Enhancing Complex Reasoning in Knowledge Graph Question Answering through Query Graph Approximation

Hongjun Jeong[*1], **Minji Kim**[*1], **Heesoo Jung**[1], **Ko Keun Kim**[†2], **Hogun Park**[†1]

[1]Sungkyunkwan University, Republic of Korea
[2]LG Electronics, Republic of Korea
{zun8861, kmjj7864, steve305}@skku.edu, kokeun.kim@lge.com, hogunpark@skku.edu

Abstract

Knowledge-grounded Question Answering (QA) aims to provide answers to structured queries or natural language questions by leveraging Knowledge Graphs (KGs). Existing approaches are mainly divided into Knowledge Graph Question Answering (KGQA) and Complex Query Answering (CQA). Both approaches have limitations: the first struggles to utilize KG context effectively when essential triplets related to the questions are missing in the given KGs, while the second depends on structured first-order logic queries. To overcome these limitations, we propose a novel framework termed Aqua-QA. Aqua-QA approximates query graphs from natural language questions, enabling reasoning over KGs. We evaluate Aqua-QA on challenging QA tasks where KGs are incomplete in the context of QA, and complex logical reasoning is required to answer natural language questions. Experimental results on these datasets demonstrate that Aqua-QA outperforms existing methods, showcasing its effectiveness in handling complex reasoning tasks in knowledge-grounded QA settings.

1 Introduction

Knowledge Graphs (KGs) are structured representations of factual knowledge, typically encoded as triples in the form of (head, relation, tail), known as facts (Ji et al., 2021). A prominent set of applications leveraging KGs includes answering structured queries and answering natural language questions, both grounded in the KGs. To achieve this, graph query languages such as SPARQL are interpreted and executed by a query engine, which traverses the KG based on the given structured query to retrieve the answers.

However, in this context, the incompleteness of KGs (Chen et al., 2020) goes beyond missing facts to include the absence of essential facts required for the query engine to perform a traversal. Thus, simply traversing the KGs often fails to retrieve answers or misses important ones (Ren et al., 2024), although answers can still be indirectly inferred from existing facts within the KGs.

To reason answers that can be logically derived from incomplete KGs, query embedding methods (Ren et al., 2020; Ren and Leskovec, 2020; Zhang et al., 2021; Xu et al., 2023; Kim et al., 2024) have been proposed. These methods encode query graphs—constructed from First-Order Logic (FOL) queries—by parameterizing entities, relations, and logical operators such as conjunction, disjunction, and negation. As they distill the core semantics of the questions into a structured form, query graphs can be seen as self-contained representations, compactly capturing structural semantics and reasoning intent (Yih et al., 2015). Effectively encoding them helps mitigate KG incompleteness by leveraging available facts without relying heavily on connectivity. However, query embedding methods require well-defined FOL inputs despite the challenges of converting natural language into FOL (Yang et al., 2024). As a result, these models struggle to reason when a natural language question is poorly translated into an FOL query.

In contrast, Knowledge Graph Question Answering (KGQA) aims to directly answer natural language questions by reasoning over KGs. From the widely used benchmarks in KGQA, WebQSP (Yih et al., 2016), and CWQ (Talmor and Berant, 2018), ensure that all essential facts required to answer each question are present within the corresponding KG. Namely, KGs are complete in the context of QA. In addition, such datasets do not consider negation operations, the identification of which is critical to logical reasoning (Varshney et al., 2025). These assumptions have shaped KGQA approaches toward retrieving supportive information (e.g., triples (Baek et al., 2023), paths (Luo et al., 2024), or subgraphs (He et al., 2025; Ding

[*] Equal contribution.
[†] Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2025, pages 27038–27056

et al., 2024)) to aid in answering questions.

However, similar to graph query languages, the retrieval approaches require traversal over the KGs, which suffer from the aforementioned limitation due to incompleteness. Furthermore, when questions involve multiple logical operators, especially the negation operation, retrieving a larger portion of the KGs becomes necessary to capture indirect paths or extended multi-hop connections. This leads to the inclusion of noisy information and an intractably large volume of retrieved data, undermining the key advantage of retrieval approaches, which reduces the reasoning space.

Additionally, when natural language questions do not contain explicit mentions that directly correspond to entities or relations in the KGs, it becomes challenging to identify the topic entity—the starting point of retrieval—or to translate the question into FOL. This increases the likelihood of retrieving irrelevant information or an FOL query that does not capture the intent of the question, which impairs the reasoning accuracy.

To address the above problems, we propose Aqua-QA, a novel approach that leverages the contextual information in natural language questions to **A**pproximate **QU**ery gr**A**phs, bypassing the traversal of KGs for KG**QA**. For the approximation of query graphs, we introduce three key components: Align module, Query decomposition module, and Relation extraction module. Align module extracts both KG information, such as entity and relation, and structural information, including logical operators and question type, in the form of embeddings. Query decomposition module decomposes complex questions into sub-questions, serving as anchor or variable nodes that represent unknown entities in the intermediate reasoning steps within the query graph. Relation extraction module extracts relations that constitute the question as discrete tokens. Finally, the extracted information is combined to construct a sequence that serves as a query graph, and Reasoner performs prediction using a pretrained Knowledge Graph Embedding (KGE) method based on the constructed sequence. To capture negation nuances, we introduce an additional layer and special token for negated relations. Extensive experiments on three datasets demonstrate that our model consistently outperforms baselines on complex questions with multiple logical operators and missing facts.

2 Related Work

2.1 KGQA

KGQA aims to retrieve answers to natural language questions over KGs. Existing methods can be broadly divided into Semantic-Parsing (SP) and Information-Retrieval (IR) approaches. SP approach (Li et al., 2023) converts natural language questions into executable queries (e.g., SPARQL) to retrieve answers. In contrast, the IR approach extracts relevant information related to questions such as triples (Baek et al., 2023), paths (Zhang et al., 2022; Luo et al., 2024), and subgraphs (He et al., 2025; Ding et al., 2024) based on the topic entities. In addition, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), which enhances the quality of answers generated by Large Language Models (LLMs) by retrieving relevant information from external knowledge sources, can also be utilized for the KG-grounded QA. Recently, a method has been proposed to construct KGs from textual documents (Edge et al., 2024). However, these SP and IR approaches overlook QA scenarios where the required facts are absent from the KGs, which may lead to reasoning failure, either by missing important answers even when the question is successfully converted into an executable query, or by failing to retrieve relevant evidence. Furthermore, as a question becomes more complex with multiple logical operators, it inherently requires retrieving answers that are further away, thereby necessitating overburdened retrieval.

2.2 Handling missing facts in KGQA

There are existing methods that aim to answer questions on KGs with missing facts (Saxena et al., 2020; Xu et al., 2024). Both methods attempt to address this problem by generating new links from topic entities. However, these models are either incapable of handling complex questions with multiple topic entities or remain underexplored in addressing questions involving negation, which require generating an extremely large number of new links. In contrast, Aqua-QA does not require a topic entity and can handle questions with negation explicitly via Align module and Relation extraction module.

2.3 Complex query answering

Complex Query Answering (CQA) aims to reason the answer from incomplete KGs for complex FOL queries consisting of multiple logical opera-

tors (Ren et al., 2024). Previous work has encoded logical queries into a low-dimensional vector space using geometric-based embeddings (Ren et al., 2020; Zhang et al., 2021), probability-based embeddings (Ren and Leskovec, 2020), positional embeddings (Kim et al., 2024), or pretrained KGE (Xu et al., 2023). However, these methods cannot solely address natural language questions because the entities, relations, and logical operators that make up the question must be explicitly parsed. Therefore, if the conversion of a natural language question into an FOL query fails to fully capture its original intent, a CQA model struggles to make reliable and robust predictions.

3 Preliminary

KG Let a KG be $\mathcal{G} = \{\mathcal{V}, \mathcal{R}, \mathcal{E}\}$, where $\mathcal{V}$ is set of entities, $\mathcal{R}$ is set of relations, and $\mathcal{E}$ is set of facts defined as $\{(u, r, v) | \, u, v \in \mathcal{V}, r \in \mathcal{R}\}$.

KGQA Let $\mathcal{A}$ be the set of answer entities and $\mathcal{T}$ be the set of topic entities. KGQA aims to find the answer entity $a \in \mathcal{A}$ based on $\mathcal{G}$, given a natural language question and topic entity $t \in \mathcal{T}$.

FOL Following the definition from BetaE (Ren and Leskovec, 2020), we define an FOL query q_{FOL} in its Disjunctive Normal Form (DNF) as follows:

$$q_{\text{FOL}}[V_?] = V_?, \; \exists V_1, ..., V_k : c_1 \vee c_2 \vee ... \vee c_n, \quad (1)$$

where $V_1, ..., V_k$ are bound variables and $V_?$ is a single target free variable. $c_1, ..., c_n$ denote clauses consisting of one or more atomic formulas or their negation $e_{ij} = r(x, y)$ or $\neg r(x, y)$, called literal, using conjunctive operator like $c_i = e_{i1} \wedge e_{i2} \wedge ... \wedge e_{im}$. Variables x and y can be either constants or variables. A relation r serves as a binary function indicating whether x and y can be connected via r, considering direction as follows:

$$r(x, y) = \begin{cases} true, & if \quad (x, r, y) \in \mathcal{E}, \\ false, & otherwise. \end{cases} \quad (2)$$

3.1 Problem Definition

Let a KG be $\mathcal{G} = \{\mathcal{V}, \mathcal{R}, \mathcal{E}_{\text{train}}\}$ and the set of unseen ground-truth facts be $\mathcal{E}_{\text{test}}$ which satisfies following condition: $\mathcal{E}_{\text{train}} \cap \mathcal{E}_{\text{test}} = \emptyset$, $\mathcal{E}_{\text{test}} = \{(u, r, v) | \, u, v \in \mathcal{V}, r \in \mathcal{R}\}$. The problem addressed in this paper is the task of finding answer entities given only $\mathcal{G}$ and a complex natural language question, which may involve multiple logical operators such as conjunction, disjunction, and negation. Assuming the existence of an FOL query, which fully captures the intent of the natural language question, every literal comprising this FOL query becomes true in Equation (2), where $\mathcal{E} = \mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{test}}$. In this case, some of the facts, that make literal true, may not exist in $\mathcal{E}_{\text{train}}$, but are present in $\mathcal{E}_{\text{test}}$.

3.2 Difference from prior setting

Let us assume that for every natural language question, the literals that constitute its corresponding FOL query hold true over the $\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{test}}$. This paper aims at a scenario where answers must be derived based on KG, $\mathcal{G} = \{\mathcal{V}, \mathcal{R}, \mathcal{E}_{\text{train}}\}$, while conventional KGQA relies on KG, $\mathcal{G} = \{\mathcal{V}, \mathcal{R}, \mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{test}}\}$, where KG is complete to find the answer in the context of QA. Additionally, we address natural language questions composed of combinations of various logical operators, including negation, a logical operator that has been underexplored in prior work.

4 Methodology

In this section, we introduce Aqua-QA, which leverages contextual information of natural language to approximate a query graph and perform reasoning based on it. Our methodology consists of 1) an Align module, 2) a Query decomposition module, and 3) a Relation extraction module, which effectively approximate a structured query from a given question. Moreover, we propose 4) a Reasoner that leverages a pretrained KGE method to infer the answer entity using the outputs from components 1), 2), and 3). The overall framework is illustrated in Figure 1, and the illustration of the Align module is in Figure 6.

4.1 Align module

Align module aims to decouple the structural information and the semantic information related to KG from natural language questions. To achieve this, we define six learnable tokens ([ENT], [REL], [TYPE], [AND], [OR], [NOT]):

$$\mathbf{E} = [\mathbf{e}_{\text{ENT}}; \mathbf{e}_{\text{REL}}; \mathbf{e}_{\text{TYPE}}; \mathbf{e}_{\text{AND}}; \mathbf{e}_{\text{OR}}; \mathbf{e}_{\text{NOT}}], \quad (3)$$

where $\mathbf{e}$ is the embedding of the token indicated by the subscript and $\mathbf{E}$ indicates the concatenated embeddings of all pre-defined tokens. [ENT] and [REL] tokens identify KG entities and relations associated with input natural language questions and align these natural language tokens into the embedding space of the KG, leveraging pretrained KGE

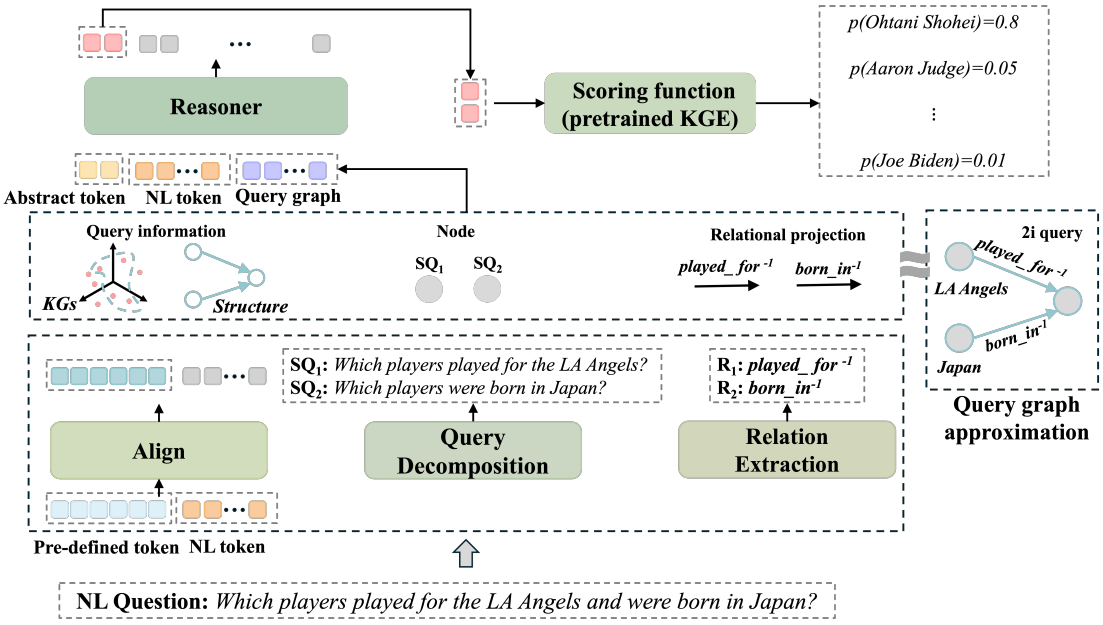

Figure 1: Overall framework of Aqua-QA.

methods such as ComplEx (Trouillon et al., 2016). [AND], [OR], and [NOT] tokens capture the structural information by identifying the existence of conjunction, disjunction, and negation operators, respectively. [TYPE] token identifies the type of questions such as single-hop or multi-hop to facilitate sharing of reasoning processes and structural relationships. We exploit 11 distinct types, shown in Figure 5, as labels.

Pretraining the Align module requires paired data consisting of natural language questions and their corresponding FOL queries. However, such data is unavailable in our setting, as we assume access only to natural language questions, without ground-truth mappings to FOL queries. Therefore, we generate pretraining data from the given KGs to obtain a self-supervision signal. In short, we construct FOL queries by randomly selecting facts from the KGs and then convert them into natural language using an LLM. The detailed process of data synthesis can be found in Appendix E. Notably, we augment data by replacing some entities with semantically similar but distinct expressions, enabling effective alignment even when the entities mentioned in the question are not explicitly present in the KGs (e.g., "Aaron Judge": "2024 American League MVP and the captain of New York Yankees").

For the pretraining of Align module, we first encode natural language questions using a Pretrained Language Model (PLM) (e.g., SBERT (Reimers and Gurevych, 2019)) and concatenate the encoded sequence with pre-defined tokens. This sequence is processed to f_{align} which consists of transformer encoder layers (Vaswani, 2017) as follows:

$$\mathbf{H}^{M}_{\text{align}} = f_{\text{align}}(\mathbf{E}; PLM(q)), \tag{4}$$

where q is a natural language question and $\mathbf{H}^{M}_{\text{align}} \in \mathbb{R}^{N_{\text{align}} \times d_{\text{KG}}}$ is the last hidden states of concatenated sequence, and N_{align} is the length of concatenated sequence. d_{KG} denotes the dimension of KGE. The [ENT] token and the [REL] token are trained with contrastive learning using the embeddings of entities (or relations) represented in natural language questions as the positive samples and randomly sampled embeddings of the other entities (or relations) as the negative samples. Both positive and negative entity embeddings are taken from the pretrained KGE.

The [TYPE] token is optimized similar to [ENT] and [REL]. We sample [TYPE] tokens of a different question type in a batch to use them as negative samples. The contrastive learning loss for [ENT], [REL], and [TYPE] tokens is expressed as:

$$\mathcal{L}_{\text{CL}} = -\sum_{i=1}^{3} \log \frac{\exp(\mathbf{h}_i^M \cdot \mathbf{e}_i^+)}{\exp(\mathbf{h}_i^M \cdot \mathbf{e}_i^+) + \sum_{j=1}^{|\mathcal{N}_i|} \exp(\mathbf{h}_i^M \cdot \mathbf{e}_{ij}^-)}.$$

Here, $\mathbf{h}_i^M$ is the i-th hidden representation of $\mathbf{H}^{M}_{\text{align}}$, where i corresponds to [ENT], [REL], and [TYPE]. $|\mathcal{N}_i|$ denotes the cardinality of negative sample set $\mathcal{N}_i$, and $\mathbf{e}_i^+$, $\mathbf{e}_{ij}^- \in \mathbb{R}^{d_{\text{KG}}}$ is positive

sample and j-th negative sample for i-th token, respectively.

In addition, to distinguish representations of relation with and without the negation operation, we introduce a linear layer represented as $\mathbf{W}_{\text{neg},1} \in \mathbb{R}^{d_{\text{KG}} \times d_{\text{KG}}}$ as:

$$\mathbf{e}^{+(-)}_{[\text{REL}]} = \begin{cases} \mathbf{W}_{\text{neg},1}\mathbf{e}^{+(-)}_{[\text{REL}]}, & \text{if negation relation } \neg, \\ \mathbf{e}^{+(-)}_{[\text{REL}]}, & \text{otherwise.} \end{cases}$$

The [AND], [OR], and [NOT] tokens are optimized using binary cross-entropy loss functions based on whether the question contains a conjunction, disjunction, or negation operation. For instance, the example illustrated in Figure 1 includes a conjunction operator, where the label for [AND] is assigned a value of 1, while the labels for other operators, such as [OR] and [NOT], are set to 0. Thus, the loss function for a single data pair can be expressed as follows:

$$\mathcal{L}_{\text{BCE}} = -\sum_{k=1}^{3} y_k \log \hat{y_k} + (1 - y_k) \log(1 - \hat{y_k}),$$

where k corresponds to [AND], [OR], and [NOT], $\hat{y_k}$ is the prediction from the k-th representation in $\mathbf{H}^M_{\text{align}}$ through task-specific heads, and y_k is a label of each token. Finally, Align module is optimized with the following loss function:

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{CL}} + \mathcal{L}_{\text{BCE}}. \tag{5}$$

4.2 Query decomposition module

For the modeling of anchor or variable nodes within the approximated query graphs, we decompose a complex question into simpler sub-questions using a vanilla LLM with few-shot demonstrations and without any fine-tuning. Similar to Plan-and-Solve prompting (Wang et al., 2023), which generates an intermediate process to tackle complex tasks, query decomposition helps clarify intermediate reasoning steps, which are analogous to the roles of anchor and variable nodes in the query graph.

4.3 Relation extraction module

To extract the relations that constitute the natural language question, we present an instruction-tuning task for the LLM by leveraging the synthetic data used to train Align module. Inspired by the RoG (Luo et al., 2024), we format the output of the LLM with special token <START>, <SEP>, and <END>. Each token represents the start, separator, and end of the relation extraction, respectively. Additionally, to capture the semantic differences in relations caused by negation, we add a special token <NEG>. For example, let q_{NL} be *"Which players played for the LA Angels and were not born in Japan?"*. Then, LLM f_{rel} is instruction-tuned to extract relations for input question q as follows:

$$\begin{aligned} f_{\text{rel}}(q) = &\text{<START>played\_for}^{-1}\text{<SEP>} \\ &\text{<NEG>born\_in}^{-1}\text{<END>}. \end{aligned} \tag{6}$$

Here, the <NEG> token serves to indicate whether a negation is applied to a relation. The extracted relations serve as a relational projection of anchor and variable nodes in the approximated query graph.

4.4 Reasoner

The reasoner aims to predict answer candidate entities by leveraging the approximated query graph. The parameters of all modules except the reasoner are frozen when training the reasoner. Given a natural language question q as input, an approximated query graph is generated from the three frozen modules. For the Align module, we exploit the last hidden states of the pre-defined tokens $\mathbf{H}^M_{\text{align}}[:6] \in \mathbb{R}^{6 \times d_{\text{KG}}}$. In the Query decomposition module, we decompose q into sub-questions and encode each of them through a PLM. Then, we get initial sub-question representations as $\mathbf{H}_{\text{sub}} \in \mathbb{R}^{N_{\text{SUB}} \times d_{\text{KG}}}$ where N_{SUB} denotes the number of sub-questions by pooling each sub-question representation and applying the linear transformation to match the dimension size with $\mathbf{H}^M_{\text{align}}$. Through the Relation extraction module, we extract the relations in q and get the corresponding embeddings $\mathbf{H}_{\text{rel}} \in \mathbb{R}^{N_{\text{REL}} \times d_{\text{KG}}}$ from the pretrained KGE where N_{REL} is the number of extracted relations. In this case, we define a negation layer $\mathbf{W}_{\text{neg},2} \in \mathbb{R}^{d_{\text{KG}} \times d_{\text{KG}}}$ to distinguish the semantic of relation embedding with or without negation since the Relation extraction module identifies the negation operator with the special token <NEG>. Finally, by concatenating all extracted embeddings $\mathbf{H}^M_{\text{align}}$, $\mathbf{H}_{\text{sub}}$, and $\mathbf{H}_{\text{rel}}$ into a single sequence, the approximated query graph is represented as $\mathbf{H}_{\text{FOL}} \in \mathbb{R}^{(6+N_{\text{SUB}}+N_{\text{REL}}) \times d_{\text{KG}}}$. To fully utilize the context of the question, we also encode a natural language question through PLM, and then apply a linear transformation again for the same purpose with $\mathbf{H}_{\text{sub}}$. This representation is denoted as $\mathbf{H}_{\text{NL}} \in \mathbb{R}^{N_{\text{NL}} \times d_{\text{KG}}}$ where N_{NL} is the length of q.

Following Q2T (Xu et al., 2023), our reasoner

consists of transformer encoder layers and learnable embeddings $\mathbf{e}_{\text{head}}, \mathbf{e}_{\text{rel}} \in \mathbb{R}^{d_{\text{KG}}}$, which serve as an abstract head and relation. The input of the reasoner is represented as follows:

$$\mathbf{H}^0 = [\mathbf{e}_{\text{head}}; \mathbf{e}_{\text{rel}}; \mathbf{H}_{\text{FOL}}; \mathbf{H}_{\text{NL}}]. \quad (7)$$

Then we process $\mathbf{H}^0$ into the reasoner f_{r} as follows:

$$\mathbf{H}^L = f_{\text{r}}(\mathbf{W}_{\text{r}}\mathbf{H}^0), \quad (8)$$

where $\mathbf{H}^L$ is the last hidden representation of the input sequence through the reasoner and $\mathbf{W}_{\text{r}} \in \mathbb{R}^{d \times d_{\text{KG}}}$ is the linear transformation to reduce the dimension size for efficient training. Finally, we compute the score of the answer candidate entity e_{a} to the q by using the pretrained KGE as a scoring function ϕ as follows:

$$\text{s}(q, e_{\text{a}}) = \phi(\mathbf{W}_{\text{ent}}\mathbf{h}^L_{\text{head}}, \mathbf{W}_{\text{rel}}\mathbf{h}^L_{\text{rel}}, \mathbf{e}_{\text{a}}). \quad (9)$$

Here, $\mathbf{h}^L_{\text{head}}$ and $\mathbf{h}^L_{\text{rel}}$ are the representations of abstract head and relation in $\mathbf{H}^L$, respectively, and $\mathbf{e}_{\text{a}}$ is the embedding of answer candidate entity. $\mathbf{W}_{\text{ent}}, \mathbf{W}_{\text{rel}} \in \mathbb{R}^{d^{\text{KG}} \times d}$ are the linear transformations to project the $\mathbf{h}^L_{\text{head}}$, $\mathbf{h}^L_{\text{rel}}$ to KG embedding space for scoring. The loss function used to train the reasoner is as follows:

$$\mathcal{L}_{\text{r}} = -\log \frac{\exp\left(\text{s}(q, e^+_{\text{a}})\right)}{\exp\left(\text{s}(q, e^+_{\text{a}})\right) + \sum_{i=1}^{|\mathcal{N}|} \exp\left(\text{s}(q, e^-_{\text{a,i}})\right)}, \quad (10)$$

where e^+_{a} denotes the answer entity for a given question and $e^-_{\text{a,i}}$ is a i-th negative sample randomly selected from the non-correct entities.

5 Experiments

In this section, we empirically validate the effectiveness of Aqua-QA, answering the following research questions. **RQ1**: Can Aqua-QA achieve better QA performance than baselines across multiple datasets? **RQ2**: How does each module in Aqua-QA affect performance? **RQ3**: Can Aqua-QA effectively deal with the challenges as follows: 1) incompleteness of KGs, where essential facts for answering questions may be missing, 2) complexity of questions with multiple logical operators, 3) ambiguity in questions that lack terms directly corresponding to entities in KGs?

5.1 Experimental setup

Dataset As discussed in Section 3, existing KGQA benchmarks are not well suited for evaluating whether Aqua-QA effectively addresses the limitations outlined in this paper. Therefore, we construct three new datasets based on CQA's data generation strategy, as it aligns with the objectives discussed in Section 3. In detail, natural language question answering datasets are generated from the widely used KGs (FB15k-237 (Toutanova and Chen, 2015), UMLS (Kok and Domingos, 2007), and CoDEx (Safavi and Koutra, 2020)). We employ the query generation process proposed in BetaE (Ren and Leskovec, 2020) to derive logical queries from the KGs. Then, the generated FOL queries are converted into natural language questions using Llama-3-70B-Instruct. In addition, the question types for all datasets follow the 14 query types defined in BetaE, as detailed in Appendix 5.

Metric We employ Hits@1, which measures whether the model's top-1 prediction matches the correct answer, as the evaluation metric. For LLM-based baselines, we treat the first answer generated by the model as the top-1 prediction.

Baseline Baselines can be broadly categorized into the IR approach, LLM, and CQA. The model for each category is as follows: 1) IR approach: KAPING (Baek et al., 2023), RAG (Lewis et al., 2020), GraphRAG (Edge et al., 2024), RoG (Luo et al., 2024), G-Retriever (He et al., 2025), EPR (Ding et al., 2024). 2) LLM: Llama-3-8B-Instruct (Dubey et al., 2024). 3) CQA: BetaE (Ren and Leskovec, 2020), ConE (Zhang et al., 2021), Q2T (Xu et al., 2023). The details of the baselines are provided in the Appendix F.

Implementation Relation extraction module uses FLAN-T5-xl (Raffel et al., 2020) and Query decomposition module uses Llama-3-8B-Instruct as the backbone model. If not explicitly specified, we do not apply the question augmentation strategy that replaces entities with semantically similar variants in pretraining data synthesis for fair comparison. We adopt ComplEx (Trouillon et al., 2016) as our backbone KGE. Every baseline using LLMs employs Llama-3-8B-Instruct as the backbone model to ensure that there is no performance difference depending on the backbone model. We adopt LogicLLaMA (Yang et al., 2024) to convert natural language questions into FOL queries (NL2FOL), which are used to generate input for the CQA models. Experiments are conducted on the A100 GPU. More detailed implementation settings are in Appendix D.

Dataset	Model	NL	w/o R	w/o T	1p	2p	3p	2i	3i	pi	ip	2u	up	2in	3in	inp	pin	pni	Ap	An
FB15k-237	KAPING	○	×	×	33.8	19.6	17.7	32.6	27.5	21.4	16.5	57.0	28.6	29.8	21.3	15.3	17.3	20.4	29.3	20.7
	RAG	○	×	○	23.1	17.5	19.1	21.3	13.8	14.5	10.7	44.3	22.6	34.2	18.8	16.8	18.1	26.0	21.2	22.6
	GraphRAG	○	×	○	13.0	5.8	5.9	5.1	4.5	3.0	5.4	9.8	7.4	5.3	3.3	5.7	4.7	3.9	7.8	4.6
	RoG	○	×	×	20.1	26.4	23.2	13.0	10.7	11.8	16.7	32.2	27.0	16.7	10.1	17.6	19.2	9.6	20.2	14.6
	G-Retriever	○	×	○	26.3	17.2	16.4	22.2	18.3	17.5	14.8	45.4	22.6	24.6	15.9	13.7	16.8	18.1	23.1	17.7
	EPR	○	×	○	36.1	14.2	12.5	14.5	7.6	10.2	5.0	19.7	6.7	17.4	8.1	7.4	7.8	12.1	14.1	10.1
	Llama-3-8B-Inst.	○	×	○	20.0	10.3	8.0	14.5	11.2	9.9	7.8	27.8	11.9	13.5	8.1	8.5	8.7	8.4	14.7	9.4
	BetaE	×	○	×	36.2	36.3	39.6	44.0	39.9	19.5	13.6	53.8	17.0	36.8	34.5	35.1	21.6	16.6	33.6	28.7
	ConE	×	○	×	36.7	40.4	42.4	52.0	49.4	23.2	15.9	68.5	20.4	49.0	42.1	38.0	25.7	18.1	38.1	34.2
	Q2T	×	○	×	39.7	50.6	47.9	60.1	58.5	28.3	17.3	71.4	21.4	52.6	47.7	35.2	28.2	30.3	42.7	38.8
	Ours	○	○	○	**72.0**	**74.3**	**69.1**	**75.5**	**70.1**	**45.1**	**31.4**	**92.6**	**47.7**	**62.3**	**54.8**	**59.2**	**47.0**	**43.9**	**65.2**	**53.2**
UMLS	KAPING	○	×	×	29.3	20.6	26.2	7.3	11.2	20.8	45.7	51.5	41.0	20.6	10.2	40.2	40.7	21.6	29.0	26.5
	RAG	○	×	○	25.5	25.7	26.7	10.3	11.7	18.8	33.2	42.5	39.0	19.4	17.9	35.2	26.3	21.6	26.1	24.0
	GraphRAG	○	×	○	10.8	10.3	7.5	7.3	7.1	8.6	14.1	34.5	20.0	13.3	7.1	15.1	8.2	12.9	13.1	11.2
	RoG	○	×	×	32.0	33.1	24.6	24.2	18.4	29.4	40.7	44.0	34.5	40.0	13.8	35.8	33.0	36.1	31.5	31.5
	G-Retriever	○	×	○	31.5	30.9	26.2	28.5	16.3	31.5	42.2	59.5	48.0	37.8	32.1	39.1	42.3	46.4	34.6	39.6
	EPR	○	×	○	59.2	33.8	32.6	31.5	23.0	24.4	31.2	34.0	17.5	39.4	24.0	38.0	32.5	39.7	31.9	34.7
	Llama-3-8B-Inst.	○	×	○	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0	1.0	1.0	0.5	0.5	0.3	0.4
	BetaE	×	○	×	66.8	**75.7**	**81.3**	69.1	**65.3**	57.9	52.8	84.0	42.0	71.7	62.2	61.5	53.6	61.9	66.1	62.0
	ConE	×	○	×	71.5	69.9	71.7	68.5	53.6	50.3	55.8	96.0	46.5	75.0	63.8	71.5	38.1	49.0	65.9	59.1
	Q2T	×	○	×	67.6	66.9	79.1	**73.9**	62.2	52.8	59.8	80.0	41.0	73.9	64.3	67.0	67.5	68.6	65.1	68.2
	Ours	○	○	○	**80.4**	72.1	75.4	68.5	63.8	**66.5**	**62.3**	**97.0**	**58.0**	**81.1**	**64.8**	**76.0**	**73.2**	**72.2**	**73.3**	**73.3**
CoDEx	KAPING	○	×	×	60.3	21.2	16.6	48.8	34.1	20.7	13.0	64.4	17.9	41.3	32.9	23.5	15.9	20.6	31.5	26.6
	RAG	○	×	○	41.0	16.6	16.6	25.2	18.3	18.3	9.7	50.8	16.4	49.4	26.0	25.1	23.6	30.4	22.8	30.6
	GraphRAG	○	×	○	27.9	5.3	3.2	13.8	9.0	8.8	4.7	17.6	5.8	13.7	6.9	11.1	5.1	5.6	9.7	8.4
	RoG	○	×	×	38.9	36.4	33.3	21.2	13.6	19.4	28.0	36.2	**28.6**	28.0	17.6	53.2	42.2	21.3	27.6	32.6
	G-Retriever	○	×	○	59.0	22.5	30.1	30.1	24.4	20.6	19.6	41.0	26.8	34.1	20.2	47.4	35.0	25.1	28.8	32.4
	EPR	○	×	○	53.9	19.4	19.7	9.2	5.9	10.7	4.3	9.5	2.4	25.5	11.9	9.9	16.9	14.8	15.0	15.8
	Llama-3-8B-Inst.	○	×	○	39.9	9.3	7.1	21.2	15.3	11.1	7.1	40.5	8.2	19.4	11.7	13.4	7.2	7.0	16.6	11.6
	BetaE	×	○	×	57.6	52.7	59.7	58.1	56.6	32.2	9.4	74.1	8.2	55.1	50.8	49.2	32.7	21.8	44.1	41.8
	ConE	×	○	×	64.4	57.6	68.5	69.2	68.0	35.1	11.7	84.1	8.0	68.0	67.5	54.8	36.4	17.5	50.4	48.6
	Q2T	×	○	×	63.8	57.8	69.0	74.4	71.3	39.1	10.5	74.4	7.4	78.5	**76.5**	42.2	28.7	24.3	50.5	49.6
	Ours	○	○	○	**81.5**	**72.0**	**78.3**	**74.6**	**74.9**	**58.4**	**34.0**	**88.3**	15.1	**81.9**	73.1	**88.3**	**89.8**	**86.2**	**62.5**	**83.9**

Table 1: Performance comparison in Hits@1 across baseline models. NL, w/oR, and w/oT indicate a constraint-free setting, allowing Natural Language input without requiring KG-based retrieval or topic entities. **Bold** scores mark the best results, while underlined scores denote the second-best. Ap and An represent the average performance on questions with or without negations, respectively.

5.2 Main results (RQ1)

We compare our framework against baselines on three datasets to evaluate its reasoning capability. As shown in Table 1, our framework significantly outperforms baselines on all datasets by effectively reasoning the answers through query graph approximation. In particular, IR approaches underperform compared to ours and CQA models, which do not require explicit traversal over the KG. These results support our claim that retrieving relevant and compact information is challenging when the KG is incomplete and the questions require multi-hop reasoning involving multiple logical operators. Furthermore, both IR approaches and CQA models need the identification of topic entity or converting NL2FOL. Since these processes propagate additional errors, it results in performance degradation.

5.3 Ablation study (RQ2)

We conduct an ablation study by removing each module (Align, Query decomposition, and Relation extraction module) to evaluate its impact on overall performance. As shown in Table 2, both Align and Query decomposition modules contribute to performance improvements for questions with and without negation. Although Relation extraction module results in a performance drop for questions without negation in FB15k-237, it shows a significant performance improvement for questions with negation in both datasets. We assume that this result is due to the negation layer, which is inherently dependent on Relation extraction module and cannot function without it. In addition, in UMLS, Relation extraction module contributes considerably to A_{p} performance compared to other modules. It still supports the effectiveness of Relation extraction module. Overall, our approach of approximating

	FB15k-237		UMLS	
	Ap	An	Ap	An
w/o align	64.42	52.48	73.08	70.84
w/o QD	64.56	52.11	72.84	71.47
w/o rel	**66.34**	49.54	67.73	52.28
Aqua-QA	65.22	**53.24**	**73.28**	**73.28**

Table 2: Ablation study of Aqua-QA for FB15k-237 and UMLS on Hits@1. Align, QD, and rel denote Align, Query decomposition, and Relation extraction module, respectively. w/o denotes our framework without the indicated module.

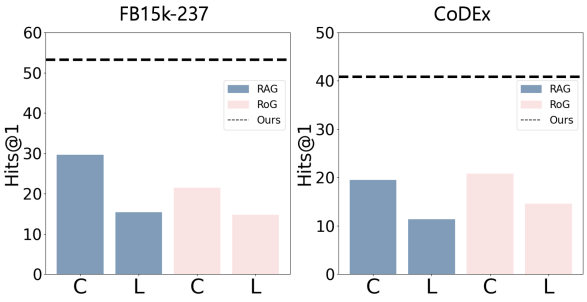

Figure 2: Hits@1 performance on questions where every answer is classified as a hard answer. Here, C denotes the complete facts setting, while L represents the limited facts setting.

the query graph with every module demonstrates its effectiveness.

5.4 Model analysis (RQ3)

	Conjunction	Disjunction	Negation
FB15k-237	98.70	93.86	98.67
CoDEx	99.83	99.62	99.99
UMLS	99.41	98.69	99.99

Table 3: Classification performance of Align module on identifying logical operators.

Effectiveness under incompleteness KGs We design an additional experiment to compare the performance on questions that have only hard answers, which cannot be inferred solely from the facts present in the given KG. As illustrated in Figure 2, we observe that IR approaches significantly underperform when evaluated with hard answers, compared to the performance in the complete facts setting. In contrast, our framework, which takes an approach that approximates the query graph, effectively reasons answers by mitigating incompleteness due to missing facts.

Handling complex question type To verify whether Align module can identify logical operations from natural language questions, we evaluate the binary classification performance of Align module. We use AUC-PR as an evaluation metric to consider the imbalance in the occurrence of each logical operator. Based on the results shown in Table 3, Align module is highly effective at identifying which logical operations are embedded. Additionally, we visualize the embedding of [TYPE] tokens with respect to types of questions. As shown in Figure 3, our model effectively clusters question types in the embedding space, with each cluster

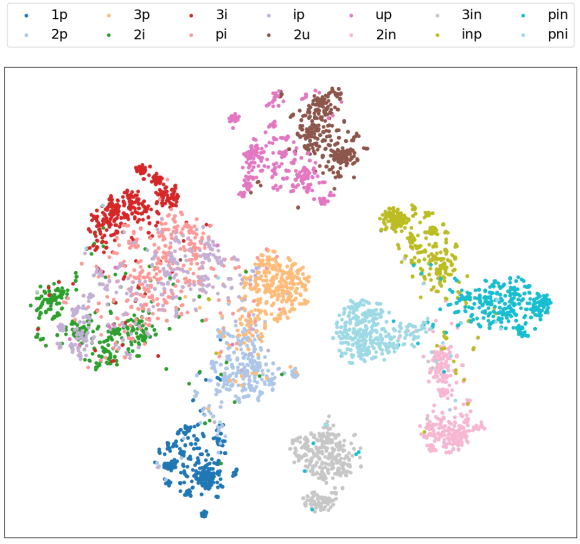

Figure 3: t-SNE visualization of [TYPE] tokens in the CoDEx dataset, using 400 samples per question type.

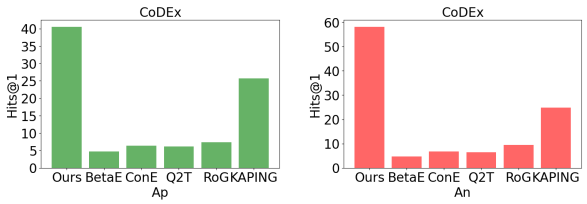

Figure 4: Performance with the question where entities in KGs are not explicitly represented in the CoDEx.

representing a specific type. Furthermore, similar question types are positioned near each other, indicating their semantic relationships.

Handling questions with implicitly represented KG entities. We conduct additional experiments to assess the robustness of our framework when a pretraining data augmentation strategy is applied, by replacing entities in test questions with semantically similar alternatives, while excluding alternative data during training. From the results in Figure 4 for the CoDEx dataset, our model outperforms baselines. Notably, although NL2FOL uses the same pretraining data as our model to ensure a fair comparison, the CQA baselines exhibit significantly lower performance. Furthermore, in IR approaches, the reasoner also must account for expression inconsistencies between the question and the information retrieved from the KGs. In contrast, our framework adopts a modular design, which ensures a clear separation of the sub-tasks handled by each component, such as sentence structure analysis, KG information alignment (e.g., entity, rela-

	Case Study 1	Case Study 2
Question	Who are the spouses of Whitney Houston that were born in Boston and are citizens of the United States of America?	Which individuals from North American federal republic, who speak a language spoken in the UK, were educated at a performing arts conservatory in New York?
KG Entities represented in a question	Whitney Houston: Whitney Houston, United States of America: USA, Boston: Boston	North American federal republic: United States of America, language spoken in the UK: English, performing arts conservatory in New York: Juilliard School
Top-3 entities with **<ENT>** token	USA, Whitney Houston, Boston	USA, English, Juilliard School
Top-3 relations with **<REL>** token	birthplace of, citizen of, spouse	country of citizenship, languages spoken, written, or signed, educated at
Structural information in question	And: True (1.00), Or: False (0.00), Not: False (0.00)	And: True (1.00), Or: False (0.00), Not: False (0.00)
Sub-questions	Who are the spouses of Whitney Houston?, Who was born in Boston?, Who are citizens of the United States of America?	What individuals from the United States were educated at a performing arts conservatory in New York?, What language is spoken in the UK?
Extracted relation	birthplace of, citizen of, spouse	country of citizenship, 'languages spoken, written, or signed', educated at
Top-1 prediction	Bobby Brown	Elmer Bernstein
Answer	**Bobby Brown**	**Elmer Bernstein**

Table 4: Two case studies of approximating the query graph from example questions in the CoDEx dataset.

tion), and reasoning. Consequently, in complex questions that involve multiple factors, such tightly coupled methods often struggle to adapt flexibly.

Case study Two real-world case studies demonstrate the effectiveness of Aqua-QA in identifying the underlying context of natural language questions by approximating them as query graphs. Table 4 presents two examples from the CoDEx dataset, with the second illustrating a more challenging case where the entities represented in the KG are expressed differently in the natural language question. We observe that the approximated query graphs closely resemble the ground-truth query graphs for each question. These results suggest that our framework can approximate the query graph using only the context extracted from the natural language question.

6 Conclusion

We propose Aqua-QA, a framework that directly approximates query graphs from natural language questions without requiring structured inputs. This approach addresses limitations such as the incompleteness of KGs, where the information necessary to answer a given question is not explicitly provided within the KGs. It also handles the complexity inherent in natural language questions. Our experiments demonstrate the effectiveness of Aqua-QA in these extreme settings, highlighting its robustness in multiple datasets and its ability to capture the structural intent of complex questions. By outperforming existing methods, Aqua-QA advances knowledge-grounded QA, providing a flexible framework for complex reasoning.

7 Limitation

While Aqua-QA effectively handles complex reasoning over incomplete KGs, it has several limitations. First, its performance heavily depends on the quality and density of the underlying KGs. In addition, significant data gaps can hinder correct answer inference, especially for questions requiring extensive background knowledge not present in the KGs. Lastly, decomposing a complex question into sub-questions using Query decomposition module may not always preserve the original intent, particularly for questions with nested logical structures or multiple layers of complexity.

8 Ethics Statement

Our methodology, Aqua-QA, aims to improve knowledge-grounded question answering by approximating query graphs from natural language questions, enabling reasoning over incomplete KGs with complex logical operations. We do not utilize any external data beyond standard benchmark datasets and the provided KGs. However, since KGs may contain biases reflecting historical or societal prejudices, our approach may inadvertently propagate or amplify these biases in the answers generated. We acknowledge this limitation and recommend further research to identify and mitigate potential biases in knowledge graphs to ensure fair and unbiased outcomes in knowledge-grounded question-answering systems.

Acknowledgements

This work was supported by LG Electronics; in part by the Institute of Information & Communications Technology Planning & evaluation (IITP) grant and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2019-II190421, IITP-2025-RS-2020-II201821, RS-2024-00438686, RS-2024-00436936, IITP-2025-RS-2024-00360227, RS-2023-00225441, RS-2024-00448809). This research was also partially supported by the Culture, Sports, and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports, and Tourism in 2024 (RS-2024-00333068).

References

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data @ ACL 2023*, pages 70–98.

Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In *Proceedings of the ACM Web Conference 2024*, pages 2106–2115.

Cícero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 626–634.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graphrag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 553–561.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2025. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations*.

Xiang Huang, Sitao Cheng, Yiheng Shu, Yuheng Bao, and Yuzhong Qu. 2023. Question decomposition tree for answering complex questions over knowledge bases. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 12924–12932.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Jeonghoon Kim, Heesoo Jung, Hyeju Jang, and Hogun Park. 2024. Improving multi-hop logical reasoning in knowledge graphs with context-aware query representation learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15978–15991.

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

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jianbin Li, Ketong Qu, Jingchen Yan, Liting Zhou, and Long Cheng. 2021. Tebc-net: An effective relation

extraction approach for simple question answering over knowledge graphs. In *Knowledge Science, Engineering and Management*, pages 154–165.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 6966–6980.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597.

Xukai Liu, Ye Liu, Kai Zhang, Kehang Wang, Qi Liu, and Enhong Chen. 2024. OneNet: A fine-tuning free framework for few-shot entity linking via large language model prompting. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13634–13651.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *Proceedings of the International Conference on Learning Representations*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8864–8880.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992.

Hongyu Ren, Mikhail Galkin, Zhaocheng Zhu, Jure Leskovec, and Michael Cochez. 2024. Neural graph reasoning: A survey on complex logical query answering. *Transactions on Machine Learning Research*.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *Proceedings of the International Conference on Learning Representations*.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33:19716–19726.

Tara Safavi and Danai Koutra. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8328–8350.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 641–651.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2071–2080.

Neeraj Varshney, Satyam Raj, Venkatesh Mishra, Agneet Chatterjee, Amir Saeidi, Ritika Sarkar, and Chitta Baral. 2025. Investigating and addressing hallucinations of LLMs in tasks involving negation. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 580–598.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:6000–6010.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539.

Somin Wadhwa, Silvio Amir, and Byron C Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 15566–15589.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 2609–2634.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. 2024. Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18410–18430.

Yao Xu, Shizhu He, Cunguang Wang, Li Cai, Kang Liu, and Jun Zhao. 2023. Query2triple: Unified query encoding for answering diverse complex queries over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11369–11382.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 6942–6959.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 201–206.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 5773–5784.

Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:19172–19183.

A Code Availability

Our code is available at https://github.com/HongjunJeongSKKU/Aqua-QA.

B Further Related Work

Align & Relation Extraction Module Our Align module and Relation extraction module are conceptually aligned with prior work on entity linking and relation extraction, as they involve mapping questions to KG entities and relations. Early neural entity linking methods (Ganea and Hofmann, 2017; Kolitsas et al., 2018) leverage context-aware similarity scoring or perform joint detection of entity mentions and their corresponding KG entities in a unified architecture. Recent advances leverage Large Language Models (LLMs) to improve flexibility in entity linking, using prompt-based in-context learning (Liu et al., 2024). For the relation extraction task, conventional approaches treat it as a standalone classification problem, often employing CNN or LSTM encoders (dos Santos et al., 2015; Vu et al., 2016). Transformer-based model, such as TEBC-Net (Li et al., 2021), combines self-attention with BiLSTM and CNN layers, improving performance in simple KGQA benchmarks. More recently, generative approaches (Wadhwa et al., 2023) reformulate relation extraction as a sequence-to-sequence task, where models generate relations as textual sequences rather than selecting from a fixed set.

Unlike conventional approaches that match discrete elements in the KG, our Align module maps natural language questions into the continuous embedding space derived from KGE. It includes specialized tokens for not only entities and relations but also logical operators such as conjunction, disjunction, and negation, enabling it to model the structural logic of the question explicitly. In addition, the Relation extraction module extracts relations while detecting whether extracted relations are applied to negation, thereby providing more explicit guidance for downstream reasoning.

Query Decomposition Module Decomposing a complex question into simpler sub-questions helps the model better understand the structure of the question. Early work on question decomposition (Talmor and Berant, 2018; Min et al., 2019) splits the question into sub-questions depending on the reasoning type of the original question. To alleviate the burden of annotated data, ONUS (Perez

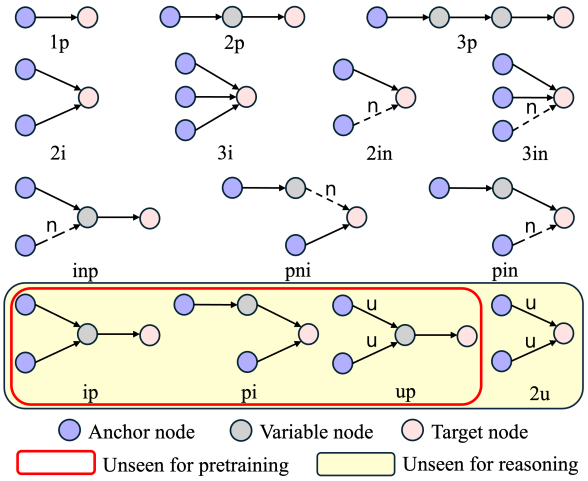

Figure 5: Query types addressed in the experiments. Unseen queries are types that are not used during the pretraining and reasoner training phase, respectively.

et al., 2020) generates a pseudo-decomposition set by mapping complex questions to the elements in a simpler question set based on similarity, and trains the model in an unsupervised manner using these sets. Recently, QDT (Huang et al., 2023) performs decomposition by generating preliminary sub-questions and inserting separators into the original question based on predefined options to enhance flexibility in response to the complexity of the question. While any query decomposition method can be used as our Query decomposition module, we adopt a vanilla LLM with few-shot demonstrations. It does not require any training cost and can handle diverse types of questions flexibly.

C Dataset Statistics

Tables 5 and 6 present the statistics for the KGs we use in our experiments—FB15k-237 (Toutanova and Chen, 2015), UMLS (Kok and Domingos, 2007), and CoDEx (Safavi and Koutra, 2020)—and the questions generated from each. Figure 5 shows all the grounded First-Order Logic (FOL) query types that we use for NL question generation. p, i, n, and u in the figure denote projection, intersection, negation, and union, respectively.

D Hyperparameters Setting

We apply a grid search to find the right hyperparameters for the reasoner, following the ranges of each hyperparameter. We set the learning rate in $[1e^{-4}, 2e^{-4}, 4e^{-4}]$, batch size in $[512, 1024]$, embedding dimension in $[768, 1024]$ and number of layers in $[6, 8]$. The optimal hyperparameter configurations for our model across different datasets are as follows. For FB15k-237, we set the learning rate to $4e^{-4}$, the batch size to 1024, and the embedding dimension to 1024. For UMLS, the learning rate is $4e^{-4}$, the batch size is 512, and the embedding dimension is 768. For CoDEx, the learning rate is $2e^{-4}$, the batch size is 1024, and the embedding dimension is 1024. Across all variants, the model consists of eight layers. The total number of parameters in our framework is summarized in Table 7.

E Pretrain Data Synthesis

For pretraining of Align module and Relation extraction module, we synthesize natural language question–FOL query pairs from KGs. First, FOL queries are generated by randomly selected facts in KGs and verbalized into natural language using LLM. For example, we randomly select the facts $(LA\ Angles,\ played\_for^{-1},\ Otani\ Shohei)$, $(Japan,\ born\_in^{-1},\ Otani\ Shohei)$ from KGs. Then, we could induce a conjunction query $q = V_? : played\_for^{-1}(LA\ Angles, V_?) \wedge born\_in^{-1}(Japan, V_?)$ and verbalize it using LLM as *"Which players played for the LA Angels and were born in Japan?"*. To assess model robustness under natural language ambiguity, particularly when entity mentions are paraphrased and not explicitly found in KGs, we utilize an LLM to rephrase entities into semantically similar yet distinct noun or noun phrases (e.g. "iPhone": "smartphone developed by Apple") based on prompt in Table 16. We then randomly replace the original entities with their rephrased counterparts. To ensure semantic consistency, we additionally verify that the meaning of each rephrased phrase remains equivalent to that of the original entity. Table 17 is an example prompt for this step. We remove any FOL and natural language pairs that overlap with the reasoning data to prevent data leakage.

F Baseline Models

We broadly categorize the baselines as follows: (1) Information Retrieval (IR) approach, (2) Complex Query Answering (CQA).

RAG (Lewis et al., 2020) retrieves a relevant subset of an unstructured textual corpus, which is an external knowledge source, and incorporates it into a prompt to enhance the response quality of LLM. GraphRAG (Edge et al., 2024) enhances the

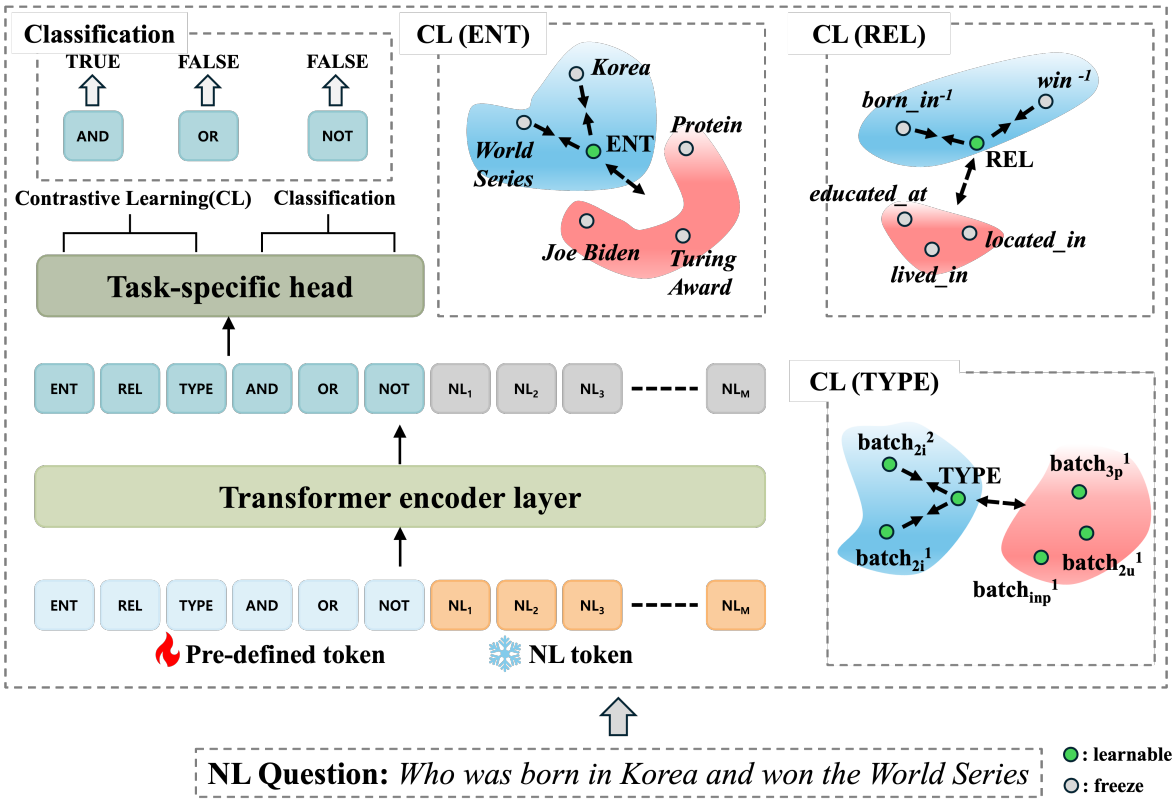

Figure 6: Illustration of Align module.

Datasets	# Entity	# Relation	# Training Edge	# Validation Edge	# Test Edge	# Total Edge
FB15k-237	14,505	237	272,115	17,526	20,438	310,079
UMLS	135	46	5,216	652	661	6,529
CoDEx	2,034	42	32,888	1,877	1,828	36,593

Table 5: Statistics of KG datasets.

Datasets	# Train	# Validation	# Test	# Synthethic
FB15k-237	789,686	77,751	75,748	821,117
CoDEx	60,509	11,898	12,868	45,331
UMLS	22,809	2,077	3,016	25,306

Table 6: Statistics of the natural language questions, including splits for training, validation, testing, and synthetic.

RAG through a hierarchical framework. First, it constructs KGs from raw text by using LLM. Then, it organizes community hierarchy from the constructed KGs and generates summaries for these communities. Finally, LLM generates responses based on these summaries. In both baselines, to treat KGs as textual documents, we convert each fact within KGs into a natural language sentence. In contrast, KAPING (Baek et al., 2023) retrieves relevant facts based on the semantic similarity between the question and the facts directly associated with the topic entity. RoG (Luo et al., 2024) fine-tunes LLM to generate supportive information in the form of paths derived from the given question. These generated paths are then utilized to identify reasoning paths within KGs. Both G-Retriever (He et al., 2025) and EPR (Ding et al., 2024) extract a subgraph from the KGs. G-Retriever encodes a subgraph using a graph encoder while simultaneously generating a textual embedding. The encoded graph representation is then fed into the LLM alongside the textual embedding, functioning as a soft prompt (Lester et al., 2021; Li and Liang, 2021). EPR constructs a subgraph using atomic patterns to reduce noisy retrieval and integrates NSM (He et al., 2021) to find the answer entities. Due to constraints in time and computational re-

	Align	QD (Llama)	RE (FLAN-T5-xl)	Reasoner	KGE	PLM	TOTAL
FB15k-237	22M	8B	3B	62M	30M	66M	11.2B
UMLS	11M	8B	3B	31M	0.2M	66M	11.1B
CoDEx	19M	8B	3B	62M	4M	66M	11.1B

Table 7: Number of parameters for each module in Ours. QD denotes Query decomposition module, and RE denotes Relation extraction module.

sources, we fine-tune RoG using LoRA (Hu et al., 2022), a parameter-efficient method for adapting LLMs. G-Retriever also employs LoRA as part of its original model design.

In CQA methods, we adopt BetaE (Ren and Leskovec, 2020), ConE (Zhang et al., 2021) and Q2T (Xu et al., 2023) as baselines. BetaE encodes logical queries based on the beta distribution, which enables negation operations that are not supported in previous work. Similarly, for addressing the limitation of existing geometric-based embedding, which cannot model negation operator, ConE (Zhang et al., 2021) introduces cartesian products of two-dimensional cones to map logical queries into embedding space. In contrast, Q2T (Xu et al., 2023) aims to mitigate inconsistency between pretraining and the downstream task. To fully exploit the capability of pretrained KGE, it handles downstream task, i.e, complex query reasoning, as a simple query reasoning, which is the objective of pretraining.

G Applicability in the Existing Setting

As discussed in Section 3, the problem addressed in this paper differs from that of conventional KGQA as follows: 1) handling questions over KGs that may not cover all the facts required to find the answer entity, 2) handling questions composed of various logical operators, including negation, which has been underexplored in previous work. However, to explore the applicability of our framework in a previous setting, we evaluate our framework on CWQ, a benchmark dataset widely used in prior studies.

The conventional KGQA assumes that the KGs are complete from the perspective of QA, and the topic entity, which indicates the region to attend, is provided. Therefore, to exploit this assumption, we use our framework as a retriever, without any internal modifications, alongside an LLM. Specifically, once our framework predicts the candidate entities, we input the shortest path connecting the topic entity and each candidate entity along with the question into the prompt, following an approach similar to RoG. We exclude RAG, GraphRAG, and G-Retriever due to computational constraints. In particular, RAG and GraphRAG require verbalizing over 7 million triples into textual form to align with their input format. This conversion step alone takes over 200 hours, making it infeasible to include them in our experiments. G-Retriever is also omitted, as its training time exceeded 100 hours.

As shown in Table 8, our framework, coupled with LLM, shows comparable performance with EPR, which is the best competitor on CWQ. This result suggests that while our framework is not a tailored approach for the conventional setting, it remains applicable with slight modifications to leverage the assumptions underlying standard KGQA settings.

Model	Hits@1
KAPING	26.5
RAG	-
GraphRAG	-
LLaMA2-Chat-7B	22.1
RoG planning + LLaMA2-Chat-7B	55.6
G-Retreiver	-
EPR	60.4
Ours + LLaMA2-Chat-7B	60.5

Table 8: Performance comparison in Hits@1 across baseline models on the CWQ dataset.

H Computational Efficiency Analysis

H.1 Comparison of inference times based on the size of KGs.

We compare the inference time of each module (Align module, Query decomposition module, Relation extraction module, and Reasoner) on the CoDEx and CWQ datasets to verify the scalability of our model on large-scale KGs. The CWQ dataset leverages Freebase, a large-scale KG, for Ques-

tion Answering (QA), encompassing over 2 million entities, whereas CoDEx is based on a comparatively smaller KG with approximately 2,000 entities. From the results in Table 9, despite the more than 1000-times difference in the number of entities between the KGs used in the CoDEx and CWQ, the inference time for each module does not exhibit a significant increase. Notably, Query decomposition module, the most time-consuming module in our framework, shows a reduced inference time on CWQ. This is because CWQ involves relatively simpler question types compared to CoDEx, leading to shorter question lengths and a relatively smaller number of generated sub-questions. These results confirm that our model remains computationally efficient at scale.

Datasets	Align	QD	Rel	Reasoner
CoDEx	8.23	948.60	191.81	22.56
CWQ	11.81	834.68	239.09	26.53

Table 9: Average inference time (in ms) for 1,100 samples in CoDEx and CWQ datasets.

H.2 Inference Time Comparison with Baseline Models

To provide a more detailed discussion of the efficiency of our method, we compare inference time against some baselines on the CoDEx dataset. Since the inclusion of LLM can significantly increase inference time, we focus our comparison on the baselines that utilize LLM. As shown in Table 10, our model achieves competitive computational efficiency. This is because, unlike other models, our framework does not require any preprocessing or retrieval of information for integration into the prompt with a question. Similarly, Llama-3-8B-Instruct and RAG exhibit high computational efficiency since they rely solely on the question or require simple similarity-based retrieval. While KAPING employs simple similarity-based retrieval, it tends to generate verbose responses that include the retrieved information, which results in additional overhead. Other baselines require multiple LLM calls or overburdened processing steps for sophisticated retrieval, which leads to computational inefficiency.

Model	Inference Time
KAPING	3,597.3
RAG	1,092.1
GraphRAG	7,032.7
Llama-3-8B-Instruct	470.8
RoG	2,058.8
G-Retreiver	6,129.5
Ours	1,171.2

Table 10: Comparison of the average inference time (in ms) against baselines.

Datasets	Type	Actual Query	Entity	Relation
FB15k-237	78.8	6.3	52.9	45.5
CoDEx	62.8	25.7	75.1	76.2
UMLS	56.2	16.9	85.4	60.7

Table 11: Accuracy (%) of converting natural language questions into first-order logic (FOL) queries.

I Prompts for Data Generation using LLM

I.1 FOL query to NL question

Table 12 shows an example prompt for converting an FOL query into a natural language question.

I.2 Question decomposition

Table 13 presents an example prompt used for question decomposition.

I.3 Topic entity extraction

Table 14 provides an example prompt for entity extraction from a natural language question, specifically for input in the context of the given entity setting. We feed the extracted entities to the baseline model, such as (Luo et al., 2024), which requires topic entities to derive the answer from the question.

I.4 NL question to FOL query

Table 15 represents an example prompt for FOL query generation. We adopt the previous method (Yang et al., 2024) along with its prompt. Additionally, Table 11 shows its performance metrics, including type, actual query, entity, and relation matching accuracy between the original FOL query and the converted FOL query. From the result, we can observe that converting natural language into first-order logic is a challenging task.

NL Question Generation Prompt

You are a natural language generator.
Your task is to understand a First-Order Logic (FOL) query and convert it into a natural language question.
Please convert the following FOL query into a natural language question.
Provide only one natural language question as output, without any additional explanation.
FOL Query Structure: $q = V?.\ \exists V : r(e, V?)$

Examples of Conversion:
$q = V?.\ \exists V$:(-/base/popstra/celebrity/breakup./base/popstra/breakup/participant('Howard Hughes', X?)) => Who did Howard Hughes participate in a breakup with?
$q = V?.\ \exists V$:(-/tv/tv_program/regular_cast./tv/regular_tv_appearance/actor('Will Arnett', X?)) => What TV programs is Will Arnett a regular cast member of?
$q = V?.\ \exists V$: (-/film/film/prequel('National Treasure', X?)) => What are the prequels to the film "National Treasure"?

FOL Query to Convert:
$q = V?.\ \exists V$: (+/people/person/profession('Manny Coto', X?)) =>

Table 12: Example prompt for FOL to NL generation.

NL Question Decomposition Prompt

You are an intelligent assistant and a natural language question generator that creates effective and concise questions.
ONLY RETURN THE SUB-QUESTION (QUESTIONS) WITHOUT ANY EXPLANATION.

Instructions:

- Extract every claim from the provided question.
- Resolve any coreference for clarity.
- Convert each claim into a concise (less than 15 words)
- Generate no more than 5 sub-questions.
- Generate sub-questions only based on information available in the original question.
- Separate multiple sub-questions with ' / ', except for certain 'p' type questions. A 'p' type question involves a sequence of steps or a path where one piece of information is used to find another (for example, "Which educational institutions have the same colors as the Indiana Pacers?" requires first finding the colors of the Indiana Pacers, then finding educational institutions with those colors). Do not separate these.
- DO NOT RETURN ANYTHING BUT THE ANSWER.

Examples:

1. Question: Which football teams have a Forward, a Goalkeeper, and a Defender on their roster?"
Response: Which football teams have a Forward? / Which football teams have a Goalkeeper? / Which football teams have a Defender?
2. Question: Who are the University of Miami graduates who are married to Melanie Griffith?
Response: Who are the University of Miami graduates? / Who is married to Melanie Griffith?
3. Question: What is Billy Boyd's profession?
Response: What is Billy Boyd's profession?

Complete the following:
Question: What is Manny Coto's profession?
Response:

Table 13: Example prompt for NL question decomposition.

Topic Entity Extraction Prompt

You are an assistant to help people get the entities from the given question, following the instructions strictly. Return only the entity from the given question without other words.

Please identify the entities in the given question.

Examples:

Q: Which sports team play in London did Drogba play for?
A: 1. London
2. Drogba
Q: When was the film One Fine Day released?
A: 1. One Fine Day
Q: Who directed the movie Into the Blue?
A: 1. Into the Blue

Your Task:
Q. What is Manny Coto's profession?
A.

Table 14: Example prompt for topic entity extraction.

NL Question to FOL Query

Translate the following Natural Language (NL) statement to a First-Order Logic (FOL) rule

NL:
What is Manny Coto's profession?

Table 15: Example prompt for NL question to FOL query.

Augmentation with Semantically Similar Variants Prompt

The task is to create a concise, semantically equivalent representation of an entity while following the patterns demonstrated in the examples.
Please convert the given entity into a semantically identical but different noun or noun phrase coincisely.
Ensure that the output captures the essential meaning of the input entity.
Please return only the output, avoiding additional explanations. Follow the format provided below:

Examples
Example 1:
Input: Japan
Output: island country near South Korea
Example 2:
Input: the United States
Output: USA
Example 3:
Input: Python
Output: programming language commonly used for data science and web development
Example 4:
Input: Albert Einstein
Output: physicist known for the theory of relativity
Example 5:
Input: The Great Wall of China
Output: an ancient series of walls and fortifications in northern China
Example 6:
Input: iPhone
Output: smartphone developed by Apple
—
Now process the following input and return response:
Input: Eleanor Roosevelt
Output:

Table 16: Example prompt for pretraining data augmentation.

Verification for Semantically Similar Variants Prompt

The task is to verify that two given inputs are semantically equivalent representations while following the patterns demonstrated in the examples.
Please verify whether the two given representations are semantically equivalent.
If they are semantically equivalent, return Yes; otherwise, return No. Please return only Yes or No, avoiding additional descriptions like "Sure!".
Follow the format provided below:
Examples
Example 1:
Representation 1: iPhone
Representation 2: smartphone made by USA enterprise
Output: No
Example 2:
Representation 1: iPhone
Representation 2: smartphone made by Apple
Output: Yes
Example 3:
Representation 1: Japan
Representation 2: country near South Korea
Output: No
Example 4:
Representation 1: Japan
Representation 2: island country near South Korea
Output: Yes
Example 5:
Representation 1: Albert Einstein
Representation 2: scientist who developed the theory of relativity
Output: Yes
Example 6:
Representation 1: Albert Einstein
Representation 2: famous scientist
Output: No
Example 7:
Representation 1: The Great Wall of China
Representation 2: ancient structure in China
Output: No
Example 8:
Representation 1: The Great Wall of China
Representation 2: ancient series of walls and fortifications in northern China
Output: Yes
—
Now process the following input and return a response starting with "Output:":
Representation 1:
Representation 2:

Table 17: Example prompt for verification of semantical similar variants.