Progressive Prefix-Memory Tuning for Complex Logical Query Answering on Knowledge Graphs

Xingrui Zhuo[1], Shirui Pan[2], Jiapu Wang[1], Gongqing Wu[1*], Zan Zhang[1], Rui Li[3], Zizhong Wei[3], Xindong Wu[1*]

[1]The Key Laboratory of Knowledge Engineering with Big Data (the Ministry of Education of China), School of Computer Science and Information Engineering, Hefei University of Technology, China
[2]Griffith University, Australia
[3]Shandong Inspur Science Research Institute, China
zxr@mail.hfut.edu.cn, s.pan@griffith.edu.au, jiapuwang9@gmail.com, {wugq,zanzhang}@hfut.edu.cn, {lirui01,wzz}@inspur.com, xwu@hfut.edu.cn

Abstract

Conducting complex logical queries over knowledge graphs remains a significant challenge. Recent research has successfully leveraged Pre-trained Language Models (PLMs) to tackle Knowledge Graph Complex Query Answering (KGCQA) tasks, which is attributed to PLMs' ability to comprehend logical semantics of queries through context learning. However, existing PLM-based KGCQA methods usually overlook the harm of disordered syntax or fragmented contexts within a serialized query, posing the problem of "*impossible language*" to limit PLMs in grasping the logical semantics. To address this problem, we propose a Progressive Prefix-Memory Tuning (PPMT) framework for KGCQA tasks, which effectively rectifies erroneous segments in serialized queries to assist PLMs in query answering. First, we propose a prefix-memory rectification mechanism embedded in a PLM module. This mechanism assigns rectification parameters in memory stores to polish the language segments of entities, relations, and queries through specific prefixes. To further capture the logical semantics in queries, we design a progressive fine-tuning strategy, which optimizes our model through a conditional gradient update process guided by knowledge translation constraints. Extensive experiments on widely used KGCQA benchmarks demonstrate the significant superiority of PPMT in terms of HR@3 and MRR. Our codes are available at https://github.com/lazyloafer/PPMT.

1 Introduction

Knowledge Graphs (KGs) [Bollacker *et al.*, 2008; Carlson *et al.*, 2010; Dong *et al.*, 2014], serving as tools to represent structural facts, offer interpretable information for various knowledge-dependent intelligent systems [Wang *et al.*, 2024c; Wang *et al.*, 2024a; Luo *et al.*, 2024]. As a fundamental KG-based task, Knowledge Graph Complex Query Answering (KGCQA) aims to utilize KGs to achieve explicit logical reasoning for complex queries. Thus, KGCQA typically translates a query into an Existing Positive First Order (EPFO) structure [Ren *et al.*, 2020], *i.e.*, a query composed of existing qualification ($\exists$), conjunction ($\wedge$), and disjunction ($\vee$), which intuitively expresses the reasoning logic on KGs. An example of an EPFO query is provided in Figure 1(a).

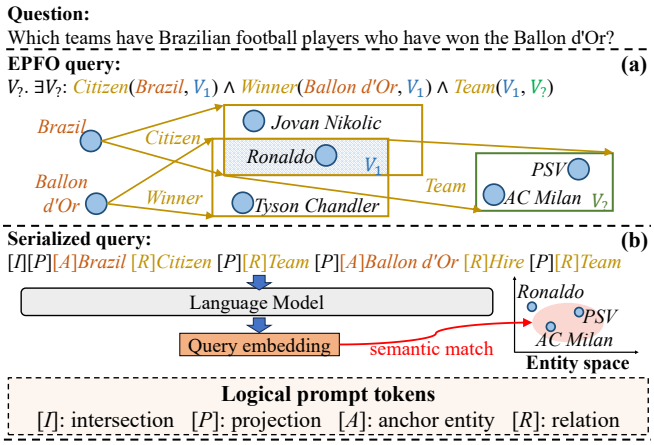

Figure 1: Mainstream KGCQA methods. (a) represents a typical KGE-based KGCQA methods that use geometric representation [Ren *et al.*, 2020] and (b) is a popular PLM-based KGCQA framework [Wang *et al.*, 2023b].

Due to the continuous emergence of new knowledge, real-world KGs remain far from complete, which limits the ability to explicitly reason EPFO queries on a KG. Building on the proven success of Knowledge Graph Embedding (KGE) in knowledge graph completion tasks [Ji *et al.*, 2022], researchers have proposed KGE-based KGCQA methods [Hamilton *et al.*, 2018; Ren *et al.*, 2020; Ren and Leskovec, 2020], which utilize knowledge representation to reason EPFO queries on incomplete KGs. Nevertheless, existing KGE-based KGCQA methods rely on local KG structures of entities and relations, lacking a global perspective on

*Corresponding author

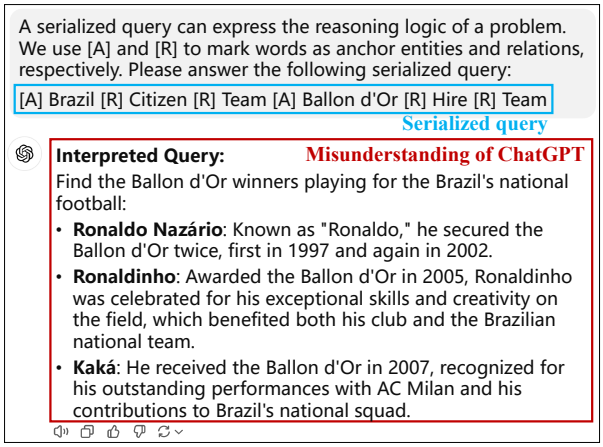

Figure 2: Case of ChatGPT misunderstanding the serialized query in Figure 1(c).

the complex logical semantics in queries [Kotnis *et al.*, 2021]. Consequently, recent studies harness Pre-trained Language Models (PLMs) to compensate for this deficiency [Zhuo *et al.*, 2025; Kotnis *et al.*, 2021; Wang *et al.*, 2023b]. As illustrated in Figure 1(b), unlike KGE-based methods, PLM-based methods convert an EPFO query into a serialized text and treat KGCQA as a textual semantic matching problem. By leveraging the global context awareness of language models developed in the pre-training stage, these methods can obtain complex entity-relation dependencies from serialized queries, thereby capturing logical semantics effectively for handling complex multi-hop queries.

Although PLM-based KGCQA methods excel at leveraging pre-trained contextual understanding, they remain significant challenges when processing serialized queries. These queries, composed of isolated entities and relations, often give rise to the phenomenon of "*impossible language*", which hinders PLMs' ability to comprehend them [Kallini *et al.*, 2024]. For instance, the serialized query in Figure 2 exemplifies an impossible language, laden with textual noise such as disordered syntax and fragmented semantics caused by concatenated entities and relations [Sharou *et al.*, 2021], ultimately disrupting the coherence required for effective reasoning. The misunderstanding of ChatGPT in Figure 2 confirms this statement. Thus, existing research [Wang *et al.*, 2023b] attempts to add trainable logical prompt tokens (Figure 1(b)) to refine the coherence of serialized queries. However, the scattered prompt tokens make the context in serialized queries more fragmented, which increases the difficulty of PLMs in understanding the query semantics and leads to serious bias in the early stages of fine-tuning [Kumar *et al.*, 2020].

To address the above issues, we propose a Progressive Prefix-Memory Tuning (PPMT) method for KGCQA tasks. First, to mitigate the interference of impossible languages on fine-tuning our model, we propose a Prefix-Memory Rectification (PMR) mechanism applied to the self-attention blocks in PLMs. This mechanism polishes the logical contexts within entities, relations, and queries by assigning specific prefixes to search rectification parameters in memory stores. To enable the model to further explore the logical semantics of a query with the condition that the textual noise is filtered out, we design a progressive fine-tuning strategy based on knowledge translation constraints, which effectively fine-tunes the PLM module through conditional gradient updating [Mokhtari *et al.*, 2020]. Moreover, to ensure parameter-efficient fine-tuning, we fix the pre-trained parameters of the token embeddings and self-attention layers in our PLM module.

Our main contributions are in three aspects:

- We propose a PMR mechanism for PLMs that are fine-tuned on KGCQA tasks, which can effectively rectify the impossible language problem in EPFO queries by enhancing the encoding process of self-attention blocks.
- To further improve our model's ability to capture the logical semantics of queries, we design a progressive fine-tuning strategy, which achieves conditional gradient updating based on knowledge translation constraints.
- Extensive experiments on two widely used benchmarks demonstrate that our method is significantly superior to 12 state-of-the-art KGCQA methods.

2 Preliminaries

In this section, we first introduce the definitions of EPFO queries and the self-attention block in PLM frameworks.

2.1 EPFO logical queries

Let $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ be a KG, where $\mathcal{E}$ and $\mathcal{R}$ are the set of entity and relation, respectively. $\tau_r(\{e_i\}, \{e_j\})$ is an assert projection that defines $r \in \mathcal{R}$ to connect two specified entities $e_i, e_j \in \mathcal{E}$.

According to [Ren *et al.*, 2020], a conjunctive EPFO query is composed of a series of existential ($\exists$) and conjunction ($\wedge$) operations:

$$
\begin{aligned}
&Q[V_?] = V_?.\exists V_1, ..., V_n : \tau_{r_1} \wedge \tau_{r_2} \wedge ... \tau_{r_k} \wedge ... \wedge \tau_{r_K}, \\
&\text{where } \tau_{r_k} = \tau_{r_k}(\{e_a\}, V), V \in \{V_?, V_1, ..., V_n\}, \\
&\text{or } \tau_{r_k} = \tau_{r_k}(V, V^*), \{V, V^*\} \in \{V_?, V_1, ..., V_n\}, V^* \neq V,
\end{aligned}
$$

where $e_a \in \mathcal{E}$ and $r_k \in \mathcal{R}$ are an anchor entity and a relation, respectively. $V_1, ..., V_n \subseteq \mathcal{E}$ are existentially quantified bound variable entities, and $V_? \subseteq \mathcal{E}$ are the target entities that we want to return. Given a query Q, the goal is to search $V_?$ to satisfy $e \in V_?$ iff $Q[e]$ is true.

In addition, a disjunctive EPFO query can be regarded as a union of multiple conjunctive queries through disjunctive ($\vee$) operations.

2.2 The Self-Attention Block in PLMs

The self-attention block is the core of PLMs. Let $\{\boldsymbol{x}_n\}_{n=1}^N$ be an input token embedding sequence, without considering position embeddings and the multi-head mechanism, the encoding process of $\boldsymbol{x}_n$ by a self-attention block can be defined as Eq. (1):

$$
\boldsymbol{z}_n = \sum_{j=1}^{N} \frac{\exp\left(\mathcal{D}_Q(\boldsymbol{x}_n) \circ \mathcal{D}_K(\boldsymbol{x}_j)\right)}{\sqrt{F} \sum_{z=1}^{N} \exp\left(\mathcal{D}_Q(\boldsymbol{x}_n) \circ \mathcal{D}_K(\boldsymbol{x}_z)\right)} \mathcal{D}_V(\boldsymbol{x}_j), \quad (1)
$$

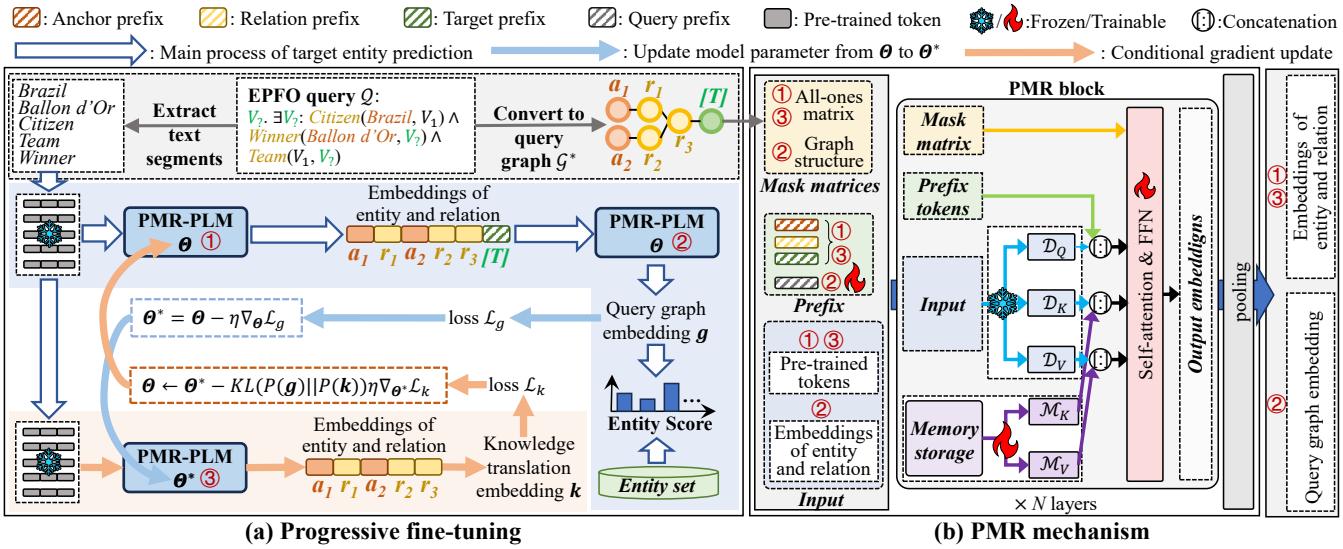

Figure 3: Overall framework of PPMT. **(a)** Given an EPFO query, a PLM module with the PMR mechanism (PMR-PLM) first ① convert the text segments of anchor entities and relations in the query to the corresponding embeddings. Then, PMR-PLM is further explored to ② obtain the graph embedding $\boldsymbol{g}$ of the query, which is utilized to predict the score of candidate target entities for the query. To optimize the model, we first update the parameters from Θ to Θ^* based on $\boldsymbol{g}$. Afterwards, the updated PMR-PLM is used to ③ recalculate the embeddings of anchor entities and relations, serving to obtain the knowledge translation embedding $\boldsymbol{k}$ and conduct conditional gradient updating. **(b)** shows the structure of the PMR mechanism, which needs to select specific prefixes and mask matrices based on the inputs of PMR-PLM at Stages ① , ② , and ③ for execution.

where $\boldsymbol{x}_n \in \mathbb{R}^F$ is the n-th F dimensional token embedding of the input sequence, $\mathcal{D}_{\{Q,K,V\}}(\cdot) : \mathbb{R}^F \to \mathbb{R}^F$ are three linear layers, and $\circ$ is an inner product operation.

The main purpose of a self-attention block is to encode $\boldsymbol{x}_n$ by weighted pooling the contextual information of $\boldsymbol{x}_n$. Therefore, numerous studies [Song *et al.*, 2023; Xue *et al.*, 2022; Zhuo *et al.*, 2024a] have captured contextual information beneficial to $\boldsymbol{x}_n$ from $\{\mathcal{D}_V(\boldsymbol{x}_n)\}_{n=1}^N$ by flexibly combining $\mathcal{D}_Q(\boldsymbol{x}_n)$ and $\{\mathcal{D}_K(\boldsymbol{x}_n)\}_{n=1}^N$, which forms a theoretical foundation of our study.

3 Methodology

In this section, we elaborate on the proposed PPMT framework in detail, which consists of two main components (Figure 3): **a PMR mechanism (Sections 3.1 and 3.2)** and **a progressive fine-tuning strategy** and **a PMR mechanism (Section 3.3)**.

3.1 The PMR Mechanism

As outlined in Section 2.2, the self-attention mechanism utilizes serialized context to embed tokens, which requires the input sequence to contain as coherent contexts as possible. Based on this, we propose a PMR mechanism to polish the language coherence of anchor entities, relations, and queries, respectively. Taking an anchor entity as an example, we provide a detailed description for our PMR mechanism.

Let $\{\boldsymbol{e}_a^{(n)} \in \mathbb{R}^F\}_{n=1}^N$ represent the token embedding sequence of an anchor entity e_a. When $\boldsymbol{e}_a^{(n)}$ is encoded by Eq. (1), the textual noise of e_a (*i.e.*, the unreasonable tokens in the sequence) is captured by $\boldsymbol{e}_a^{(n)}$, which leads to an encoding offset for $\boldsymbol{e}_a^{(n)}$. To preserve the pre-trained contextual semantic awareness of PLMs, we introduce trainable memory vectors $\{\boldsymbol{m}_s \in \mathbb{R}^F\}_{s=1}^S$ to store noise rectification parameters while freezing the pre-trained linear layers $\mathcal{D}_{\{Q,K,V\}}(\cdot)$. Therefore, Eq. (1) is improved to Eq. (2), which can rectify the local noise (token-view noise) for $\boldsymbol{e}_a^{(n)}$:

$$\boldsymbol{z}_n = \sum_{j=1}^{N} \frac{\alpha_{nj}}{\sqrt{F}\mu} \mathcal{D}_V(\boldsymbol{e}_a^{(j)}) + \sum_{s=1}^{S} \frac{\beta_{ns}}{\sqrt{F}\mu} \mathcal{M}_V(\boldsymbol{m}_s)), \tag{2}$$

where $\alpha_{nj} = \exp\left(\mathcal{D}_Q(\boldsymbol{e}_a^{(n)}) \circ \mathcal{D}_K(\boldsymbol{e}_a^{(j)})\right)$, $\beta_{ns} = \exp\left(\mathcal{D}_Q(\boldsymbol{e}_a^{(n)}) \circ \mathcal{M}_K(\boldsymbol{m}_s)\right)$, $\mu = \sum_{z=1}^{N} \alpha_{nz} + \sum_{l=1}^{S} \beta_{nl}$, and $\mathcal{M}_{\{K,V\}}(\cdot) : \mathbb{R}^F \to \mathbb{R}^F$ are two trainable linear layers. Eq. (2) is interpreted as the scaling and translation of the encoding result of Eq. (1) in an entity space, where $\frac{\alpha_{nj}}{\sqrt{F}\mu}$ is the scaling factor and $\sum_{s=1}^{S} \frac{\beta_{ns}}{\sqrt{F}\mu} \mathcal{M}_V(\boldsymbol{m}_s)$ is the translation rate.

In general, after obtaining the encoded token embeddings ($\{\boldsymbol{z}_n\}_{n=1}^N$) of e_a according to Eq. (2), the embedding of e_a is calculated by $\boldsymbol{e}_a = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{z}_n$. However, although Eq. (2) can alleviate the interference of textual noise on each $\boldsymbol{z}_n$'s encoding process, the existence of $\boldsymbol{z}_n$ is still unreasonable for e_a if the corresponding $\boldsymbol{e}_a^{(n)}$ is a noise token. Therefore, $\boldsymbol{e}_a$ obtained solely by weighted averaging all $\boldsymbol{z}_n$ may be offset due to noise $\boldsymbol{z}_n$. To address this problem, we design a prefix

factor $\boldsymbol{\varepsilon}$ to rectify global noise of e_a:

$$\boldsymbol{\varepsilon} = \sum_{j=1}^{N} \frac{\gamma_j}{\sqrt{F}\phi} \mathcal{D}_V(\boldsymbol{e}_a^{(j)}) + \sum_{s=1}^{S} \frac{\sigma_s}{\sqrt{F}\phi} \mathcal{M}_V(\boldsymbol{m}_s)), \tag{3}$$

where $\phi = \sum_{z=1}^{N} \gamma_z + \sum_{l=1}^{S} \sigma_l$, $\gamma_j = \exp(\boldsymbol{p}_a \circ \mathcal{D}_K(\boldsymbol{x}_j))$, and $\sigma_s = \exp(\boldsymbol{p}_a \circ \mathcal{M}_K(\boldsymbol{m}_s))$. $\boldsymbol{p}_a$ is a specific prefix embedding shared by all anchor entities, which helps Eq. (3) to obtain pre-trained semantics that are beneficial to e_a and the parameters for rectifing textual noise in e_a. According to the above design, the embedding of e_a can be represented as $\boldsymbol{e}_a = \text{PMR}(\{\boldsymbol{e}_a^{(n)}\}_{n=1}^{N}, \boldsymbol{p}_a)$:

$$\text{PMR}(\{\boldsymbol{e}_a^{(n)}\}_{n=1}^{N}, \boldsymbol{p}_a) = \frac{1}{N+1}(\boldsymbol{\varepsilon} + \sum_{n=1}^{N} \boldsymbol{z}_n), \tag{4}$$

where $\boldsymbol{z}_n$ and $\boldsymbol{\varepsilon}$ are obtained by Eqs. (2) and (3), respectively.

Similarly, we can obtain the embeddings of relations and target entities based on Eqs. (5) and (6), respectively.

$$\boldsymbol{r} = \text{PMR}(\{\boldsymbol{r}^{(n)}\}_{n=1}^{N}, \boldsymbol{p}_r), \tag{5}$$

$$\boldsymbol{e}_t = \text{PMR}(\{\boldsymbol{e}_t^{(n)}\}_{n=1}^{N}, \boldsymbol{p}_t), \tag{6}$$

where $\{\boldsymbol{r}^{(n)} \in \mathbb{R}^F\}_{n=1}^{N}$ and $\{\boldsymbol{e}_t^{(n)} \in \mathbb{R}^F\}_{n=1}^{N}$ are the token embedding sequences of a relation and a target entity, respectively, and $\boldsymbol{p}_r$, $\boldsymbol{p}_t \in \mathbb{R}^F$ are the specific prefix embeddings shared by all relations and target entities, respectively.

Please note that for the convenience of describing the PMR mechanism, we simplify PMR$(\cdot)$ in Eqs. (4)-(6). In actual deployment, the PMR mechanism is deployed to the multi-head attention block of each layer in our PLM module. In addition, a trainable Feed Forward Network (FFN) with layer normalization is used to aggregate multi-head attention information.

3.2 Embeddings of Query Graph and Knowledge Translation

Given a query $\mathcal{Q} = (\mathcal{A}, \mathcal{R}^*, \mathcal{G}^*)$, where $\mathcal{A}$ and $\mathcal{R}^*$ are the embedding sets of anchor entities and relations within $\mathcal{Q}$, respectively. As shown in Figure 3(a), $\mathcal{G}^*$ is the query graph of $\mathcal{Q}$ that considers $\mathcal{A} \bigcup \mathcal{R}^*$ as a node embedding set and masks the target entity node using a specific prefix embedding $\boldsymbol{p}_t$ (refer to *[target]* in Figure 3). According to Section 3.1, the embeddings of all anchor entities and relations in a query can be obtained. Afterwards, we carry out corresponding query graph embedding and knowledge translation embedding.

Query Graph Embedding. The purpose of query graph embedding is to encode the logical context of a query in the form of the logical structure. To ensure that the model can further rectify the textual noise introduced by entities and relations from the perspective of the query while obtaining the logical context in the query graph, we share the model parameters mentioned in Section 3.1 for obtaining the query graph embedding $\boldsymbol{g}$:

$$\boldsymbol{g} = \text{PMR}(\mathcal{A} \cup \mathcal{R}^* \cup \{\boldsymbol{p}_t\}, \boldsymbol{p}_q), \tag{7}$$

where $\boldsymbol{p}_q \in \mathbb{R}^F$ is a specific prefix embedding shared by all queries and $\boldsymbol{g} \in \mathbb{R}^F$ is the graph embedding of query $\mathcal{Q}$.

In addition, we construct a mask matrix for the attention weights in PMR$(\cdot)$ based on the structure of $\mathcal{G}^*$. The specific construction rules of the mask matrix are as follows:

Rule 1: $\mathcal{A} \cup \mathcal{R}^* \cup \{\boldsymbol{p}_t\}$ are visible to themselves and one-hop neighbors in $\mathcal{G}^*$.

Rule 2: Memory vectors $\{\boldsymbol{m}_s \in \mathbb{R}^F\}_{s=1}^{S}$ (first provided in Eq. (2)) are visible to $\mathcal{A} \cup \mathcal{R}^* \cup \{\boldsymbol{p}_t\}$.

Rule 3: $\{\boldsymbol{m}_s \in \mathbb{R}^F\}_{s=1}^{S}$ and $\mathcal{A} \cup \mathcal{R}^* \cup \{\boldsymbol{p}_t\}$ are visible to the prefix $\boldsymbol{p}_q$.

Knowledge Translation Embedding. Unlike the complex logical contexts within query graphs, the knowledge translation operation aims to provide a more intuitive mapping between queries and target entities by elucidating the spatial geometric dependencies between entities and relations, which alleviates the interference of the inductive logical context bias (possibly caused by textual noise in queries) on the model [Zhuo *et al.*, 2024b]. However, considering the limitation of translation embedding in capturing complex logical semantics in queries [Liu *et al.*, 2022], the knowledge translation embedding of $\mathcal{Q}$ is used to assist in refining the optimization result of Eq. (7) (see Section 3.3 for details). Specifically, we use the GQE algorithm [Hamilton *et al.*, 2018] to implement knowledge translation embedding of $\mathcal{Q}$:

$$\boldsymbol{k} = \text{GQE}(\mathcal{A}, \mathcal{R}^*). \tag{8}$$

When $\mathcal{Q}$ only contains the relation projection operation, Eq. (8) is a standard GQE encoding process. When $\mathcal{Q}$ contains the conjunction logic, Eq. (8) will average the GQE results of all the longest sub-projection paths in $\mathcal{Q}$ to obtain $\boldsymbol{k} \in \mathbb{R}^F$.

3.3 Progressive Fine-tuning and Inference

Let $\boldsymbol{\Theta}_l$ represent the model parameter in the l-th fine-tuning step, $\boldsymbol{e}_t^+$ and $\{\boldsymbol{e}_t^{-(i)}\}_{i=1}^{I}$ are the embeddings of sampled positive and negative target entities, respectively, obtained by Eq. (6). The loss functions of Eqs. (7) and (8) are represented as Eqs. (9) and (10), respectively:

$$\mathcal{L}_g = -\log\left(\frac{\exp(\boldsymbol{e}_t^+ \circ \boldsymbol{g})}{\exp(\boldsymbol{e}_t^+ \circ \boldsymbol{g}) + \sum_{i=1}^{I} \exp(\boldsymbol{e}_t^- \circ \boldsymbol{g})}\right), \tag{9}$$

$$\mathcal{L}_k = -\log\left(\frac{\exp(\boldsymbol{e}_t^+ \circ \boldsymbol{k})}{\exp(\boldsymbol{e}_t^+ \circ \boldsymbol{k}) + \sum_{i=1}^{I} \exp(\boldsymbol{e}_t^- \circ \boldsymbol{k})}\right). \tag{10}$$

In the progressive fine-tuning process, we take $\boldsymbol{g}$, which contains richer logical context, as the main objective of optimization. The geometric dependencies of entity-relation in $\boldsymbol{k}$ is used as auxiliary constraints to assist the model in correcting the inductive logical context bias that may exist in $\boldsymbol{g}$. Let $\boldsymbol{\Theta}_l$ be the model parameter at the l-th fine-tuning step, the specific updating process is as follows.

First, $\boldsymbol{\Theta}_l$ is updated by Eq. (11):

$$\boldsymbol{\Theta}_l^* = \boldsymbol{\Theta}_l - \eta \nabla_{\boldsymbol{\Theta}_l} \mathcal{L}_g. \tag{11}$$

Then, we use Eq. (12) to rectify the update biases that may occur in $\boldsymbol{\Theta}_l^*$ due to the inductive logical context, *i.e.*, the conditional gradient updating:

$$\boldsymbol{\Theta}_{l+1} = \boldsymbol{\Theta}_l^* - \text{KL}(P(g)||P(k))\eta \nabla_{\boldsymbol{\Theta}_l^*} \mathcal{L}_k, \tag{12}$$

Dataset	Entities	Relations	Training Queries	validating Queries	Testing Queries
FB15k-237	14,505	474	748,445	60,101	62,812
NELL-995	63,361	400	539,910	48,927	49,034

Table 1: Element statistics of the experimental datasets.

where η is the learning rate, $\mathrm{KL}(P(g)||P(k))$ is a descent direction [Mokhtari *et al.*, 2020] calculated by the KL divergence, and $P(g)$ and $P(k)$, calculated using $\boldsymbol{g}$ and $\boldsymbol{k}$, respectively, represent the distributions of predicted scores at the l-th step for the same set of positive and negative samples.

It is important to note that after obtaining embeddings of entities and relations and performing Eq. (11), we first clear the gradients of the model. Subsequently, we obatin embeddings of entities and relations again and execute Eq. (12). Therefore, $\nabla_{\boldsymbol{\Theta}_l^*}\mathcal{L}_k$ does not propagate through $\boldsymbol{\Theta}_l^*$ to $\boldsymbol{\Theta}_l$, that is, Eq. (12) only corrects the updated baises within $\boldsymbol{\Theta}_l^*$.

When inferring candidate entities of a conjunctive query, we utilize $\boldsymbol{g}$ to obtain the inner product scores of all entity embeddings that are obtained by Eq. (6). For a disjunctive query, we first obtain the scores for all entities against each inextensible sub-conjunctive query, and use the fuzzy logic theory [Chen *et al.*, 2022] to aggregate these scores across all sub-conjunctive queries as the final entity scores for the disjunctive query.

4 Experiments

In this section, we mainly answer the following research questions: **RQ1**. Does the PPMT method significantly outperform the state-of-the-art baselines in KGCQA tasks? **RQ2**. Does the PMR mechanism and progressive fine-tuning approach enhance the effectiveness of our method? **RQ3**. Is the fine-tuning strategy effective for the FFN module within the PPMT framework? **RQ4**. Is the PPMT framework adaptable to various pre-trained language models?

4.1 Dataset Descriptions

As we focus on handling the impossible language problem in serialized EPFO queries, we follow the previous work [Wang *et al.*, 2023b] to conduct experiments on the KGCQA benchmarks pre-processed by [Ren *et al.*, 2020]. Table 1 provides the statistics of the experimental datasets.

According to the setup of previous studies [Ren *et al.*, 2020; Ren and Leskovec, 2020; Zhang *et al.*, 2021], we train PPMT using five fundamental query types (1p, 2p, 3p, 2i, and 3i). It is then evaluated across a broader range of nine query types (1p, 2p, 3p, 2i, 3i, ip, pi, 2u, and up).

4.2 Baselines and Evaluation Metrics

We compare our method with 12 state-of-the-art baselines, which can be split into three mainstream categories of KGCQA methods, *e.i.*, **geometric-based KGE methods** (GQE [Hamilton *et al.*, 2018], Q2B [Ren *et al.*, 2020], ConE [Zhang *et al.*, 2021], and TEMP [Hu *et al.*, 2022]), **logical-based KGE methods** (BetaE [Ren and Leskovec, 2020], FuzzQE [Chen *et al.*, 2022], CQD-Beam [Arakelyan *et al.*, 2021], LMPNN [Wang *et al.*, 2023c], Var2Vec [Wang *et al.*, 2023a], and UltraQuery [Bai *et al.*, 2024]), and **PLM-based KGCQA methods** (BiQE [Kotnis *et al.*, 2021] and SILR [Wang *et al.*, 2023b]).

To evaluate the performance of each model, we use two standard metrics, Mean Reciprocal Rank (MRR) and top-3 Hit Rate (HR@3).

4.3 Implementation Settings

In our experiments, we first convert EPFO queries to query graphs shown in Figure 3(a) that are suitable for the proposed PPMT method. Then, we train our model within 30 epochs on a NVIDIA A100 GPU with 95 GB memory. The batch size for training queries and the number of sampled negative target entities for each training query are set to 128. We use the Adam optimizer to update the model parameters and set the learning rate to 1.5×10^{-4} with 10% linear warmup steps. Furthermore, we evaluate the adaptability of our PPMT framework using three BERT-like backbones [Devlin *et al.*, 2019] (BERT, Distil-BERT, and RoBERTa). The dimension F in Section 3 is based on the embedding dimension of different PLMs. We explore the optimal value of the memory matrix's dimension S in the PMR mechanism from $\{1, 5, 10, 15, 20, 25, 30, 35, 40\}$.

4.4 Main Results (RQ1)

Table 2 provides the experimental results of each method on two datasets. Our proposed PPMT outperforms other methods significantly in terms of average HR@3 and MRR, with outstanding performance on the four fundamental multi-hop query types (2p, 3p, 2i, and 3i). This success is attributed to the proposed PMR mechanism and progressive fine-tuning strategy, which effectively refine the context and learn sufficient logical semantics of serialized queries.

Benefiting from the context acquisition capability of the Transformer architecture and the pre-trained latent knowledge representations, PLM-based baselines (BiQE and SILR) demonstrate relatively strong performance. However, the challenge for PLM-based approaches lies in managing textual noise. This difficulty often leads them to fit to noisy data distributions, which hinders their understanding of logical semantics and limits their performance on multi-hop complex queries (*e.g.*, 2p, 3p, 2i, and 3i).

Based on the conclusion of [Li and Liang, 2021], we consider that the infix-prompt method used by SILR may prevent prompt tokens from effectively rectifying textual noises in query graphs. In contrast, PPMT's distinct separation of entity/relation and query encoding processes ensure that prompt tokens always serve as prefixes, which facilitates more effective context adjustment of all pre-trained tokens within texts.

4.5 Ablation Studies (RQ2)

We conduct ablation experiments to further analyze the effectiveness of the PMR mechanism and the progressive fine-tuning method of our PPMT framework. The related experimental results is provided in Table 3.

Effectiveness of the PMR mechanism. We validate the effectiveness of the PMR mechanism from two aspects: prefix tokens and memory matrices. As shown in Table 3, PPMT

FB15k-237

Method	Avg.		1p		2p		3p		2i		3i		ip		pi		2u		up	
	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR
GQE	0.221	0.203	0.404	0.346	0.214	0.193	0.147	0.145	0.262	0.250	0.390	0.355	0.087	0.086	0.162	0.156	0.164	0.145	0.155	0.151
Q2B	0.268	0.237	0.467	0.421	0.240	0.232	0.186	0.184	0.324	0.254	0.453	0.355	0.113	0.112	0.198	0.177	0.239	0.217	0.193	0.183
ConE	0.296	0.271	0.465	0.420	0.270	0.253	0.231	0.218	0.350	0.316	0.488	**0.445**	0.120	0.113	0.241	0.220	0.305	0.267	0.201	0.188
TEMP	0.294	0.263	0.457	0.420	0.278	0.271	0.234	0.231	**0.369**	0.300	**0.496**	0.397	0.117	0.126	0.229	0.193	0.276	0.255	0.189	0.178
BetaE	0.262	0.238	0.426	0.383	0.248	0.231	0.208	0.197	0.311	0.277	0.439	0.392	0.102	0.099	0.217	0.196	0.237	0.209	0.174	0.165
FuzzQE	0.269	0.247	0.462	0.418	0.273	0.255	0.233	0.218	0.294	0.264	0.404	0.363	0.125	0.122	0.200	0.185	0.237	0.215	0.198	0.186
CQD-Beam	0.290	0.266	0.512	**0.467**	0.288	0.274	0.221	0.211	0.352	0.316	0.457	0.417	0.129	0.112	0.249	0.216	0.284	0.256	0.121	0.133
LMPNN	0.312	0.284	**0.524**	0.459	0.291	0.289	0.237	0.235	0.348	0.303	0.460	0.426	0.189	0.168	0.252	0.221	0.289	0.264	0.218	0.196
Var2Vec	0.284	0.268	0.503	0.455	0.273	0.266	0.220	0.209	0.329	0.273	0.401	0.423	**0.199**	**0.177**	0.171	0.180	0.244	0.226	**0.221**	**0.207**
UltraQuery	0.292	0.269	0.445	0.406	0.265	0.252	0.216	0.205	0.354	0.318	0.481	0.436	0.172	0.169	0.263	0.233	0.240	0.221	0.193	0.185
BiQE†	—	—	0.439	0.401	0.281	0.222	0.239	0.108	0.333	0.288	0.474	0.360	0.110	0.112	0.177	0.143	—	—	—	—
SILR	0.296	0.263	0.471	0.432	0.302	0.288	0.249	0.240	0.358	0.282	0.484	0.414	0.113	0.100	0.222	0.195	0.283	0.253	0.181	0.167
PPMT‡	**0.323**	**0.292**	0.481	0.435	**0.329**	**0.305**	**0.271**	**0.252**	0.362	**0.322**	**0.496**	0.436	0.160	0.152	**0.264**	**0.235**	**0.334**	**0.297**	0.212	0.202

NELL-995

Method	Avg.		1p		2p		3p		2i		3i		ip		pi		2u		up	
	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR
GQE	0.246	0.211	0.418	0.311	0.228	0.193	0.205	0.174	0.316	0.273	0.447	0.408	0.081	0.080	0.186	0.170	0.199	0.159	0.139	0.13
Q2B	0.305	0.254	0.555	0.413	0.266	0.227	0.233	0.208	0.343	0.288	0.480	0.414	0.132	0.125	0.212	0.193	0.369	0.266	0.163	0.155
ConE	0.334	0.304	0.575	0.520	0.268	0.249	0.283	0.268	0.380	0.335	0.531	0.484	0.121	0.112	0.217	0.203	0.419	0.369	0.214	0.195
TEMP	0.373	0.311	0.625	0.533	0.343	0.284	0.342	0.286	0.410	0.333	0.552	0.471	0.141	0.129	0.209	0.185	0.477	0.368	0.262	0.211
BetaE	0.322	0.279	0.565	0.515	0.269	0.251	0.283	0.259	0.352	0.210	0.491	0.442	0.109	0.102	0.223	0.205	0.394	0.354	0.209	0.180
FuzzQE	0.340	0.305	0.603	0.549	0.357	0.314	0.336	0.307	0.326	0.291	0.450	0.397	0.135	0.129	0.230	0.208	0.376	0.331	0.246	0.222
CQD-Beam	0.375	0.324	0.667	0.604	0.350	0.309	0.288	0.259	0.410	0.348	0.529	0.443	**0.171**	**0.167**	**0.277**	0.252	**0.531**	**0.410**	0.156	0.134
LMPNN	0.367	0.328	0.671	0.606	0.355	0.314	0.313	0.282	0.393	0.342	0.533	0.452	0.165	0.158	0.249	0.226	0.443	0.393	0.178	0.186
Var2Vec	0.319	0.297	**0.681**	**0.607**	0.269	0.241	0.225	0.216	0.338	0.310	0.488	0.482	0.166	0.154	0.230	0.221	0.224	0.204	**0.251**	**0.239**
UltraQuery	0.276	0.250	0.428	0.389	0.230	0.203	0.225	0.207	0.345	0.308	0.493	0.444	0.146	0.140	0.276	**0.256**	0.169	0.148	0.173	0.153
BiQE†	—	—	0.587	0.467	0.305	0.209	0.326	0.197	0.371	0.305	0.531	0.473	0.103	0.119	0.187	0.201	—	—	—	—
SILR	0.346	0.302	0.641	0.521	0.317	0.289	0.348	0.312	0.374	0.330	0.552	0.478	0.088	0.080	0.181	0.191	0.433	0.338	0.182	0.187
PPMT‡	**0.387**	**0.342**	0.640	0.562	**0.363**	**0.322**	**0.362**	**0.323**	**0.412**	**0.350**	**0.569**	**0.496**	0.169	**0.167**	0.254	0.240	0.482	0.403	0.241	0.219

† BiQE [Kotnis *et al.*, 2021] does not provide a prediction method for disjunctive queries (2u and up).

‡ The results of PPMT are uniformly trained using BERT as the pre-trained model. The memory sizes used for different datasets are shown in Figure 5.

Table 2: Main Experimental results. Bold font and underline represent the best and second-best results, respectively.

Model	FB15k-237		NELL-995	
	Avg. HR@3	Avg. MRR	Avg. HR@3	Avg. MRR
PPMT	**0.323**	**0.292**	**0.387**	**0.342**
-prefix	0.320	0.289	0.368	0.324
-mem	0.291	0.268	0.330	0.305
-pft	0.317	0.287	0.355	0.313
-pft-KL	0.300	0.271	0.346	0.311
-pft-$\mathcal{L}_k$	0.295	0.270	0.349	0.307
-pft-$\mathcal{L}_g$	0.272	0.250	0.329	0.286

Table 3: Results of ablation experiments. Bold font is the best results. "*-prefix*", "*-mem*", and "*-pft*" indicate that PPMT does not use the prefix tokens, memory matrices, and the progressive fine-tuning strategy, respectively. KL, $\mathcal{L}_k$, and $\mathcal{L}_g$ are defined in Eqs. (12), (10), and (9), respectively.

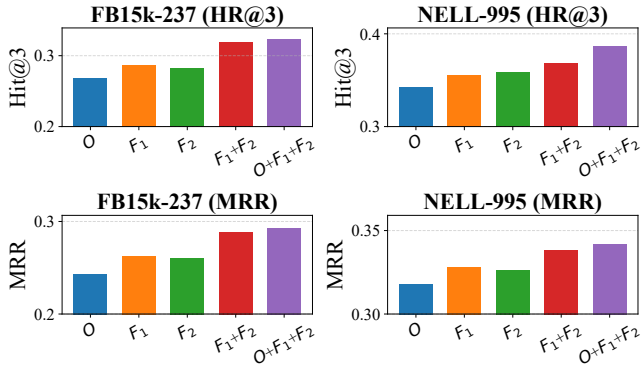

Figure 4: Results of fine-tuning different components.

is comprehensively superior to PPMT (*-prefix*) and PPMT (*-mem*). In addition, we find that the memory matrices have a greater improvement on the model compared to the prefix vectors, because memory matrices can correct finer-grained textual noise through spatial scaling and translating embeddings. In contrast, prefix vectors adjust the embeddings of entities, relations, and queries from a broader perspective, potentially overlooking some finer-grained noise.

Effectiveness of the progressive fine-tuning method. We define four variants, $\mathbf{\Theta}_l$ can be updated by Eqs. (11) and (12) without the intermediate gradient clearing operation (*-pft*), $\mathbf{\Theta}_l$ can be updated by Eqs. (11) and (12) without KL divergence weighting (*-pft-KL*), $\mathbf{\Theta}_l$ is only updated by Eq. (9) (*-pft-*$\mathcal{L}_k$), and $\mathbf{\Theta}_l$ is only updated by Eq. (10) (*-pft-*$\mathcal{L}_g$), to comprehensively evaluate the effectiveness of our progressive fine-tuning method. Table 4 shows that the performance of different variants has decreased to a certain extent compared to PPMT, with PPMT (*-pft-*$\mathcal{L}_g$) showing the most significant decrease in effectiveness, as it is difficult to mine the complex logical semantics in EPFO queries using only knowledge transformation embeddings.

4.6 Further Analysis

In this section, we mainly analyze the details of the model, including fine-tuning measures, model adaptability, and memory size selection.

Analysis of Fine-tuning FFN (**RQ3**). Let O represent the linear layer in the SA block, and F_1 and F_2 denote the two lin-

Basic Model	FB15k-237		NELL-995	
	Avg. HR@3	Avg. MRR	Avg. HR@3	Avg. MRR
BERT	0.323	0.292	0.387	0.342
Distil-BERT	0.303	0.276	0.381	0.334
RoBERTa	0.318	0.281	0.385	0.340

Table 4: PPMT's performance on different basic models.

ear layers in the FFN module, respectively. We fine tune the different combinations of these components, and the results are shown in Figure 4 (PPMT adopts an $O + F_1 + F_2$ fine-tuning method). Our experimental results indicate that fine-tuning the FFN module significantly enhances the model's performance because the FFN module integrates pre-trained information with additional memory parameters, enabling the model to comprehensively learn the representations of entities, relations, and queries.

Adaptability Analysis of PPMT (**RQ4**). As shown in Table 4, we use BERT, Distil-BERT, and RoBERTa as the basic models for PPMT to verify its adaptability under unified training parameters. We find that the PPMT framework can achieve considerable results on several variants of BERT, and the training parameter level is more advantageous than full parameter fine-tuning.

Memory Size Selection. Figure 5 provides the experimental results of PPMT with different sizes of memory matrices (*i.e.*, S of the memory matrix $\boldsymbol{M}$ in Section 3.2). When $S \in [5, 15]$, the performance of the model significantly improves, which results from the increased number of rectification parameters in the memory matrices, allowing for a more comprehensive fitting of the implicit rectification rules for entities, relations, and queries. In addition, the memory size varies across different datasets, influenced by the sizes of the entities and relations involved. When $S \in [15, 25]$, PPMT performs the best on FB15k-237; because the number of entities in NELL-995 is much greater than that in FB15k-237, PPMT performs the best on NELL-995 when $S \in [25, 35]$. In our experiments, we set S to 20 and 30 for FB15k-237 and NELL-995, respectively.

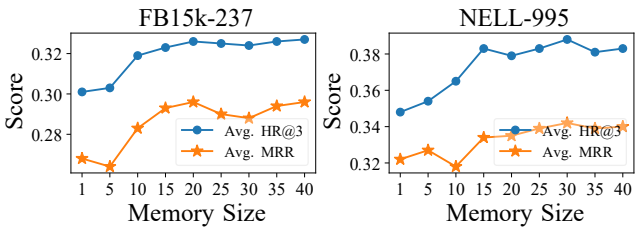

Figure 5: Results of PPMT with different memory sizes.

5 Related Work

The KGCQA task is an extension of knowledge graph link prediction [Bordes *et al.*, 2013; Sun *et al.*, 2019; Trouillon *et al.*, 2016; Arakelyan *et al.*, 2021; Zhu *et al.*, 2022]. Therefore, early KGCQA methods focus on using logical expressions [Demeester *et al.*, 2016; Rocktäschel *et al.*, 2015] to enhance understanding of knowledge graph relation paths. However, the above methods based on logical paths may exhibit exponential retrieval complexity when dealing with multi-hop problems, and are not suitable for incomplete KGs.

To achieve knowledge retrieval with linear complexity and enhance the model's generalization, various KGE-based KGCQA methods [Liu *et al.*, 2022; Yang *et al.*, 2022; Choudhary *et al.*, 2021; Huang *et al.*, 2022; Bai *et al.*, 2023; Zhu *et al.*, 2022; Ren *et al.*, 2021; Luo *et al.*, 2023] have been proposed, which implement query level embeddings from different perspectives and can obtain answers through only one round of traversal of candidate entities. GQE [Hamilton *et al.*, 2018] uses the KGE method to model the relation projection of queries, achieving query embeddings that can be used for retrieval. Q2B [Ren *et al.*, 2020] utilizes box embedding to achieve spatial geometry modeling for complex queries. ConE [Zhang *et al.*, 2021] extends the spatial representation of complex queries using cone embedding. Methods such as BetaE [Ren and Leskovec, 2020], FuzzQE [Chen *et al.*, 2022], and Var2Vec [Wang *et al.*, 2023a] attempt to combine probability logic with beta distribution, fuzzy logic, and triangular norms to achieve semantic expansion of query embeddings. However, due to the sparsity of incomplete KGs, the above methods may overlook the latent semantics of queries, thereby limiting prediction quality.

PLMs achieve remarkable results in various NLP tasks because of their ability to flexibly obtain contextual information of data through the Transformer framework [Wang *et al.*, 2024b; Jin *et al.*, 2024]. With more research proving the latent knowledge representation ability of PLMs, the PLM-based KGCQA framework has become a new trend [Kotnis *et al.*, 2021; Wang *et al.*, 2023b]. For instance, BiQE [Kotnis *et al.*, 2021] introduces a bidirectional Transformer framework to capture fine-grained semantic representations in structural queries. SILR [Wang *et al.*, 2023b] achieves fine-tuning of PLMs on EPFO queries through soft prompt learning [Liu *et al.*, 2021]. However, the textual noise in a query has caused serious interference to PLMs in understanding the semantics of the query, which is currently overlooked by PLM-based methods and a focus of our study.

6 Conclusion

In this paper, we analyze the "impossible language" dilemma faced by existing PLM-based KGCQA methods and propose a novel PPMT framework to overcome this challenge. Specifically, we propose a PMR mechanism that leverages specific prefixes to extract rectification parameters from the memory stores of each self-attention block in a PLM module. These parameters can polish the erroneous segments in serialized queries, thereby suppressing the interference of the impossible language problem on PLM's understanding of query logical semantics. Then, a progressive fine-tuning strategy is proposed to assist PLMs in understanding the logical semantics of queries. The experimental results highlight the prediction performance, effectiveness of components, fine-tuning strategies, and PPMT's adaptability, underscoring the effectiveness of our proposed method.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 62120106008 and 62472136; the Program for Innovative Research Team at the University of the Ministry of Education under Grant IRT_17R32; Anhui Provincial Science and Technology Fortification Plan, China, under Grant 202423k09020015; and the Key Laboratory of Knowledge Engineering with Big Data (the Ministry of Education of China) under Grant BigKEOpen2025-01.

References

[Arakelyan *et al.*, 2021] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *ICLR*. OpenReview.net, 2021.

[Bai *et al.*, 2023] Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. Answering complex logical queries on knowledge graphs via query computation tree optimization. In *ICML*, volume 202, pages 1472–1491. PMLR, 2023.

[Bai *et al.*, 2024] Jiaxin Bai, Xin Liu, Weiqi Wang, Chen Luo, and Yangqiu Song. A foundation model for zero-shot logical query reasoning. In *NeurIPS*, volume 36, pages 30534–30553. Curran Associates, Inc., 2024.

[Bollacker *et al.*, 2008] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In Jason Tsong-Li Wang, editor, *ACM SIGMOD*, pages 1247–1250. ACM, 2008.

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

[Carlson *et al.*, 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313. AAAI Press, 2010.

[Chen *et al.*, 2022] Xuelu Chen, Ziniu Hu, and Yizhou Sun. Fuzzy logic based logical query answering on knowledge graphs. In *AAAI*, pages 3939–3948. AAAI Press, 2022.

[Choudhary *et al.*, 2021] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *WWW*, pages 1373–1384. ACM / IW3C2, 2021.

[Demeester *et al.*, 2016] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, pages 1389–1399. The Association for Computational Linguistics, 2016.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186. Association for Computational Linguistics, 2019.

[Dong *et al.*, 2014] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *ACM SIGKDD*, pages 601–610. ACM, 2014.

[Hamilton *et al.*, 2018] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *NeurIPS*, pages 2030–2041, 2018.

[Hu *et al.*, 2022] Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Xiaoli Li, Ru Li, and Jeff Z. Pan. Type-aware embeddings for multi-hop reasoning over knowledge graphs. In *IJCAI*, pages 3078–3084. ijcai.org, 2022.

[Huang *et al.*, 2022] Zijian Huang, Meng-Fen Chiang, and Wang-Chien Lee. Line: Logical query reasoning over hierarchical knowledge graphs. In *ACM SIGKDD*, pages 615–625. ACM, 2022.

[Ji *et al.*, 2022] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE TNNLS*, 33(2):494–514, 2022.

[Jin *et al.*, 2024] Can Jin, Tong Che, Hongwu Peng, Yiyuan Li, Dimitris N. Metaxas, and Marco Pavone. Learning from teaching regularization: Generalizable correlations should be easy to imitate. In *NeurIPS*, pages 966–994. Curran Associates, Inc., 2024.

[Kallini *et al.*, 2024] Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. Mission: Impossible language models. In *ACL*, pages 14691–14714. Association for Computational Linguistics, 2024.

[Kotnis *et al.*, 2021] Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. Answering complex queries in knowledge graphs with bidirectional sequence encoders. In *AAAI*, pages 4968–4977. AAAI Press, 2021.

[Kumar *et al.*, 2020] Ankit Kumar, Piyush Makhija, and Anuj Gupta. Noisy text data: Achilles' heel of BERT. In *W-NUT@EMNLP*, pages 16–21. Association for Computational Linguistics, 2020.

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597. Association for Computational Linguistics, 2021.

[Liu *et al.*, 2021] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[Liu *et al.*, 2022] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. Mask and reason: Pre-training knowledge graph transformers for complex logical queries. In *ACM SIGKDD*, pages 1120–1130. ACM, 2022.

[Luo *et al.*, 2023] Haoran Luo, Haihong E, Yuhao Yang, Gengxian Zhou, Yikai Guo, Tianyu Yao, Zichen Tang, Xueyuan Lin, and Kaiyang Wan. NQE: n-ary query

embedding for complex query answering over hyper-relational knowledge graphs. In *AAAI*, pages 4543–4551. AAAI Press, 2023.

[Luo *et al.*, 2024] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR*. OpenReview.net, 2024.

[Mokhtari *et al.*, 2020] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *J. Mach. Learn. Res.*, 21:105:1–105:49, 2020.

[Ren and Leskovec, 2020] Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *NeurIPS*, pages 19716–19726, 2020.

[Ren *et al.*, 2020] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *ICLR*. OpenReview.net, 2020.

[Ren *et al.*, 2021] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. LEGO: latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *ICML*, volume 139, pages 8959–8970. PMLR, 2021.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *NAACL*, pages 1119–1129. The Association for Computational Linguistics, 2015.

[Sharou *et al.*, 2021] Khetam Al Sharou, Zhenhao Li, and Lucia Specia. Towards a better understanding of noise in natural language processing. In *RANLP*, pages 53–62. INCOMA Ltd., 2021.

[Song *et al.*, 2023] Jiechong Song, Chong Mou, Shiqi Wang, Siwei Ma, and Jian Zhang. Optimization-inspired cross-attention transformer for compressive sensing. In *CVPR*, pages 6174–6184. IEEE, 2023.

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

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[Wang *et al.*, 2023a] Dingmin Wang, Yeyuan Chen, and Bernardo Cuenca Grau. Efficient embeddings of logical variables for query answering over incomplete knowledge graphs. In *AAAI*, pages 4652–4659. AAAI Press, 2023.

[Wang *et al.*, 2023b] Siyuan Wang, Zhongyu Wei, Meng Han, Zhihao Fan, Haijun Shan, Qi Zhang, and Xuanjing Huang. Query structure modeling for inductive logical reasoning over knowledge graphs. In *ACL*, pages 4706–4718. Association for Computational Linguistics, 2023.

[Wang *et al.*, 2023c] Zihao Wang, Yangqiu Song, Ginny Y. Wong, and Simon See. Logical message passing networks with one-hop inference on atomic formulas. In *ICLR*. OpenReview.net, 2023.

[Wang *et al.*, 2024a] Jiapu Wang, Zheng Cui, Boyue Wang, Shirui Pan, Junbin Gao, Baocai Yin, and Wen Gao. IME: integrating multi-curvature shared and specific embedding for temporal knowledge graph completion. In *ACM WWW*, pages 1954–1962. ACM, 2024.

[Wang *et al.*, 2024b] Jiapu Wang, Kai Sun, Linhao Luo, Wei Wei, Yongli Hu, Alan Wee-Chung Liew, Shirui Pan, and Baocai Yin. Large language models-guided dynamic adaptation for temporal knowledge graph reasoning. In *NeurIPS*, pages 8384–8410. Curran Associates, Inc., 2024.

[Wang *et al.*, 2024c] Jiapu Wang, Boyue Wang, Junbin Gao, Shirui Pan, Tengfei Liu, Baocai Yin, and Wen Gao. MADE: multicurvature adaptive embedding for temporal knowledge graph completion. *IEEE TCYB*, 54(10):5818–5831, 2024.

[Xue *et al.*, 2022] Dizhan Xue, Shengsheng Qian, Quan Fang, and Changsheng Xu. MMT: image-guided story ending generation with multimodal memory transformer. In *ACM MM*, pages 750–758. ACM, 2022.

[Yang *et al.*, 2022] Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. GammaE: Gamma embeddings for logical queries on knowledge graphs. In *EMNLP*, pages 745–760. Association for Computational Linguistics, 2022.

[Zhang *et al.*, 2021] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. In *NeurIPS*, pages 19172–19183, 2021.

[Zhu *et al.*, 2022] Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-symbolic models for logical queries on knowledge graphs. In *ICML*, volume 162, pages 27454–27478. PMLR, 2022.

[Zhuo *et al.*, 2024a] Xingrui Zhuo, Shengsheng Qian, Jun Hu, Fuxin Dai, Kangyi Lin, and Gongqing Wu. Multi-hop multi-view memory transformer for session-based recommendation. *ACM TOIS*, 42(6):144:1–144:28, 2024.

[Zhuo *et al.*, 2024b] Xingrui Zhuo, Gongqing Wu, Zan Zhang, and Xindong Wu. Geometric-contextual mutual infomax path aggregation for relation reasoning on knowledge graph. *IEEE TKDE*, 36(7):3076–3090, 2024.

[Zhuo *et al.*, 2025] Xingrui Zhuo, Jiapu Wang, Gongqing Wu, Shirui Pan, and Xindong Wu. Effective instruction parsing plugin for complex logical query answering on knowledge graphs. In *ACM WWW*, pages 4780–4792. ACM, 2025.