson score, and selecting the top-K.

Next, from this pool, we remove the "positive" rules that lead to the true answer t and then randomly sample k_{neg} rules from the remaining set to serve as our hard negatives.

For a query (h_i, r_i, t_i), let $\mathbf{R}_i^+$ and $\mathbf{R}_i^-$ consist of the k_{pos} positive rules $\{r_i \leftarrow r_{b_{i1}}^+, \cdots, r_i \leftarrow r_{b_{ik_{pos}}}^+\}$ and k_{neg} hard negative rules $\{r_i \leftarrow r_{b_{i1}}^-, \cdots, r_i^- \leftarrow r_{b_{ik_{neg}}}^-\}$ respectively. The instances generated in this step, subsequently provided to our model, comprise rule-enriched triplets, $((h_1, r_1, t_1), \mathbf{R}_1^+, \mathbf{R}_i^-)$, $((h_2, r_2, t_2), \mathbf{R}_2^+, \mathbf{R}_2^-)$, etc.

4.2 Model Architecture, Loss function, and Training

The SLogic framework is composed of two primary neural encoders ((see training box of Figure 1).) The first one is a *subgraph encoder*, which uses a Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2018) to process the query-centric subgraph $\mathcal{G}_h$. The second one is a *rule encoder*, which employs a Gated Recurrent Unit (GRU) (Cho et al., 2014) to encode the sequential rule body $\mathbf{r}_b$.

The embeddings from these two encoders, along with an embedding of the query relation r and the pre-computed static features of the rule (e.g., confidence, support), are concatenated and passed through a final Multi-Layer Perceptron (MLP) to yield the query-specific rule score $\phi(h, r, \mathbf{r}_b)$.

We train SLogic using a learning-to-rank framework. The objective is to assign a higher score to a "positive" rule that correctly entails a known fact than to a "negative" rule that does not. We employ a margin-based ranking loss for each training pair.

$$\mathcal{L} = \max(0, \epsilon - (\phi(h, r, \mathbf{r}_b^+) - \phi(h, r, \mathbf{r}_b^-))) \tag{1}$$

where ϵ is a predefined margin hyperparameter, $\mathbf{r}_b^+$ is the body of a positive rule, and $\mathbf{r}_b^-$ is the body of a hard negative rule.

The model learns a scoring function $\phi(\cdot)$ for each rule $r \leftarrow \mathbf{r}_b$ w.r.t. a query $(h, r, ?)$ (i.e., $\phi(h, r, \mathbf{r}_b)$). This strategy forces the model to leverage the contextual information in the query-specific subgraph $\mathcal{G}_h$ to distinguish between globally plausible rules and those that are truly relevant to a possible query $(h, r, ?)$.

4.3 Inference and Final Ranking

At inference time, our goal is to answer a query $\mathbf{q} = (h, r, ?)$ by leveraging our trained SLogic model to find the most plausible reasoning paths. Our method operates as a re-ranking framework, first identifying a set of high-quality candidate rules and then using SLogic to score them based on the specific query context. This is followed by a mathematically-grounded aggregation step to produce the final answer ranking.

Input: query $\mathbf{q}(h, r, ?)$, model $\phi(\cdot)$, hyperparameters

Output: a vector of scores that measure the plausibility of different entities answering $\mathbf{q}$

1. Candidate rule generation from global rule base
2. For each rule $r \leftarrow \mathbf{r}_b$, predict $\phi(h, r, \mathbf{r}_b)$ from model $\phi(\cdot)$
3. Update each rule's score ϕ to a confidence score w_i using softmax with temperature
4. For all the entities, calculate their potential to answer $\mathbf{q}$
   (a) Calculate a rule grounding score $ground(h, \mathbf{r}_b)$
   (b) Apply a binarization function to the grounding score $B(ground(h, \mathbf{r}_b))$
   (c) Compute $\mathbf{v}_{\text{ans}}$ using equation 2, which represents the entities potential to answer $\mathbf{q}$
5. Return $\mathbf{v}_{\text{ans}}$

Figure 2: Algorithm to answer a query $\mathbf{q}(h, r, ?)$

Figure 2 shows the steps to answer the given query.

Step 1 generates candidate rules. For any given query $(h, r, ?)$, exhaustively scoring all rules in the global rule base would be computationally prohibitive. To obtain a manageable yet high-quality candidate set, we first identify all rules that are locally applicable to h, and then select the top-N among them based on their Wilson score.

Step 2 calculates a context-aware (localized) importance score for each candidate rule. In particular, each candidate rule $\mathbf{r}_{b_i}$ is passed to our trained scoring function $\phi(h, r, \mathbf{r}_{b_i})$, which leverages the query's unique subgraph context $\mathcal{G}_h$. This step assigns a dynamic, context-aware score to each rule, moving beyond static metrics to assess its true relevance for the specific query. Next, the context-aware scores are converted into a confidence distribution using a temperature-controlled softmax (Step 3). The weight w_i for each rule is calculated as $w_i = \frac{\exp(\phi'_i/T)}{\sum_j \exp(\phi'_j/T)}$. The temperature T is an important hyperparameter. As $T \to 0$, the softmax approximates an argmax function, concentrating all weight onto the single highest-scoring rule. This allows the model to rely solely on the *strongest contextual signal*.

Utilizing the rules and their context-aware score, Step 4 calculates the potential of each entity to answer the given query. If the score of an entity e is high, it means that it is very probably that (h, r, e) is a valid fact. The potential is measured using a score that aggregates the grounding score and the rules' local importance score. Each rule is grounded to produce an answer vector (Steps 4a). A naive grounding counts the number of paths from h to other entities that can be reached by applying the rule $r \leftarrow \mathbf{r}_{b_i}$. Let $\mathbf{g}_i = \text{ground}(h, \mathbf{r}_{b_i})$. The value $\mathbf{g}_i[j]$ is the number of paths from h to an entity e_j when applying the rule $r \leftarrow \mathbf{r}_{b_i}$. If no such path exists, $\mathbf{g}_i[j] = 0$. This grounding operation can be efficiently conducted through sparse matrix multiplication.

A naive grounding score, can be a noisy signal dominated by high-degree nodes. To mitigate this effect, we squash the path counts using the `tanh` function, controlled by a tanh scale hyperparameter τ, $B(\mathbf{g}_i) = \tanh(\mathbf{g}_i/\tau)$. This provides a soft, saturated count that is controlled by the tanh scale hyperparameter τ, preventing an unbounded influence from numerous paths.

Finally (Step 4c), a vector $\mathbf{v}_{\text{ans}}$ that captures all the entities' plausibility to answer the given query is computed as the weighted sum of all the binarized, grounded rule vectors as in equation 2.

$$\mathbf{v}_{\text{ans}} = \sum_i w_i \cdot B(\mathbf{g}_i) \qquad (2)$$

From the values in $\mathbf{v}_{\text{ans}}$, we can rank all the entities. The rank of the true tail entity t is then used to compute the MRR and Hits@k.

5 Experiments: SLogic's performance in KGC

Datasets. We conduct our experiments on widely used benchmark datasets WN18RR (Dettmers et al., 2017), FB15k-237 (Toutanova & Chen, 2015), YAGO3-10 (Suchanek et al., 2007). For all datasets, we augment the data by adding inverse triplets (t, r^{-1}, h) for each original triplet (h, r, t), a common practice in the literature. The details about the datasets are described in Appendix A.

Baselines. Slogic is compared with 5 non-rule based approaches including TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), ConvE (Dettmers et al., 2018), ComplEx (Trouillon et al., 2016), and RotateE (Sun et al., 2019), and 6 rule based approaches including AMIE (Galárraga et al., 2013), Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), RNNLogic (Qu et al., 2021), RLogic(Cheng et al., 2022), and NCRL Cheng et al. (2023).

Evaluation metrics. We evaluate our model using Mean Reciprocal Rank (MRR), and Hits@k (k=1, 10) under the standard filtered setting (Bordes et al., 2013). For each test triple (h, r, t), we evaluate by predicting t for the forward query $(h, r, ?)$ and h for the inverse query $(t, r^{-1}, ?)$. Rule-based systems frequently produce tied scores for many entities, which can lead to misleadingly optimistic ranks; for instance, a naive approach would assign a rank of 1 if no rules apply and all entities receive a score of zero. To robustly handle all such ties, we adopt the expected rank strategy from Qu et al. (2021) for tie-breaking, where the rank is calculated as $m + (n+1)/2$; here, m is the number of entities with a strictly higher score than the correct answer, and n is the number of other entities sharing the same score. For queries with head entities not seen during training, we fall back to ranking answers based on the tail entity frequency for the given relation.

Default setting of hyperparameters. For rule base construction, the length of rule body (or the depth of the DFS) L is set to 5 (for WN18RR) and 3 (for both FB15K-237 and YAGO3-10). In subgraph extraction, the number of subgraph hops k is set to be 4 for WN18RR dataset and 1 for both FB15K-237 and YAGO3-10 datasets. The hyperparameter α is 100 for all the datasets. For model training, the margin ϵ is set to be 1, k_{pos} and k_{neg} are set to be 5 and 20 respectively for all the three datasets. During inference stage, we set the default parameters as $N = 50$ for WN18RR and FB15k-237, and $N = 10$ for YAGO3-10. For all datasets, we use $T = 0.5$ and $\tau = 2.0$.

Further details on the experimental setting, including hardware configuration, implementation specifics, and training procedures, are provided in Appendix B.

5.1 Comparisons with baseline methods

Table 1 shows the effectiveness of SLogic when comparing with the baselines. It shows that SLogic outperforms other methods on WN18RR and YAGO3-10, but performs less effectively on FB15K-237. This is because WN18RR contains a larger number of distinct relations, resulting in relatively low rule support and making it harder to distinguish among the rules. The numbers for other systems are taken from Cheng et al. (2022; 2023). Since NCRL† employs a different evaluation protocol, where the rank is computed as $m+1$, compared to the strategy used for all other methods. To ensure a fair comparison, we reran NCRL using our evaluation metric and reported these results as NCRL*.

	Models	WN18RR			FB15K-237			YAGO3-10		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
None-rule based	TransE	0.23	2.2	52.4	0.29	18.9	46.5	0.36	25.1	58.0
	DistMult	0.42	38.2	50.7	0.22	13.6	38.8	0.34	24.3	53.3
	ConvE	0.43	40.1	52.5	**0.32**	21.6	50.1	0.44	35.5	61.6
	ComplEx	0.44	41.0	51.2	0.24	15.8	42.8	0.34	24.8	54.9
	RotatE	0.47	42.9	55.7	**0.32**	**22.8**	**52.1**	0.49	40.2	**67.0**
Rule-based Learning	AMIE	0.36	39.1	48.5	0.23	14.8	41.9	0.25	20.6	34.3
	Neural-LP	0.38	36.8	40.8	0.24	17.3	36.2	-	-	-
	DRUM	0.38	36.9	41.0	0.23	17.4	36.4	-	-	-
	RNNLogic*	0.46	41.4	53.1	0.29	<u>20.8</u>	44.5	0.34	24.2	52.5
	RLogic	0.47	44.3	53.7	<u>0.31</u>	20.3	<u>50.1</u>	0.36	25.2	50.4
	NCRL†	0.67	56.3	85.0	0.30	20.9	47.3	0.38	27.4	53.6
	NCRL*	0.27	22.6	33.9	0.17	9	32.9	0.14	4	33.5
	SLogic	**<u>0.49</u>**	**<u>44.7</u>**	**<u>55.8</u>**	0.28	20.5	42.2	**<u>0.50</u>**	**<u>42.8</u>**	<u>63.4</u>

Table 1: SLogic vs. Baselines in KG completion task (**Bold**/<u>Underlined</u> numbers: best among all methods/best among all rule learning methods; '–': could not be run on our machine)

5.2 Sensitivity analysis and ablation study

This section analyzes the sensitivity of our model to several key hyperparameters and presents an ablation study by removing different components during the inference stage.

Effect of positive and negative sampling ratio. This analysis evaluates our model's sensitivity to the two parameters, k_{pos} and k_{neg}. As shown in Table 2, the results demonstrate that the model is highly robust to different parameter settings. Performance remains stable within a narrow margin across all tested configurations on all the datasets, with the configuration $k_{\text{pos}} = 5$ and $k_{\text{neg}} = 20$ consistently ranking among the best. The key practical implication of this robustness is that an exhaustive and computationally expensive search for optimal values is unnecessary. This is particularly valuable for larger datasets such as FB15K-237 and YAGO3-10, where memory constraints make more resource-intensive configurations infeasible.

Effect of # of rules per query and subgraph hops. Figure 3 reveals that optimal hyperparameters are highly dependent on the knowledge graph's structure. As shown in (a), the ideal number of inference rules varies significantly: performance on WN18RR peaks with a moderate 50-70 rules, while YAGO3-10 performs best with the fewest (10), and FB15k-237 shows more volatile behavior. This suggests that for large graphs like YAGO3-10, the top-ranked rules are highly reliable and adding more only introduces noise.

	WN18RR			FB15k-237			YAGO3-10		
k_{pos}	$k_{\text{neg}} = 10$	$k_{\text{neg}} = 20$	$k_{\text{neg}} = 40$	$k_{\text{neg}} = 10$	$k_{\text{neg}} = 20$	$k_{\text{neg}} = 40$	$k_{\text{neg}} = 10$	$k_{\text{neg}} = 20$	$k_{\text{neg}} = 40$
1	0.4849	0.4827	0.4845	0.2760	0.2758	0.2763	0.4865	0.4835	0.4975
5	0.4852	0.4871	0.4841	0.2782	0.2796	-	0.4986	0.5031	0.5011
10	0.4856	0.4880	0.4850	0.2787	0.2787	-	0.5036	0.5031	-

Table 2: Sensitivity test on the effect of sampling ratio (k_{pos} vs. k_{neg}) on MRR. Darker cells indicate higher MRR. Certain configurations for FB15k-237 and YAGO3-10 (marked with -) were omitted due to excessive memory requirements.

Figure 3(b) shows that a larger local context (more hops) brings more benefit on WN18RR although the performance increase is not obvious when having more than 2 hopes. We cannot access a larger local context using more than 1-hop for the much denser graphs FB15k-237 and YAGO3-10.

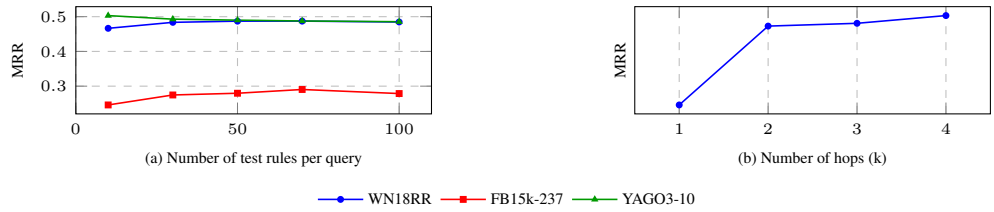


Figure 3: Sensitivity analysis of MRR across three datasets. The optimal hyperparameters and the feasibility of using deep subgraph contexts vary significantly with KG structure. Higher number of hops could not be run for FB15k-237 and YAGO3-10 on our machine.)

Ablation study and effect of inference hyperparameters. Figure 4 shows the effect of including/excluding the two components of (1) normalizing confidence score and (2) binarization of ground score. Including the normalization score calculation consistently improves the performance (from ✗ to low softmax temperature ($T = 0.5$). When this component is included, the performance is not sensitive to the value of T. Regarding the binarization component, including it (✗) helps with YAGO3-10, but not the other two datasets.

5.3 Case Study: The Impact of Local Context on Rule Scoring

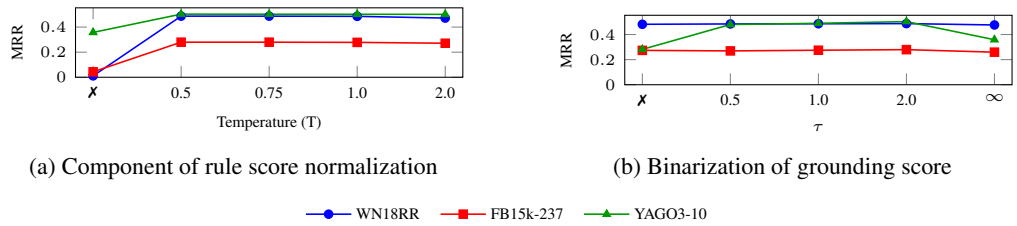
To demonstrate our model's ability to perform context-aware reasoning, we present a case study to compare the scoring of two rules on the YAGO3-10 dataset. We identified two distinct queries for the relation `isLocatedIn`, which is shortened to be `isL`. The rules and their global Wilson scores are held constant.
Rule A: `isL(X,Y):-isL(X,Z1),participatedIn(Z1,Z2),isL(Z2,Y)` (W-Score: 0.8545)
Rule B: `isL(X,Y):-isL(X,Z1),hasCapital(Z1,Z2),isL(Z2,Y)` (W-Score: 0.9578)

For the two queries (see Table 3), the different contexts come from the different head entities and their corresponding 1-hop subgraph.

Table 3: A case study showing SLogic's contextual scoring capability.

	Case 1	**Case 2**
Query	`(Old_Shatterhand_(film),isL,?)`	`(U. of Alaska System,isL,?)`
Subgraph Context	Dominated by **film-industry** entities, e.g., actors (`Lex Barker`) and directors (`Hugo Fregonese`).	Dominated by **geographical** entities, e.g., cities (`Fairbanks, Alaska`) and countries (`United States`).
Model's Score for A	**-0.3528**	-1.6444
Model's Score for B	-2.4848	**0.0251**

As shown in Table 3, our model's final preference between these two rules flips depending on the local subgraph encoder in SLogic. Query 1 comes from the film industry, this context aligns better with the event-based rule (Rule A). Thus, our model after learning from the query-specific subgraphs



(a) Component of rule score normalization

(b) Binarization of grounding score

Figure 4: Ablation study of inference components and effect of inference hyperparameters. The ✗ symbol denotes that the component is disabled.

gives it a higher local score (-0.3528) than Rule B despite Rule B's global score is higher. Query 2 on the other hand is more related to geographical and institutional information. Thus, Rule B receives a higher local score (0.0251) than Rule A.

5.4 Efficiency Analysis

For the rule-based method, we report the time required for rule collection across the different methods as well as the the training time to obtain the performance shown in Table 1. Inference time is excluded from the comparison, since all methods employ comparable inference procedures.

Method	Sub-component time	WN18RR	FB15k-237	YAGO3-10
SLogic	Mining time	0.25	54	56
	Negative sampling time	149	189	826
	Training time	552	892	291
	Total time	**701**	**1135**	**1173**
RNNLogic	Mining time	88	279	87
	Training time	332	252	491
	Total time	**420**	**513**	**578**
DRUM	**End to end training**	**55**	**717**	-
NCRL	**End to end training**	**6**	**126**	**336**

Table 4: Running time (in minutes) across different methods and datasets ('–': Out of memory)

Table 4 shows that SLogic requires more training time than the baseline methods for all the datasets. This overhead arises primarily because generating positive–negative sample pairs substantially increases the number of training instances. Under the default setting of $k_{\text{pos}} = 5$ and $k_{\text{neg}} = 20$, the resulting number of instances (triplets associated with positive-negative rule pairs) is approximately 100 times larger than the number of triplets in the original graph. For YAGO3-10 dataset, to improve efficiency, only 10% of the graph is used to extract the positive rule base and generate training instances, keeping the training time low. In contrast, identifying negative samples is more time-consuming, as this step requires access to the entire graph.

6 Conclusion

We addressed the problem of knowledge graph completion with a novel rule-based framework. Unlike existing approaches that rely on globally fixed rule confidences, our method leverages the query context, represented as a subgraph of the head entity, to dynamically recalculate rule importance. This context-aware mechanism enables more accurate reasoning by emphasizing rules that are highly relevant to the specific query. Experimental results demonstrate that our approach consistently outperforms other rule-based baselines, as well as embedding-based methods, particularly on knowledge graphs with fewer distinct relations. Beyond improving accuracy, our method also provides interpretable predictions by explicitly grounding query-specific rules.

Reproducibility Statement

To facilitate reproducibility, we provide an anonymized code repository at: https://anonymous.4open.science/r/slogic-81FE/. The repository contains the datasets used in our experiments, as well as scripts for data preprocessing, model training, and evaluation. It also includes detailed instructions for reproducing all main results reported in the paper.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. Rlogic: Recursive logical rule learning from knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 179–189, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539421. URL https://doi.org/10.1145/3534678.3539421.

Kewei Cheng, Nesreen K. Ahmed, and Yizhou Sun. Neural compositional rule learning for knowledge graph reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=F8VKQyDgRVj.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

William W. Cohen, Fan Yang, and Kathryn Mazaitis. Tensorlog: Deep learning meets probabilistic dbs. *CoRR*, abs/1707.05390, 2017. URL http://arxiv.org/abs/1707.05390.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476, 2017. URL http://arxiv.org/abs/1707.01476.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with pytorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

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

Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp. A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context. In *ESWC*, pp. 179–193, 2019. URL https://doi.org/10.1007/978-3-030-21348-0_12.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. URL https://api.semanticscholar.org/CorpusID:2785507.

Yunshi Lan and Jing Jiang. Query graph generation for answering multi-hop complex questions from knowledge bases. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 969–974, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.91. URL https://aclanthology.org/2020.acl-main.91/.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3111–3119, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. {RNNL}ogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tGZu6DlbreV.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/0c72cb7ee1512f800abe27823a792d03-Paper.pdf.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (eds.), *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pp. 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. URL https://doi.org/10.1007/978-3-319-93417-4_38.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 697–706, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242667. URL https://doi.org/10.1145/1242572.1242667.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HkgEQnRqYQ.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4007. URL https://aclanthology.org/W15-4007.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2071–2080. JMLR.org, 2016.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. ISSN 01621459, 1537274X. URL http://www.jstor.org/stable/2276774.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2014. URL https://api.semanticscholar.org/CorpusID:2768038.

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/0e55666a4ad822e0e34299df3591d979-Paper.pdf.

Yongqi Zhang and Quanming Yao. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 912–924, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512008. URL https://doi.org/10.1145/3485447.3512008.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: a general graph neural network framework for link prediction. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Zhaocheng Zhu, Xinyu Yuan, Mikhail Galkin, Sophie Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. A*net: a scalable path-based reasoning approach for knowledge graphs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

A DATASETS

This work utilizes three widely-used benchmark datasets for knowledge graph completion: WN18RR, FB15k-237 and YAGO3-10. These datasets are standard in the field as they have been curated to prevent test triple leakage from the training set. To support rule learning methodologies, we preprocess each knowledge graph by adding inverse triplets. The statistics of datasets are summarised in Table 5.

- WN18RR, introduced in Dettmers et al. (2017), a subset of the WN18 dataset, which is designed to be an intuitive dictionary and thesaurus for natural language processing tasks. In WN18RR, entities represent word senses and relations define the lexical connections between them.
- FB15k-237, introduced in Toutanova & Chen (2015), a frequently used benchmark dataset derived from Freebase. It is a large, online collection of structured data from various sources, including user-contributed wiki data.
- YAGO3-10, introduced in Suchanek et al. (2007), a subset of the large-scale semantic knowledge base YAGO3. It is constructed by integrating information from multiple authoritative sources, including Wikipedia, WordNet, and GeoNames.

Dataset	#Entities	#Relations	#Train	#Validation	#Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Table 5: Dataset statistics

B EXPERIMENTAL SETUP

B.1 HARDWARE

All experiments were conducted on a Dell PowerEdge R7525 server. This machine is equipped with 512 GiB of system RAM and powered by two AMD EPYC 7313 CPUs, providing a total of 32 physical cores (64 threads) running at a base clock speed of 3.0 GHz. For model training and inference, we utilized a single NVIDIA A100 GPU with 80 GiB of VRAM.

B.2 IMPLEMENTATION DETAILS

Our model and training pipeline are implemented using PyTorch and PyTorch Geometric PyTorch Geometric (PyG) library (version 2.5.2) (Fey & Lenssen, 2019). Below, we provide detailed descriptions of the model architecture, data handling procedures, and training configuration.

Model Architecture. The SLogic model is a hybrid neural network composed of three main parts:

1. A **relation embedding layer** (`torch.nn.Embedding`) that provides dense vector representations for all relations in the knowledge graph. A designated padding index is used to handle variable-length rule bodies.
2. The **subgraph encoder** is a stack of Relational Graph Convolutional Network (`RGCNConv`) layers. We use 1 GNN layer for both FB15k-237 and YAGO3-10 dataset and 2 GNN layers for WN18RR, each followed by a ReLU activation and a dropout layer (p=0.5). The encoder takes the node feature matrix and the subgraph's edge information as input and produces final node embeddings. We extract two outputs: the embedding of the head node itself and a graph-level embedding computed via global mean pooling.
3. The **rule encoder** is a single-layer Gated Recurrent Unit (`torch.nn.GRU`) that processes the sequence of relation embeddings corresponding to a rule body. The final hidden state of the GRU is used as the rule's semantic embedding.

These components are integrated by a final scoring MLP. The feature vector for the MLP is a concatenation of: (1) the head node embedding from the GNN, (2) the graph-level embedding from the GNN, (3) the query relation embedding, (4) the rule body embedding from the GRU, and (5) a 4-dimensional vector of the rule's static statistics (support, confidence, Laplace confidence, and Wilson score). This combined vector is passed through a two-layer MLP with a ReLU activation and dropout to produce the final scalar score.

Data handling and leakage prevention. We use a custom PyTorch Geometric `Dataset` class to load the pre-computed subgraphs and training metadata. A critical aspect of our data loading process is the prevention of data leakage. During training, for a given triple (h, r, t), the GNN must not have access to the direct edge (h, r, t) in the subgraph $\mathcal{G}_h$, as this would allow it to solve the task trivially.

To prevent this, our `Dataset` class, during the loading of each individual sample, dynamically removes both the target edge (h, r, t) and its corresponding inverse edge (t, r^{-1}, h) from the subgraph's `edge_index` and `edge_attr` tensors before the subgraph is passed to the model. This ensures that the GNN must rely on the broader structural context rather than a simple edge-detection shortcut.

Training details. The model is trained end-to-end by minimizing a margin-based ranking loss (`torch.nn.MarginRankingLoss`) with a margin of $\epsilon = 1.0$. We use the Adam optimizer Kingma & Ba (2014) with a learning rate of 0.001. For each training step, a batch of positive and negative rule pairs is processed. The rule bodies, which are sequences of relation IDs of variable length, are left-padded to the maximum length in the batch using our designated padding index. The model is trained for 5 epochs for all the datasets. The embedding dimensionality for both the relations and the RGCN layers was set to 128.

C USE OF LARGE LANGUAGE MODELS (LLMS)

ChatGPT and Gemini were used to correct grammatical errors and enhance the clarity of the writing.