Prompt-fused framework for Inductive Logical Query Answering

Zezhong Xu[♣♡]**, Peng Ye**[♣]**, Lei Liang**[◇]**, Huajun Chen**[♣♡♠]**, Wen Zhang**[♣♡∗]

[♣]Zhejiang University, [◇]Ant Group
[♠]Alibaba-Zhejiang University Joint Institute of Frontier Technology
[♡]Zhejiang University-Ant Group Joint Laboratory of Knowledge Graph
{xuzezhong, yep, huajunsir, zhang.wen}@zju.edu.cn
leywar.liang@antgroup.com

Abstract

Answering logical queries on knowledge graphs (KG) poses a significant challenge for machine reasoning. The primary obstacle in this task stems from the inherent incompleteness of KGs. Existing research has predominantly focused on addressing the issue of missing edges in KGs, thereby neglecting another aspect of incompleteness: the emergence of new entities. Furthermore, most of the existing methods tend to reason over each logical operator separately, rather than comprehensively analyzing the query as a whole during the reasoning process. In this paper, we propose a query-aware prompt-fused framework named Pro-QE, which could incorporate existing query embedding methods and address the embedding of emerging entities through contextual information aggregation. Additionally, a query prompt, which is generated by encoding the symbolic query, is introduced to gather information relevant to the query from a holistic perspective. To evaluate the efficacy of our model in the inductive setting, we introduce two new challenging benchmarks. Experimental results demonstrate that our model successfully handles the issue of unseen entities in logical queries. Furthermore, the ablation study confirms the efficacy of the aggregator and prompt components. The code is available at *https://github.com/yep96/Pro_QE*.

1. Introduction

Knowledge graphs (KGs) have garnered interest in recent years. Their applications span natural language processing, information retrieval, and data analytics. The task of addressing logical queries on KGs encompasses multiple entities, relations, and logical operators. For instance in Figure 1, *Who has won the Turing Award in developing countries?* could be translated into a logical query that requires traversing the KG to find the answer. In this query, the anchor nodes are the *Turing Award* and *Developing Countries*, while the target answer, such as *Qizhi Yao*, represents the desired response.

Nonetheless, the existence of noise and incompleteness within KGs often leads to missing edges in the path from the anchor to the answer nodes, thereby impeding the retrieval of all pertinent answers through sub-graph matching. To tackle this challenge, a promising approach is to embed the query into a vector space, which facilitates the utilization of meaningful entity embeddings to handle missing or erroneous edges. Typically, query embedding methods, such as GQE (Hamilton et al., 2018), Q2B (Ren et al., 2020), BetaE (Ren and Leskovec, 2020), FuzzyQE (Chen et al., 2021b), and GNN-QE (Zhu et al., 2022), involve breaking down the query into simpler sub-queries, reasoning over sub-queries step by step. Subsequently, a distance function is employed to select entities that closely resemble the query results as the answer

nodes.

Q: Who has won the Turing Award in developing countries?
$q = V_? . \exists V : LiveIn(V, \mathtt{DCs}) \wedge Win(\mathtt{Turing\_Award}, V_?)$
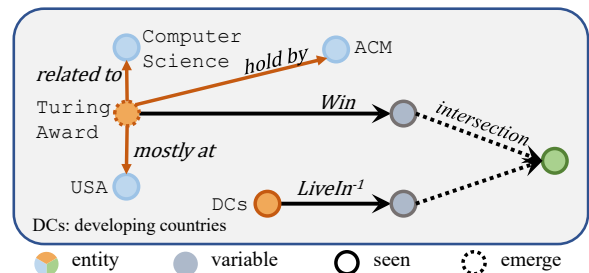

Figure 1: The example of an inductive logical query.

Despite the notable achievements of current query embedding methods in handling missing edges within KGs, there remain two unresolved issues. The first issue pertains to the inductive query scenario. Existing query embedding models are primarily designed to handle scenarios where edges are missing in KG. However, KGs are dynamic and constantly evolving, which means new entities and relations emerging frequently. For instance, in Figure 1, the anchor node *the Turing Award* is a new entity, so its representation is randomly initialized. We call this scenario the inductive setting, indicating queries may involve entities that were not present before. This task is challenging and necessitates advanced techniques and methodologies. The inductive GNN-QE model (Galkin et al., 2022) is currently the only model that considers inductive scenarios. However, it still overlooks the coherence

---

between the entire query and the inferred results, which leads us to the second unresolved issue.

The second issue concerns holistic query understanding in KGs. Complex KG queries involve multiple conditions and can exhibit intricate interdependencies. Most current models handle query reasoning by dividing queries into sub-queries. While these approach simplifies the problem and improves efficiency in some cases, they hinder the model's ability to establish connections between different constraints from a holistic perspective. For example, consider a query asking *which US presidents were lawyers in American history*. A common approach involves querying for each logical operation separately: first querying for *US presidents in history*, then for *individuals who were lawyers*, and finally intersecting the results. However, this approach may retrieve non-Americans when querying for lawyers, unnecessarily expanding the scope. Conversely, when querying for lawyers, emphasizing nationality information can quickly retrieve relevant results without redundant operations.

This article aims to explore methodologies for conducting complex query reasoning in an inductive scenario. Our primary objectives include **Contextual Awareness**: The model should have the ability to obtain the representation of newly emerged entities by leveraging the representations of surrounding known entities. **Query Awareness**: The model should comprehensively understand the query being posed and discern the role of each node within the query. **Relevance Awareness**: The model should recognize that identical known entities can assume different roles in different contexts during the aggregation process. For example, the attributes of an esports athlete and a basketball athlete are distinct, even though both are athletes. To address these objectives, we've developed a logical query-answering framework called *Pro-QE* considering the three aspects.

We devised two benchmarks under the inductive scenario and conduct the experiments to evaluate the superior performance of our model. Furthermore, the ablation study supports the effectiveness of different module designs. Our contributions can be summarized as follows:

- We emphasize the significance of the inductive setting in the context of logical queries, highlighting the need for models to effectively handle new emerging entities.

- We advocate for a holistic approach to logical query processing, going beyond the decomposition of queries into sub-queries and considering the overall impact of the query on each step of the reasoning process.

- We establish new benchmarks for logical query processing under the inductive setting, con-

ducting extensive experiments across various scenarios and achieving state-of-the-art results.

2. Related work

2.1. Logical Query Answering

Query answering in knowledge graphs has been widely studied in the literature. GQE(graph-query embedding) (Hamilton et al., 2018) first proposes to embed a query into a low-dimensional vector space and use the final embedding to find answers. Since GQE only supports intersection ($\wedge$) logical operator, Q2B (query to box) (Ren et al., 2020) replaces the vector embedding with hyperrectangles. To support the negation ($\neg$) operator, BetaE (Ren and Leskovec, 2020) is proposed which models the query and entities using beta distributions. CQD (Continuous Query Decomposition) (Arakelyan et al., 2021) applies beam search to an embedding model and could be trained with simple queries. ConE (Cone Embeddings) (Zhang et al., 2021) proposes a new geometry model that embeds entities and queries using Cartesian products of two-dimensional cones. BiQE (Bidirectional Query Embedding) (Kotnis et al., 2021) uses the transformer-based to incorporate the query context. TEMP (TypE-aware Message Passing) (Hu et al., 2022) introduces a type-aware model making use of type information to assist logical reasoning. FuzzQE (Chen et al., 2021b) satisfies the axiomatic system of fuzzy logic to reason. Q2P (Query2Particles) (Bai et al., 2022) encodes each query into multiple vectors. GNN-QE (Zhu et al., 2022) concentrates on the interpretation of the variables along the query path. Inductive GNN-QE (Galkin et al., 2022)(based on (Zhu et al., 2022)) propose using NBFNet (Zhu et al., 2021) as the one-hop(1p) projection to solve the inductive reasoning. ENeSy (Neural-Symbolic Entangled) (Xu et al., 2022b) uses the neural and neural reasoning results to enhance each other to alleviate the problem of cascading error.

2.2. Inductive Link Prediction

Rule-based methods focus on mining rules to provide us with an interpretable reasoning process and can be applied to inductive link prediction. AMIE (Galárraga et al., 2013) and AMIE+ (Galárraga et al., 2015) use predefined metrics such as confidence and use three operations, including dangling atom, instantiated atom, and closing atom that add different types of atoms to incomplete rules. Anyburl (Meilicke et al., 2019) proposes a framework that can mine rules in an efficient way. Rudik (Ortona et al., 2018) can mine positive and

negative rules, which can be used for link prediction tasks and other tasks. More recently, differentiable rule learning methods attracted more attention, which can determine the rule confidence and structure of the rule. Neural-LP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) based on Tensorlog (Cohen et al., 2020) generate the possibilities of different relationships in each step, and the symbolic rules can be parsed with the relations' weight. Neural-Num-LP (Wang et al., 2019b) extends Neural-LP to learn the numerical rules. Ruleformer (Xu et al., 2022a) utilizes context information to generate different rules according to a specific environment. Neural Logic Inductive Learning (Yang and Song, 2019) based-on transformer to mine non-chain-like rules which can extend the diversity rules. More recent works exploit Graph Neural Networks, which can be applied to graph structures via the existing neighbors. (Hamaguchi et al., 2017) compute the embeddings of OOKB entities, exploiting the limited auxiliary knowledge provided at test time. LAN (Logic Attention Network) (Wang et al., 2019a) aggregates neighbors with both rule-based attention and neural-based weights. INDIGO (Liu et al., 2021) accomplishes inductive knowledge graph completion based on a graph convolutional network with pair-wise encoding. Moreover, GraIL (Teru et al., 2020), CoMPILE (Mai et al., 2021), and TACT (Chen et al., 2021a) learn the ability of relation prediction by subgraph extraction and GNNs independent of any specific entities. MorsE (Chen et al., 2022) considers transferring universal, entity-independent meta-knowledge by meta-learning.

3. Problem Formulation

A KG can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{T})$, where $\mathcal{V}$ denotes a set of entities, $\mathcal{R}$ signifies a set of relations, and $\mathcal{T}$ represents a set of triplets. Each triplet $r(e_i, e_j)$ in $\mathcal{T}$ is defined as a relation $r \in \mathcal{R}$ existing between entities e_i and e_j from the set $\mathcal{V}$. This notation illustrates that the relation r holds between the entities e_i and e_j.

3.1. Logical Query Answering

As depicted in Figure 1, a query comprises constants (utilized as anchor nodes), variables (employed as intermediate nodes and answer nodes), relation projections $r(a, b)$, signifying binary functions over variables or constants, and logic operators ($\vee, \wedge$, and $\neg$). Utilizing its disjunctive norm form (Ren and Leskovec, 2020), the query can be represented as a disjunction of several conjunctions, as shown below:

$$q[V_?] := V_? . \exists V_1, V_2, \ldots, V_k : c_1 \vee c_2 \vee \cdots \vee c_n. \quad (1)$$

where c_i denotes either a single literal or a conjunction of multiple literals, i.e., $c_i = a_{i1} \wedge \cdots \wedge a_{im}$, where a_{ij} is an atom or the negation of an atom. The atoms can take the following forms: $r(e_a, V)$ or $\neg r(e_a, V)$ or $r(V', V)$ or $\neg r(V', V)$. Here, e_a represents one of the constants (as there might be several anchor nodes), and $V, V' \in V_1, V_2, \ldots, V_k, V_?$ are the variables, with the constraint that an atom should satisfy $V \neq V'$. The operations above are precisely defined as follows:

Relational Projection: Given an entity set $\mathcal{S} \subseteq \mathcal{V}$ and a relation $r \in \mathcal{R}$, the relational projection operator returns an entity set $\mathcal{S}'$ that includes entities connected to at least one of the entities in $\mathcal{S}$ via the relation r, i.e., $\mathcal{S}' = e' \in \mathcal{V} | \exists r(e, e'), e \in \mathcal{S}$.

Union: Given sets of entities $S_1, S_2, \ldots. S_n$, the union operator returns a new set $\mathcal{S}'$ that contains the entities appearing in any one of the sets, i.e., $\mathcal{S}' = \bigcup_{i=1}^{n} S_i$.

Intersection: Given sets of entities $S_1, S_2, \ldots, S_n$, the intersection operator returns a new set $\mathcal{S}'$ that contains the entities belonging to the intersection of these sets, i.e., $\mathcal{S}' = \bigcap_{i=1}^{n} S_i$.

Complement: Given a set of entities $\mathcal{S}$, this operator returns a new entity set $\mathcal{S}'$, which is the complement of $\mathcal{S}$, i.e., $\mathcal{S}' = \mathcal{V} - \mathcal{S}$.

3.2. Inductive Setting

The inductive setting, as explored in this paper, addresses scenarios in which there might be new entities that only appear in the validation and test sets. We encounter two distinct entity sets: $\mathcal{V}_{train}$ and $\mathcal{V}_{test}$. The former represents the entity set in the training set, while the latter represents all the entities in the test process. The inclusion of new entities in the test query introduces a unique set of challenges, notably the necessity to obtain representations of these new entities. Additionally, new triplets associated with emerging entities could result in an overlap between the training and test graphs. This overlap may lead to an increase in the number of observed answers to the same query in the training set, making it challenging to distinguish between novel and existing answers. We have classified the queries based on the location of new entities in the query: 1. New entities only appear in the anchor nodes. 2. New entities only appear in the answer set. 3. New entities appear in both the anchor nodes and the answer set. It is worth noting that there exists another type of query where new entities only appear in the reasoning process. Since the answer set can still be obtained using only the training data, we do not consider this type of query.
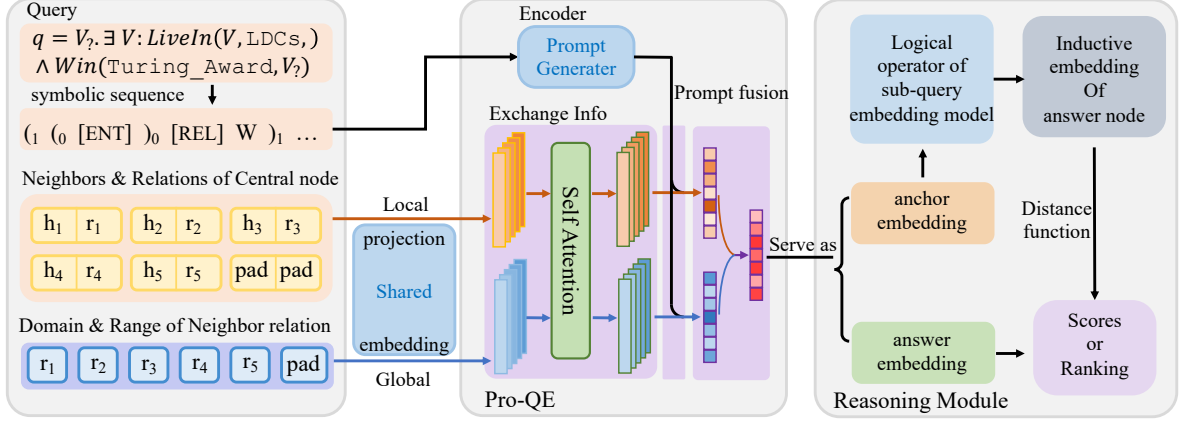
Figure 2: Framework of the proposed Pro-QE model.

4. Methodology

In this section, we introduce the comprehensive architecture of Pro-QE, which is depicted in Figure 2. Specifically, we leverage contextual information to obtain an appropriate representation of entities. During this process, we utilize neighbor nodes and relations to capture local information, and relation domains and ranges to capture global information. To enhance relevance awareness of different neighbors, we incorporate the attention mechanism. Additionally, we integrate the entire query information from a holistic perspective, which enables the model to better comprehend the overall query and its impact on each step of reasoning.

4.1. Inductive Representation Learning

To obtain a suitable representation of an entity, even in the case of it being an emerging entity, we employ a context-informed approach. During this process, we conscientiously consider both local and global information to derive a comprehensive representation. To ensure heightened relevance awareness, we leverage a self-attention mechanism for seamless information exchange.

Local Information. An entity's representation can be determined by its contextual information, *i.e.*, the subgraph in KG. We define an input embedding denoted as $\mathbf{e}^I$, which serves as input to derive representations of other entities connected to it. We could get the output embedding of the central entity using the input embedding of its neighbors, and the output embedding could be utilized for reasoning. In this paper, we consider the subgraph composed of the one-hop neighboring nodes.

Consider a central entity e_i with its one-hop neighboring entity set denoted as $\mathcal{N}_e(i)$ and relation set as $\mathcal{N}_r(i)$. For each entity $e_j \in \mathcal{N}_e(i)$ and relation $r_j \in \mathcal{N}_r(i)$ linking e_i and e_j, the input embedding $\mathbf{e}_j^I$ and relation embedding $\mathbf{r}_j$ are randomly initial-

ized. We apply a projection operator to e_j with relation r, to derive the projected representation $\mathbf{e}_j^P$. The projection could be any method that could make the link prediction. For instance, if we employ GQE (Hamilton et al., 2018) or FuzzQE (Chen et al., 2021b), the projection operator is applied in the following manner:

$$\mathbf{e}_j^P = P(\mathbf{e}_j^I) = \mathbf{e}_j^I + \mathbf{r_j} \qquad (2)$$

$$\mathbf{e}_j^P = P(\mathbf{e}_j^I) = \mathbf{g}(\mathbf{LN}(\mathbf{W}_{r_j}\mathbf{e}_j^I + \mathbf{b}_{r_j})) \qquad (3)$$

where the $\mathbf{g}$ is a mapping function, and $\mathbf{LN}$ is Layer Normalization function. $\mathbf{r_j}$ and $\mathbf{W}_{r_j}$ are both relation parameters according to the projection function, which could be seen as $\mathbf{r}_j$. After performing the projection on each entity-relation pair, we derive a set of representations, in which each vector partially reflects the information of the central entity.

Global Information. Since all of this information is exclusively derived from the subgraph, it may contain biases and noise. If the central entity is linked to a few-shot entity (not an emerging entity), the local representation derived from it may lack accuracy. Thus, it is necessary to explore more robust representations on a larger scale. In this paper, we consider the domain and range of relations, as entities which are related to the same domain or range are likely to exhibit similar features.

For each relation r, we define two entity sets based on the entities related to it: one set encompasses the entities serving as the head of the relation, signifying the domain of the relation. The other set contains the tail entity of the relation, representing the range of the relation. We initialize a representation $\mathbf{e}_{do(r)}$ and $\mathbf{e}_{ra(r)}$ for each such set to encapsulate all the entities within it. For a central entity and its related relation set, the corresponding embedding set $\mathbf{e}_{do(r_1)}...\mathbf{e}_{do(r_m)}, \mathbf{e}_{ra(r_1)}...\mathbf{e}_{ra(r_n)}$ could be perceived as reflecting information of e_i, where $m + n$ is the number of relations connected with e_i.
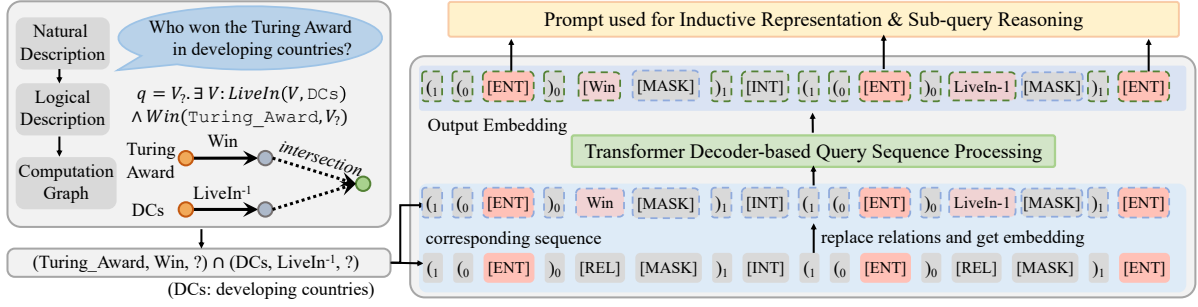
Figure 3: Architecture of the query prompt processing system.

Information Exchange and Aggregation. Now we have obtained an embedding set, $\mathbf{E}^L$ and $\mathbf{E}^G$, from both local and global perspectives, but have not established any associations between these embeddings. In reality, there exist direct or implicit connections between different neighbors. For example, two sports players may share a link to the same entity *sports player*, but their sport is *esports* and *basketball*, respectively. Consequently, even though both are associated with the *sports player* entity, their embeddings may vary when employed in subsequent tasks.

To facilitate information exchange between neighboring nodes, we utilize the attention module:

$$\mathbf{E}^{L\prime} = \text{Attn}(\mathbf{W}_q^L \mathbf{E}^L, \mathbf{W}_k^L \mathbf{E}^L, \mathbf{W}_v^L \mathbf{E}^L) \quad (4)$$

$$\mathbf{E}^{G\prime} = \text{Attn}(\mathbf{W}_q^G \mathbf{E}^G, \mathbf{W}_k^G \mathbf{E}^G, \mathbf{W}_v^G \mathbf{E}^G) \quad (5)$$

where the $\mathbf{W}_q^L$, $\mathbf{W}_k^L$ and $\mathbf{W}_v^L$ are the metrics used for modeling the Query, Key, and Value for the self-attention mechanism for Local embedding. $\mathbf{W}_q^G$, $\mathbf{W}_k^G$ and $\mathbf{W}_v^G$ are the same for Global information.

The Attn represents the self-attention with the Query, Key, and Value, which are denoted as Q, K, and V as input:

$$\text{Attn}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V \quad (6)$$

Here, Q, K, and V represent the Query, Key, and Value inputs for this attention layer.

However, we have used the context-aware and relevance-aware information, but lacking query-specific awareness. To address this limitation, we introduce a prompt mechanism.

4.2. Logical Query Prompt

As previously mentioned, in the sub-query model, each reasoning step lacks a comprehensive understanding of the overall query. We employ a method that transforms the entire query into a symbolic query statement and utilizes the decoder architecture of the transformer(Vaswani et al., 2017) to handle the query sequence.

The figure depicted in Figure 3 illustrates the comprehensive process of this step. We utilize the symbol [ENT] to represent anchor nodes and answer nodes, while [MASK] is employed to signify intermediate entities during the reasoning process. [r_id] is used to denote specific relations utilized in the projection operation, while [INT], [UNI], and [NEG] stand for intersection, union, and negation operators, respectively. Additionally, to preserve the order of different operations within a symbol sequence, we adopt [BCK_L] and [BCK_R] to represent brackets.

Next, to get the sequence, we begin with each [ENT] symbol representing an anchor node and enclose it within a pair of brackets [BCK_L0] and [BCK_R0]. If the subsequent operation involves projection or negation, we append the [r_id] or [NEG] operator, followed by a [MASK] symbol signifying an intermediate node in the computational graph. This procedure is repeated until an intersection or union operation is encountered, and then the elements requiring intersection or union are enclosed within higher-level brackets, i.e., [BCK_L1] and [BCK_R1] and so on. Finally, we add [MASK] at the end of the sequence to represent the operation itself and the result after the operation. If a union or intersection operation is required later, a pair of brackets representing the branch should be added outside the sequence representing that branch. Finally, an [ENT] token will be added at the end of the sequence to represent the answer node.

Capitalizing on the transformer model's exceptional ability to process sequence data, we exploit the transformer's decoder architecture to handle the symbol sequences generated through the aforementioned method. Specifically, we convert the symbol sequence into corresponding embeddings. It is crucial to note that each embedding of the [r_id] specific to a particular relation is distinct.

$$\mathbf{O} = \text{Decoder}(\text{Seq}(q)) \quad (7)$$

Here, Seq(q) signifies the symbol sequence generated based on query q, which is utilized as input to the decoder, and the resulting output sequence is embedded as $\mathbf{O}$.
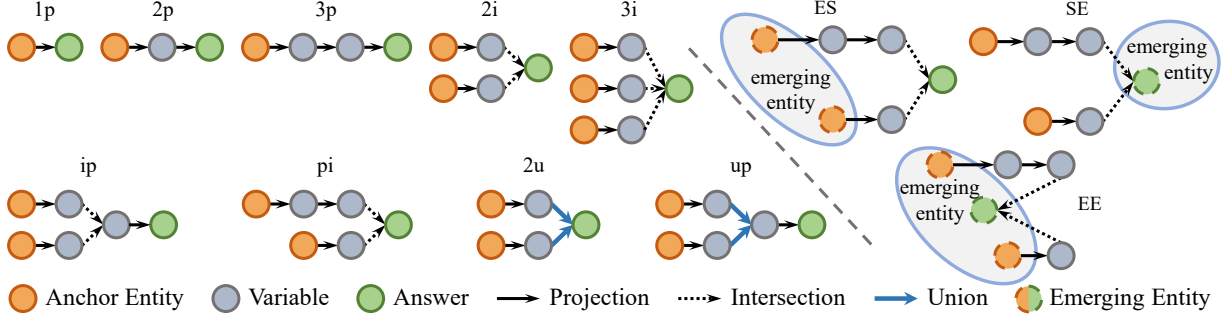
Figure 4: The query structures and query types used for grounding the specific queries on the KGs. The p, i, and u denote the projection, intersection, and union operator, respectively.

For each vector in $\mathbf{E}^{L\prime}$ and $\mathbf{E}^{G\prime}$, we compute the dot product with the prompt to establish the correlation between the embedding and the prompt. Subsequently, this correlation is employed as the weight coefficient during the aggregation of the anchor node's representation. For the local information:

$$\alpha_{ij} = \mathbf{p}_i \cdot \mathbf{e}'_j, \mathbf{e}'_j \in \mathbf{E}^{L\prime} \tag{8}$$

$$\mathbf{e}_i^L = \sum_{j=0}^{\mathcal{N}_e(i)} \alpha_{ij}\mathbf{e}'_j \tag{9}$$

For the global embedding, we use the same way to get $\mathbf{e}_i^G$, and the final embedding $\mathbf{e}_i = (\mathbf{e}_i^G + \mathbf{e}_i^L)/2$.

4.3. Reasoning and Training

To train the model, let's consider a query q and the set of candidate answers $\mathcal{V}_q$, we choose a number of negative samples, denoted as e'. The embedding of the anchor nodes and the samples could be obtained using the methods described before. The query embedding result could be obtained using existing models like GQE (Hamilton et al., 2018) with the anchor node(s) embedding(s). Our objective is to optimize the embeddings in such a way that the positive sample $\mathbf{e}$ is closer to the query embedding $\mathbf{q}$, while the negative samples $\mathbf{e}'_i$, where $i = 1 \ldots k$, are pushed farther away from $\mathbf{q}$. To achieve this goal, we calculate the loss using the distance between the query embedding and each of the positive and negative samples:

$$L = -\log\sigma(\gamma - \mathrm{D}(\mathbf{q}, \mathbf{e})) - \frac{1}{k}\sum_{i=1}^{k}\log\sigma(\mathrm{D}(\mathbf{q}, \mathbf{e}'_i) - \gamma) \tag{10}$$

where γ is a scalar margin. D is the distance function. The batch loss for training samples can be formulated as follows:

$$L_b = \sum_{i=1}^{l} \frac{1}{w}L_i \tag{11}$$

Table 1: Statistics of the constructed datasets. Note that it contains all 9 query structures.

Dataset	Train	EE(Valid / Test)	ES(V/T)	SE(V/T)
FB15k-237	900,000	22,500	22,500	22,500
NELL-995	585,000	18,000	18,000	18,000

where l denotes the total number of samples in a given batch, L_i represents the loss calculated according to Equation 10 for query q_i, and $\frac{1}{w}$ denotes the weight coefficient for the sample's loss. In this article, we adopt $\frac{1}{w} = \sqrt{|\mathcal{V}_{q_i}|}$ as the weight coefficient.

5. Experiment

In this section, we provide comprehensive details of our experiments. Firstly, we curated two inductive logical query benchmarks. Secondly, we established two baseline models to serve as a point of comparison in our experiments. Lastly, we performed ablation experiments to analyze our model, to verify the effectiveness of each component.

5.1. Benchmarks Construction

We establish the required benchmarks using the FB15k-237 (Toutanova et al., 2015) and NELL-995 (Xiong et al., 2017) datasets, which are the same as the datasets used in the previous works of logical query answering, and Table 1 shows the statistics of the datasets:

Random Sampling of Unseen Entities and Triplet Splitting: We employ random selection to extract 20% of entities from the entity set $\mathcal{V}$, forming the set of emerging entities, denoted as $\mathcal{V}_{test}$. The remaining entities are considered seen entities and are represented as $\mathcal{V}_{train}$. For a triplet $r(h, t) \in \mathcal{T}$, if any entity is present in the emerging entity set, i.e., $h \in \mathcal{V}_{test}$ or $t \in \mathcal{V}_{test}$, it is an auxiliary triplet. Otherwise, it is considered a training triplet. The triplet set $\mathcal{T}$ can be split into an auxiliary triplet set $\mathcal{T}_a$ and a training triplet set $\mathcal{T}_t$.

Table 2: The inductive logical query answering results of different structures and types on FB15k-237.

Query Structure	Mean(GQE)			Feature(GQE)			Pro-QE(GQE)			Pro-QE(Q2B)		
	EE	ES	SE	EE	ES	SE	EE	ES	SE	EE	ES	SE
1p	20.0	18.2	12.3	12.2	12.7	14.4	26.1	25.2	34.0	28.1	28.4	37.8
2p	12.9	15.4	11.2	7.6	11.2	11.4	17.2	24.8	25.0	19.0	27.4	24.6
3p	9.5	15.6	9.7	5.9	12.0	9.5	14.7	21.4	20.5	15.5	25.2	19.5
2i	12.9	20.4	12.0	12.9	31.1	12.7	19.9	34.4	31.5	21.0	34.2	35.8
3i	14.1	23.6	11.2	13.7	40.7	12.5	24.6	38.7	32.4	28.2	40.3	40.2
pi	10.3	17.5	8.8	10.1	26.2	11.2	15.6	30.5	25.8	17.0	32.2	28.6
ip	13.8	25.0	11.9	11.6	32.7	11.2	17.4	34.1	21.9	19.8	38.0	22.4
2u	9.4	29.5	12.6	7.6	35.4	11.5	11.3	43.8	23.6	12.1	45.5	25.8
up	12.6	31.0	12.8	9.1	34.5	12.5	16.1	42.6	22.9	16.8	35.1	21.5
avg	12.8	21.8	11.4	10.1	26.3	11.9	18.1	32.8	26.4	**19.7**	**35.1**	**28.5**

Table 3: The inductive logical query answering results of different structures and types on NELL-995.

Query Structure	Mean(GQE)			Feature(GQE)			Pro-QE(GQE)			Pro-QE(Q2B)		
	EE	ES	SE	EE	ES	SE	EE	ES	SE	EE	ES	SE
1p	4.0	6.7	15.6	4.7	8.5	25.2	7.2	13.5	35.8	8.6	14.8	40.2
2p	18.24	17.0	10.5	10.8	12.2	12.7	29.8	36.2	25.7	31.8	38.2	27.3
3p	4.4	10.2	6.9	5.2	8.1	9.7	11.0	17.0	16.8	12.0	19.8	18.8
2i	9.1	23.7	10.8	10.1	30.2	14.8	15.0	38.4	24.3	13.7	41.3	31.1
3i	8.7	35.7	6.2	9.1	37.1	10.6	14.4	45.5	18.3	13.6	50.6	25.2
pi	9.3	20.1	11.9	9.1	26.1	18.3	18.8	34.1	32.6	19.3	37.9	40.9
ip	24.3	30.3	17.5	17.8	42.1	14.7	38.7	57.3	32.9	37.0	56.6	32.5
2u	8.9	29.6	11.0	8.5	38.6	14.2	11.7	46.5	23.0	17.2	53.9	30.2
up	22.9	32.8	15.5	15.0	50.9	13.2	33.5	57.3	27.8	33.4	58.7	26.5
avg	12.2	22.9	11.8	10.0	28.2	14.8	20.0	38.4	26.4	**20.7**	**41.3**	**30.3**

Ground the queries. The training triplet set $\mathcal{T}_t$ is utilized to construct the training graph $\mathcal{G}_t$. The entire graph $\mathcal{G}$ is used during the validation and testing steps. For different query structures in Figure 4, we ground the query in $\mathcal{G}_t$ to create the training query, where both the anchor and answer entities are exclusively seen entities. In contrast, to generate the validation and test queries, we ground the query in the graph $\mathcal{G}$, resulting in the anchor and answer entities potentially including unseen entities.

Classify the queries. The validation and test queries may contain emerging entities, so they could be categorized into four types: (1) Emerging anchor entities and emerging answer entities, (2) Emerging anchor entities and seen answer entities, (3) Seen anchor entities and emerging answer entities, and (4) Seen anchor entities and seen answer entities, denoted as **EE, ES, SE, SS** type queries. We use the first three types of queries to evaluate the inductive query reasoning ability since the last type can be answered only using the training graph.

5.2. Experimental Configurations

5.2.1. Evaluation Protocol

The evaluation of EE and ES type queries involves considering all entities in $\mathcal{V}_q$ obtained by traversing the graph because the anchor entity in these queries is an unseen entity. In contrast, for SE type queries, the answer set may contain both seen and unseen entities, and only the entities belonging to $\mathcal{V}_{test}$ are evaluated. The answer set $A = \mathcal{V}_q$ and $\mathcal{V}_q \backslash \mathcal{V}_{test}$ for $q \in EE \| ES$ and $q \in SE$, respectively. We calculate the Mean Reciprocal Rank (MRR):

$$MRR(q) = \frac{1}{|A|} \sum_{v \in A} \frac{1}{R(v)}, \qquad (12)$$

where $R(v)$ means the rank of v in all candidate entities according to our results.

5.2.2. Baselines

In our study on inductive logical query reasoning, we incorporated our proposed model, Pro-QE, with existing query embedding methods on KG such as GQE (Hamilton et al., 2018) and Q2B (Ren et al., 2020). To provide a baseline for comparison, we implemented two alternative methods. The first method involves computing the average representation of neighbor embeddings as the central embedding. The second method relies on evaluating entity features based on their neighbor relations and entities. Specifically, we train the embeddings on the training graph for the seen entities. During the test process, we compare the emerging entity with all the seen entities and use the embedding of

the seen entity that shares the most common feature (neighbor relation and entity) with the emerging entity as the anchor node.

5.2.3. Experimental Setup

Our model was implemented using the PyTorch and trained on RTX3090. We employed the ADAM optimizer. The learning rate was set to be 0.0001, and it was halved at the halfway point of the training steps to aid the model's convergence. We set the embedding dimension to 512 for entities, relation domains, relation ranges, and relations. The training batch sizes were set to 32, and the negative sample size was also set to 32. For the loss function, the margin parameter (γ) was set to be 24. We set the number of neighbors to 32 for FB15k-237 and 5 for NELL-995 datasets during inference, since it could overlap all the neighbors of 85% entities in the datasets.

5.3. Main Results

We conducted a comprehensive comparison of our model's performance against two baseline models: Mean and Feature matching, both of which are based on GQE (Hamilton et al., 2018). Additionally, we enhanced our model's capabilities by incorporating query embedding techniques from both GQE (Hamilton et al., 2018) and Q2B (Ren et al., 2020). It is important to note that the query embedding techniques employed in this investigation were designed to handle only positive queries. Integrating models equipped with negation operations would extend the overall model's capabilities to manage the negation operator.

The results shown in Table 2 and Table 3, revealed that our model consistently outperformed the baselines. Notably, our model achieved an average improvement of approximately 6.9%, 13.3%, and 17.1% (equivalent to 53.9%, 61.0%, and 150%, respectively) on EE, ES, and SE query types, compared to the Mean model, which also utilizes context information, for the FB15k-237 dataset. For the NELL-995 dataset, our model achieved an MRR improvement of 8.5%, 18.4%, and 18.5% (equivalent to 70.0%, 80.3%, and 156.8% respectively) for the three different query types, respectively.

Furthermore, we made the observation that our model based on Q2B demonstrated the most superior overall performance compared to our model based on GQE. Specifically, our results indicated that the performance of our model based on Q2B outperformed that of our model based on GQE, with an improvement of 1.6%, 2.3%, and 2.1% on FB15k-237, and 0.7%, 2.9%, and 3.9% on NELL-995, respectively. This finding is consistent with the relative performance of the two models reported in

Table 4: Ablation study results on NELL-995.

	w/o Global			w/o Info Exchange			w/o Query Prompt		
	EE	ES	SE	EE	ES	SE	EE	ES	SE
1p	6.9	12.7	34.3	6.4	10.8	31.9	5.0	8.2	25.0
2p	30.5	38.0	25.1	32.9	41.4	27.3	24.4	23.9	18.2
3p	7.5	15.2	15.0	9.9	13.9	15.9	6.9	10.9	12.1
2i	12.8	34.2	22.8	11.6	29.6	21.6	11.0	26.9	16.9
3i	12.5	37.2	17.5	11.0	36.5	15.0	9.9	34.2	10.9
pi	17.4	31.4	32.5	17.5	28.7	32.0	14.7	24.5	22.4
ip	38.9	57.5	32.6	34.8	54.2	30.6	32.6	45.7	25.7
2u	10.8	44.4	20.4	9.2	37.2	19.1	9.3	36.0	15.3
up	34.2	57.0	26.4	31.7	54.2	26.6	28.6	45.4	21.2

the original source paper (Ren et al., 2020). Additionally, the results highlighted that queries of type EE were the most challenging.

5.4. Ablation Study

To evaluate the effectiveness of the different modules, we conducted ablation experiments on the NELL-995 dataset.

To ensure that our results were consistent, we used the same logical query reasoning module, GQE (Hamilton et al., 2018). Our results are summarized in Table 4. We also compared the average MRR results of different models in Figure 5.

The results demonstrated that when disregarding *global information*, the performance of inductive reasoning was significantly hindered, showing a decline of approximately 0.9%, 2.0%, and 1.2% in absolute values (equivalent to 4.7%, 5.5%, and 4.8% relatively) for the average MRR metrics of the three query types, respectively, suggesting that local information, which can potentially be perturbed and biased, tends to limit the model's potential.

Moreover, the results revealed a noteworthy improvement in the model's performance when we incorporated the information exchange mechanism. There was an average increase of approximately 1.6%, 4.3%, and 2.0% (or 8.7%, 12.6%, and 8.2%, respectively) for the three query types (EE, ES, and SE). This indicates that the interaction between neighbors can enhance the quality of aggregation. Finally, the most significant improve-
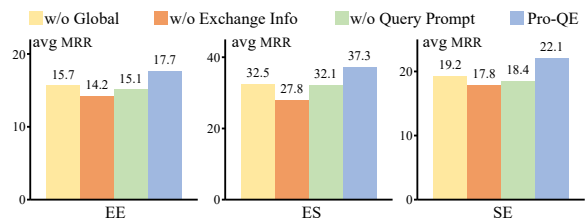

Figure 5: The average MRR results on NELL-995 of Pro-QE without different module.

ment is achieved with the *query prompt* module. The MRR results improved by about 4.2%, 10.0%, and 7.7% (equivalent to relative improvements of

26.6%, 35.2%, and 41.2%) for the EE, ES, and SE query types, respectively. This improvement was particularly pronounced for the *3p* and *pi* queries, which has the longest length from the anchor node to the answer node, and this means that there is more overall information hidden in the query.

6. Conclusion

In this paper, we address the problem of inductive logical query answering with emerging entities and present a novel framework, called Pro-QE. The model adheres to three key principles - utilizing both local and global contexts, incorporating a self-attention mechanism for information exchange, and employing a query prompt encoder to obtain an accurate representation of unseen entities. The results of the experiments conducted on these benchmarks demonstrate the efficacy of the proposed model and principles. Future work may include introducing new relations in inductive settings.

Acknowledgements

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex query answering with neural link predictors. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. 2022. Query2particles: Knowledge graph reasoning with particle embeddings. *CoRR*, abs/2204.12847.

Jiajun Chen, Huarui He, Feng Wu, and Jie Wang. 2021a. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6271–6278. AAAI Press.

Mingyang Chen, Wen Zhang, Yushan Zhu, Hongting Zhou, Zonggang Yuan, Changliang Xu, and Huajun Chen. 2022. Meta-knowledge transfer for inductive knowledge graph embedding. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 927–937. ACM.

Xuelu Chen, Ziniu Hu, and Yizhou Sun. 2021b. Fuzzy logic based logical query answering on knowledge graph. *CoRR*, abs/2108.02390.

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.

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

Mikhail Galkin, Zhaocheng Zhu, Hongyu Ren, and Jian Tang. 2022. Inductive logical query answering in knowledge graphs. *CoRR*, abs/2210.08008.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1802–1808. ijcai.org.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2030–2041.

Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Xiaoli Li, Ru Li, and Jeff Z. Pan. 2022. Type-aware embeddings for multi-hop reasoning over knowledge graphs. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 3078–3084. ijcai.org.

Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. 2021. Answering complex queries in knowledge graphs with bidirectional sequence encoders. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4968–4977. AAAI Press.

Shuwen Liu, Bernardo Cuenca Grau, Ian Horrocks, and Egor V. Kostylev. 2021. INDIGO: gnn-based inductive knowledge graph completion using pairwise encoding. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 2034–2045.

Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. 2021. Communicative message passing for inductive relation reasoning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4294–4302. AAAI Press.

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. ijcai.org.

Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. 2018. Robust discovery of positive and negative rules in knowledge bases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1168–1179. IEEE.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. DRUM: end-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15321–15331.

Komal K. Teru, Etienne G. Denis, and William L. Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9448–9457. PMLR.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1499–1509. The Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. 2019a. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7152–7159. AAAI Press.

Po-Wei Wang, Daria Stepanova, Csaba Domokos, and J Zico Kolter. 2019b. Differentiable learning of numerical rules in knowledge graphs. In *International Conference on Learning Representations*.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 564–573. Association for Computational Linguistics.

Zezhong Xu, Peng Ye, Hui Chen, Meng Zhao, Huajun Chen, and Wen Zhang. 2022a. Ruleformer: Context-aware differentiable rule mining over knowledge graph. *CoRR*, abs/2209.05815.

Zezhong Xu, Wen Zhang, Peng Ye, Hui Chen, and Huajun Chen. 2022b. Neural-symbolic entangled framework for complex query answering. *CoRR*, abs/2209.08779.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2319–2328.

Yuan Yang and Le Song. 2019. Learn to explain efficiently via neural logic inductive learning. *arXiv preprint arXiv:1910.02481*.

Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19172–19183.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. 2022. Neural-symbolic models for logical queries on knowledge graphs. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 27454–27478. PMLR.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal A. C. Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29476–29490.