RTQA : Recursive Thinking for Complex Temporal Knowledge Graph Question Answering with Large Language Models

Zhaoyan Gong[♠], **Juan Li**[♠], **Zhiqiang Liu**[♠♢], **Lei Liang**[♣♢], **Huajun Chen**[♠♢], **Wen Zhang**[♠♢†]
[♠] Zhejiang University [♣] Ant Group
[♢] ZJU-Ant Group Joint Lab of Knowledge Graph
{gongzhaoyan,zhang.wen}@zju.edu.cn

Abstract

Current temporal knowledge graph question answering (TKGQA) methods primarily focus on implicit temporal constraints, lacking the capability of handling more complex temporal queries, and struggle with limited reasoning abilities and error propagation in decomposition frameworks. We propose **RTQA**, a novel framework to address these challenges by enhancing reasoning over TKGs without requiring training. Following recursive thinking, RTQA recursively decomposes questions into sub-problems, solves them bottom-up using LLMs and TKG knowledge, and employs multi-path answer aggregation to improve fault tolerance. RTQA consists of three core components: the Temporal Question Decomposer, the Recursive Solver, and the Answer Aggregator. Experiments on MultiTQ and TimelineKGQA benchmarks demonstrate significant Hits@1 improvements in "*Multiple*" and "*Complex*" categories, outperforming state-of-the-art methods. Our code and data are available at https://github.com/zjukg/RTQA.

1 Introduction

In the real world, entities and relationships evolve dynamically, making Temporal Knowledge Graphs (TKGs) with time-aware quadruples more challenging yet practically significant for Question Answering (QA) compared to static Knowledge Graphs (KGs) (Chen, 2024). For instance, the question "*Who is the US President in 2025?*" can be answered using the quadruple *(Trump, president of, United States, 2025)*, representing a simple temporal query.

Recent TKGQA research targets complex queries, including implicit temporal constraints, multi-constraint combinations, multi-hop reasoning, and multi-granular time, as exemplified by the question in Figure 1: "*Before Kuwait, which country received the Government Delegation of North Korea's visit last?*" Such queries are highly relevant to real-world applications.

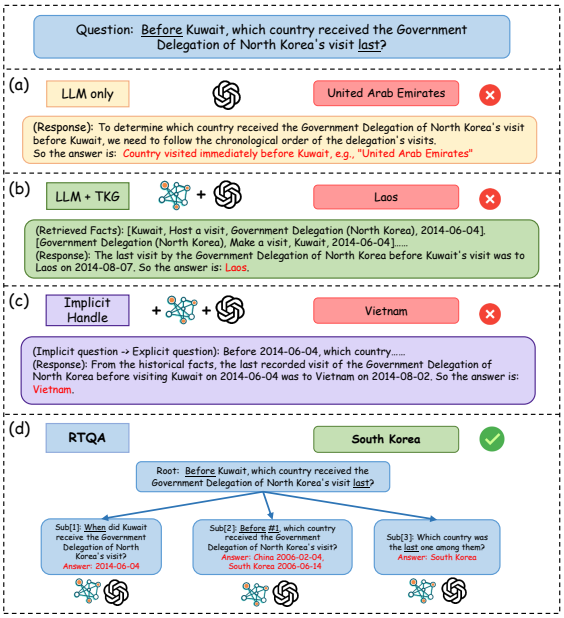

Figure 1: Motivation comparison: Prior methods (a–c) fail at multi-constraint reasoning, while (d) **RTQA** solves it via recursive sub-question decomposition.

While prior works have focused on simple (Saxena et al., 2021; Mavromatis et al., 2022) or implicit temporal questions (Chen et al., 2022; Qian et al., 2024; Jia et al., 2024), the integration of Large Language Models (LLMs) into TKGQA offers new opportunities due to their excellent reasoning ability. However, two key challenges persist:

(1) Limited reasoning for complex temporal queries. LLMs often hallucinate (Azaria et al., 2024; Zhao et al., 2024) when addressing intricate questions. As shown in Figure 1(a), relying solely on internal knowledge yields incorrect answers like "*United Arab Emirates*". (b) Incorporating TKG facts resolves simpler queries but fails to handle implicit constraints like "*before Kuwait*", producing errors such as "*Laos*". (c) Single-round question rewriting converts implicit constraints to

† Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 9865–9882
November 4-9, 2025

explicit timestamps (e.g., "*before 2014-06-04*"), but struggles with combined constraints like "*before/last*", leading to incorrect answers like "*Vietnam*". Developing frameworks for multi-fact and multi-constraint temporal reasoning remains critical.

(2) Error propagation in decomposition frameworks. Existing methods lack fault tolerance, allowing sub-question errors to propagate. For example, in Figure 2(b), the query "*When Stalin ended his leadership in his own country, what job did Churchill work for?*" is decomposed into sub-questions: "*Stalin was the leadership of which country?*" yields "*Soviet Union*", followed by "*When was Stalin end his leadership in #1*[1]*?*" answered incorrectly as "*1929*", leading to the erroneous final answer "*Chancellor of the Exchequer*" for "*When #2*[2]*, what job did Churchill work for?*". Addressing error propagation issues and building more robust frameworks that enhance fault tolerance in LLM reasoning or fact retrieval processes represents another significant challenge.

To address these challenges, we introduce **RTQA** (**R**ecursive **T**emporal Knowledge Graph **Q**uestion **A**nswering), a novel TKGQA framework that decomposes complex temporal questions into sub-questions and performs recursive bottom-up reasoning. By integrating external knowledge from TKGs, RTQA enhances LLMs' ability to tackle intricate temporal queries.

Following a divide-and-conquer strategy, RTQA mimics human problem-solving by breaking down complex questions into manageable parts. As shown in Figure 1(d), a question is split into three sub-questions: extracting implicit time, applying a "before" constraint, and applying a "last" constraint. The answer to Sub[1] ("2014-06-04") informs Sub[2], which generates entity-time pairs (e.g., "China 2008-02-04, South Korea 2006-06-14") satisfying the "before #1" constraint. Sub[3] then selects the entity that meets the "last" constraint, producing the final answer, "South Korea".

To reduce error propagation in sub-questions, we designed a multi-path answer aggregation module that combines answers from both sub-questions and the original question, selecting the most reliable response. As illustrated in Figure 2(a), we define `IR_answer` and `child_answer`. Atomic

[1]#1 is a placeholder for the answer "Soviet Union" to Sub[1]: "Which country did Stalin lead?"

[2]#2 is anather placeholder, representing the answer "1929" to the Sub[2]: "When was Stalin end his leadership in #1?"

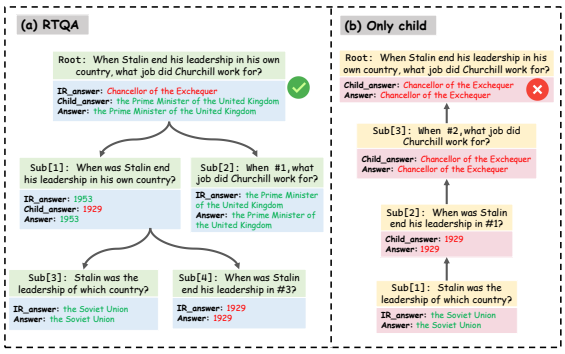

Figure 2: Comparison of **RTQA** and **Only-Child** strategies. RTQA mitigates error propagation by integrating child_answer with IR_answer, while Only-Child relies solely on child_answer, compounding earlier errors.

questions (sub[3], sub[4]) rely on a single answer source, while non-atomic questions (sub[1], sub[2], Root) aggregate multiple sources. For instance, when sub[4] incorrectly outputs "*1929*," sub[1]'s `IR_answer` correctly identifies "*1953*," preventing error propagation and ensuring the accurate final answer, "*the Prime Minister of the UK*" instead of the wrong "*Chancellor of the Exchequer*."

We conduct extensive experiments on two challenging TKGQA benchmarks. RTQA consistently outperforms state-of-the-art methods, with notable gains in the "*Multiple*" and "*Complex*" categories. Our main contributions are summarized as follows:

- We introduce a framework that recursively decomposes complex temporal questions into sub-questions, reasoning bottom-up to derive accurate answers.
- We aggregate answers from multiple sources for each question *original, intermediate, atomic*, mitigating error propagation and enhancing framework robustness.
- Our training-free, plug-and-play approach requires no computational overhead, adapts to various large models, and demonstrates significant performance gains in complex temporal question answering.

2 Related Work

2.1 TKGQA

TKGQA methods can be categorized into semantic parsing-based approaches and embedding-based approaches, with a recent emergence of methods leveraging large language models.

Semantic Parsing-based methods Semantic parsing-based methods convert natural language

questions into logical expressions to query TKGs, as seen in TEQUILA (Jia et al., 2018), SYGMA (Neelam et al., 2021), SF-TQA (Ding et al., 2022), and Prog-TQA (Chen et al., 2024b). These approaches offer high accuracy when queries are well-formed but struggle with complex questions due to syntax errors in logical expressions, leading to query failures.

Embedding-based Methods TKG Embedding-based methods encode questions and TKG quadruples as low-dimensional vectors, ranking answers by vector semantic similarity. CronKGQA (Saxena et al., 2021) introduces learnable reasoning, TempoQR (Mavromatis et al., 2022) enhances embeddings with contextual and temporal modules, and MultiQA (Chen et al., 2023) aggregates multi-granular time information. Other approaches incorporate graph neural networks (Jia et al., 2024; Liu et al., 2023; Sharma et al., 2023). These methods ensure high execution rates but can only handle simple questions and perform poorly on complex temporal questions.

LLM-based Methods Recent LLM-based approaches, such as ARI (Chen et al., 2024c), GenTKGQA (Gao et al., 2024), FAITH (Jia et al., 2024), and TimeR4 (Qian et al., 2024), leverage LLMs for TKGQA. Unlike these methods, which often require retraining, our **RTQA** framework is training-free and plug-and-play, handling complex queries with multiple entities, multi-hop reasoning, and compound temporal constraints while maintaining compatibility with various LLMs.

2.2 Question decomposing

Question decomposition emulates human problem-solving by breaking complex queries into simpler sub-questions, a strategy effective for multi-hop reasoning in KGQA (Cao et al., 2022; Khot et al., 2023; Trivedi et al., 2022; Cao et al., 2023). However, existing approaches inadequately address temporal questions, necessitating advanced frameworks like RTQA.

3 Preliminary

Temporal constraint defines a condition related to a specific time point or interval that must be met by both the answer and its supporting evidence. This includes 13 Allen temporal relations (Allen, 1984), 3 temporal set relations, duration comparisons, and sorting mechanisms (Sun et al., 2025).

TKG A temporal knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{F}\}$ is a directed graph where vertices are a set of entities $\mathcal{E}$. The edges are a set of predicates $\mathcal{P}$ with timestamps $\mathcal{T}$. The quadruple set $\mathcal{F} = \{(s, p, o, t) \mid \mathcal{E} \times \mathcal{P} \times \mathcal{E} \times \mathcal{T}\}$ represents the temporal facts, where s and o are subject and object, respectively, and p is the predicate between s and o at timestamp t.

TKGQA is a task to infer the correct answer to a natural language question $q \in \mathcal{Q}$ based on relevant quadruples $f = (s, p, o, t)$ in the TKG, where the answer can be either an entity name or a timestamp.

4 Method

4.1 Method Overview

Inspired by the divide-and-conquer principle, RTQA enables efficient handling of complex temporal dependencies. As shown in Figure 3, **Temporal Question Decomposer** (Section 4.2) firstly transforms complex temporal questions into a series of simpler sub-questions by identifying implicit temporal constraints (e.g., "*before*", "*last*") and converting them into explicit temporal expressions. It also extracts relevant multi-hop facts and temporal granularity information. For example, the question *"Before Kuwait, which country received the Government Delegation of North Korea's visit last?"* is decomposed into three sub-questions: (1) *When did Kuwait receive the visit?* (2) *Which countries received the visit before #1?* (3) *Which was the latest among them?* Next, **Recursive Solver** (Section 4.3) leverages the reasoning ability of LLMs and the factual knowledge from the TKG to recursively solve sub-questions in a bottom-up manner. The resulting answers are used to replace placeholders in parent questions, forming a progressive reasoning chain. For instance, the answer *"2014-06-04"* to sub-question (1) serves as the temporal reference for sub-question (2), which then filters out earlier visits, and sub-question (3) selects the latest one from the filtered list. This recursive approach effectively handles both implicit and compound temporal constraints. Finally, **Answer Aggregator** (Section 4.4) consolidates results, evaluating candidates (e.g., "*South Korea*" vs. "*Vietnam*") to ensure accuracy and robust fault tolerance.

4.2 Temporal Question Decomposer

The goal of this stage is to decompose a complex temporal question Q into a series of sub-questions

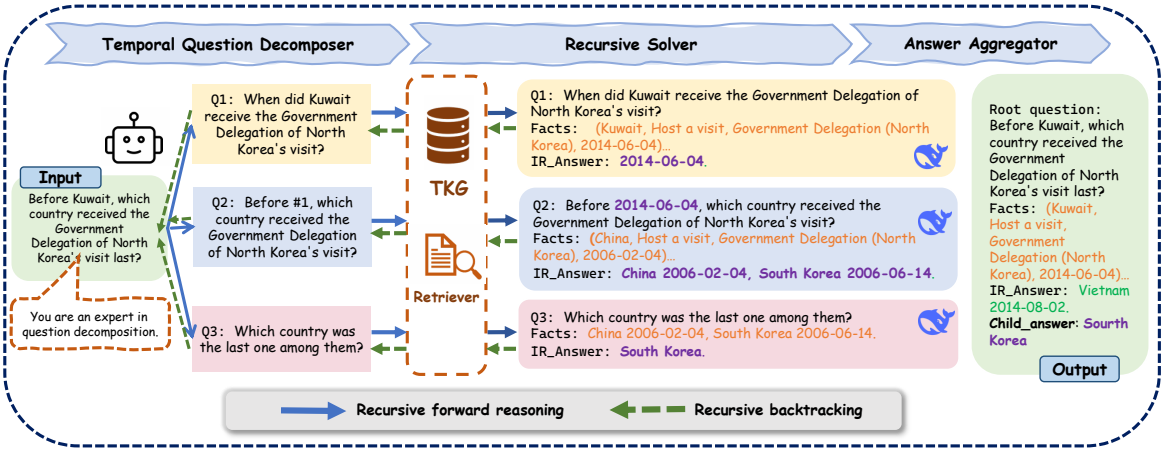

Figure 3: An illustration of the **RTQA** framework applied to a complex temporal question. The framework consists of three stages: (I) **Temporal Question Decomposer**, which breaks down the original query into sub-questions with explicit temporal constraints; (II) **Recursive Solver**, where each sub-question is solved using an LLM and retrieved TKG facts; and (III) **Answer Aggregator**, which integrates the sub-answers to produce the final answer. The reasoning process follows a bottom-up recursive traversal from the root of the decomposition tree, enabling robust aggregation of intermediate results.

T, where Q is the root node in T. Basically, as shown in Figure 3, the query tree is generated by LLMs with few-shot prompting.

The question decomposition process can be formalized as a sequence of transformations applied to an input question Q. Let the instruction template be denoted as $\mathcal{I}$, and the question type as $\tau = \textbf{Type}(Q)$, where $\text{Type}(\cdot)$ is the type identification function. The prompt is constructed and the LLM response is obtained as follows:

$$p \leftarrow \text{BuildPrompt}(Q, \tau, \mathcal{I}), \quad (1)$$

$$r_{\text{llm}} \leftarrow \text{LLMCaller}(Q, p), \quad (2)$$

where $\text{BuildPrompt}(\cdot)$ denotes the prompt construction function, and $\text{LLMCaller}(\cdot)$ represents the LLM call function. The structured response is then parsed, and the temporal decomposition tree is constructed:

$$S \leftarrow \text{ParseStruct}(r_{\text{llm}}), \quad (3)$$

$$\mathcal{T} \leftarrow \text{BuildTree}(S), \quad (4)$$

where $\text{ParseStruct}(\cdot)$ extracts structured elements from the LLM output, and $\text{BuildTree}(\cdot)$ organizes them into a hierarchical decomposition tree $\mathcal{T}$.

Each node $q^i \in \mathcal{T}$ contains: (i) a node index `idx`, (ii) the question text `question_text`, (iii) a list of child nodes `sons`, (iv) the parent node index `fa`, (v) metadata including the question type label `qlabel`, and (vi) the `gold_answer`. This structure maintains hierarchical and semantic fidelity throughout the reasoning process.

The construction of the prompts is tailored to various types of temporal questions. For each type, 5–10 question examples are carefully selected from the validation set, with their sub-question decompositions manually crafted. The specific prompts constructed for each category, along with their corresponding decompositions, are illustrated in the Figure 6, 7, provided in the Appendix C.1 for more details.

4.3 Recursive Solver

Recursive solving process. We adopt a recursive post-order traversal to solve the query decomposition tree, starting from the root and proceeding in a bottom-up manner. The solver is formalized as a unified recursive function $\text{Solve}(q^i, \mathcal{T}, \mathcal{R}, \theta)$, where $\mathcal{R}$ denotes the retriever for TKG grounding, and θ denotes the reasoning LLM.

For a leaf node $q^i \in \mathcal{T}$, the solver first retrieves relevant facts from the TKG and then invokes the LLM to generate an answer, as defined below:

$$\mathcal{F}^i \leftarrow \text{Retrieve}(q^i, \mathcal{R}), \quad (5)$$

$$a^i \leftarrow \text{Reason}(q^i, \mathcal{F}^i, \theta). \quad (6)$$

For a non-leaf node q^i, the solver recursively processes each child $q^{c_j} \in \texttt{sons}(q^i)$, where $j = 1, \ldots, n$ and n is the number of sub-questions. For the first child, the answer is computed directly:

$$q^{c_1}_{\text{updated}} \leftarrow q^{c_1}, \quad (7)$$

$$a^{c_1} \leftarrow \text{Solve}(q^{c_1}_{\text{updated}}, \mathcal{T}, \mathcal{R}, \theta). \quad (8)$$

Subsequent questions are updated by replacing placeholders (e.g., $\#k$) with prior answer a^k:

$$q^{c_2}_{\text{updated}} \leftarrow \text{Replace}(q^{c_2}, \{a^{c_1}\}), \quad (9)$$

$$a^{c_2} \leftarrow \text{Solve}(q^{c_2}_{\text{updated}}, \mathcal{T}, \mathcal{R}, \theta), \quad (10)$$

$$\vdots \quad (11)$$

$$q^{c_n}_{\text{updated}} \leftarrow \text{Replace}(q^{c_n}, \{a^{c_1}, \ldots, a^{c_{n-1}}\}), \quad (12)$$

$$a^{c_n} \leftarrow \text{Solve}(q^{c_n}_{\text{updated}}, \mathcal{T}, \mathcal{R}, \theta), \quad (13)$$

where $\{a^{c_1}, \ldots, a^{c_{j-1}}\}$ denotes answers from prior sub-questions used for reference replacement.

After solving all sub-questions, the final answer of this non-leaf node q^i is aggregated via a summary function:

$$a^i_{child} = \text{Summarize}(q^i, \{a^{c_1}, \ldots, a^{c_n}\}). \quad (14)$$

This recursive procedure ensures consistent resolution of complex temporal queries across all levels of the tree.

Relevant Facts Retriever. The quadruples or quintuples in TKG are converted into natural language statements in the following two forms:

```
{subject} {predicate} {object} in {time}
{subject} {predicate} {object} from {start} to {end}
```

The statements are then embedded using a dense encoder along with input question. The top-K most relevant facts are retrieved based on similarity.

Explainable reasoning with LLM. The RTQA framework employs a post-processing module to distill concise, standardized answers from LLM outputs. Step-by-step reasoning, guided by precise instructions, ensures transparency by preserving the full inference chain, a hallmark of RTQA's interpretability. The LLM concludes with a structured summary, *So the answer is:*, enabling reliable extraction of the final entity or timestamp while retaining the reasoning for clarity. The prompts driving this process are detailed in Appendix C.2 Figure 8, 9.

Time Expression Standardization. Before invoking the recursive solver, all time expressions are standardized to the ISO 8601[3] format (`yyyy-mm-dd`). This preprocessing step ensures consistent handling of temporal references across different granularities (year, month, day), addressing the variability in natural language expressions and improving the accuracy of temporal reasoning.

[3] https://www.iso.org/iso-8601-date-and-time-format.html

4.4 Answer Aggregator

The aggregator selects the most plausible final answer by fusing two candidates: a^i_{IR} and a^i_{child}. Specifically, a^i_{IR} is produced by retrieving relevant TKG facts and applying LLM reasoning. a^i_{child} aggregates answers from the child nodes of the query tree. In cases of ambiguity, the aggregator leverages the original query context to choose the most appropriate answer, ensuring alignment with user intent. The detailed prompt design for this aggregation process is provided in Appendix C.3 Figure 10. This process is formalized as:

$$a^i_{\text{final}} = \text{Aggregator}(a^i_{\text{IR}}, a^i_{\text{child}}). \quad (15)$$

In conclusion, the answer aggregator serves as a critical module to prevent errors from propagating upstream by selecting one of three answer sources as the final answer.

5 Experiment

5.1 Experimental Setup

Datasets We evaluate RTQA on two challenging TKGQA benchmarks: MULTITQ (Chen et al., 2023) and TIMELINEKGQA (Sun et al., 2025). MULTITQ offers large-scale QA pairs with diverse temporal granularities, while TIMELINEKGQA covers questions with varying complexity and time formats. The test sets contain 54,584 and 8,344 questions, respectively. Detailed statistics and category distributions are provided in Appendix A.

Baselines We compare RTQA against three types of baselines on MULTITQ: (1) **Pre-trained LMs**, including BERT (Devlin et al., 2019), DistillBERT (Sanh et al., 2019), ALBERT (Lan et al., 2020), LLaMA2 (Touvron et al., 2023), and ChatGPT; (2) **TKG embedding-based methods**, including EmbedKGQA (Saxena et al., 2020), CronKGQA (Saxena et al., 2021), and MultiQA (Chen et al., 2023); (3) **LLM-based methods**, including ARI (Chen et al., 2024c) and TimeR4 (Qian et al., 2024). For TIMELINEKGQA, due to its complexity, existing embedding-based models are not directly applicable. Following (Sun et al., 2025), we adopt a Retrieval-Augmented Generation (RAG) baseline.

Implementation Details We used the OPENAI API (gpt-4o-mini[4]) for temporal question decom-

[4] https://platform.openai.com/docs/models/gpt-4o-mini

Table 1: Performance comparison of baselines and RTQA on Hits@1 and Hits@10 across various question types and answer types on MultiTQ testset. The best and second best results are marked in **bold** and underlined, respectively.

Model	Hits@1					Hits@10				
	Overall	Question Type		Answer Type		Overall	Question Type		Answer Type	
		Multiple	Single	Entity	Time		Multiple	Single	Entity	Time
BERT	0.083	0.061	0.092	0.101	0.040	0.441	0.392	0.461	0.531	0.222
DistillBERT	0.083	0.074	0.087	0.102	0.037	0.482	0.426	0.505	0.591	0.216
ALBERT	0.108	0.086	0.116	0.139	0.032	0.484	0.415	0.512	0.589	0.228
LLaMA2	0.185	0.101	0.220	0.239	0.055	-	-	-	-	-
ChatGPT	0.102	0.077	0.147	0.137	0.020	-	-	-	-	-
EmbedKGQA	0.206	0.134	0.235	0.290	0.001	0.459	0.439	0.467	0.648	0.001
CronKGQA	0.279	0.134	0.337	0.328	0.156	0.608	0.453	0.671	0.696	0.392
MultiQA	0.293	0.159	0.347	0.349	0.157	0.635	**0.519**	0.682	**0.733**	0.396
ARI	0.380	0.210	0.680	0.394	0.344	-	-	-	-	-
TimeR4	0.728	0.335	0.887	0.639	**0.945**	-	-	-	-	-
RTQA	**0.765**	**0.424**	**0.902**	**0.692**	0.942	**0.768**	0.427	**0.907**	0.697	**0.942**

Table 2: Performance(Hits@1) comparison of RAG baseline and RTQA across *Simple*, *Medium*, *Complex* on TimelineKGQA test dataset.

Model	Hits@1			
	Overall	Simple	Medium	Complex
RAG baseline	0.235	0.704	0.092	0.009
RTQA	**0.298**	0.608	**0.218**	**0.135**

Table 3: Ablation studies of RTQA on MultiTQ.

Model	Overall	Question Type		Answer Type	
		Multiple	Single	Entity	Time
RTQA	**0.765**	**0.424**	**0.902**	**0.692**	**0.942**
w/o decomposer	0.709	0.214	0.890	0.596	0.958
w/o multi-answer	0.752	0.341	0.904	0.667	0.942
w/o fact retrieval	0.070	0.015	0.090	0.096	0.013

position, and the DEEPSEEK API (deepseek-v3-250324) for answer reasoning on the MultiTQ dataset. For TimelineKGQA, all stages used OPENAI (gpt-4o-mini). The temperature was set to 0 for deterministic outputs. We employed the BGE-M3[5] (Chen et al., 2024a) model via Hugging Face to generate dense embeddings of TKG triples and questions, though hybrid retrieval was not used. Dense retrieval and clustering were performed using FAISS (Douze et al., 2024), following (Qian et al., 2024). To avoid excessive context, we limited reasoning inputs to the top 50 retrieved facts.

5.2 Main Results

We present the experimental results in comparisons between our model and existing state-of-the-art baseline models on the MultiTQ and TimelineKGQA datasets in Table 1 and Table 2.

Performance Comparison on MultiTQ We evaluate performance using Hits@1 and Hits@10, with breakdowns by question type (multiple, single) and answer type (entity, time).[6] As shown in Table 1, RTQA outperforms all baselines across nearly all metrics. It achieves a Hits@1 of 0.765, surpassing the second-best model TimeR4 (0.728). For question types, RTQA scores 0.424 on multiple and 0.902 on single, demonstrating strong adaptability to varying complexities. On Hits@10, RTQA maintains the lead with 0.768 overall, and excels on time answers with a score of 0.942, highlighting its effectiveness in handling temporal reasoning in TKGQA.

Pre-trained language models such as BERT, DistillBERT, and ALBERT perform poorly, with Hits@1 below 0.11, indicating that generic pre-trained models are insufficient for temporal reasoning. While models like EmbedKGQA, CronKGQA, and MultiQA perform reasonably on single-choice and entity questions, they struggle with multiple and time answers. TimeR4, which integrates LLMs for TKGQA, shows better performance but still falls short of RTQA.

Performance Comparison on TimelineKGQA To evaluate the generalization of RTQA, we compare it with the RAG baseline on the TimelineKGQA dataset, focusing on questions of varying complexity: Simple, Medium, and Complex (see Table 2). RTQA achieves an overall Hits@1 of 0.298, outperforming RAG (0.235) by 27%. Its advantage becomes more pronounced as question complexity increases.

[5] https://huggingface.co/BAAI/bge-m3

[6] Baseline results are from (Qian et al., 2024).

Table 4: Experiment results of multi-granular time on Hits@1.

Model	Equal			Before/After			Equal Multi		
	Day	Month	Year	Day	Month	Year	Day	Month	Year
BERT	0.049	0.103	0.136	0.150	0.164	0.175	0.064	0.102	0.090
DistillBERT	0.041	0.087	0.113	0.160	0.150	0.186	0.096	0.127	0.089
ALBERT	0.069	0.082	0.132	0.221	0.277	0.308	0.103	0.144	0.144
EmbedKGQA	0.200	0.336	0.218	0.392	0.518	0.511	0.145	0.321	0.263
CronKGQA	0.425	0.389	0.331	0.375	0.474	0.450	0.295	0.333	0.251
MultiQA	0.445	0.393	0.350	0.379	0.548	0.525	0.308	0.321	0.283
RTQA	**0.916**	**0.959**	**0.967**	**0.842**	**0.898**	**0.787**	**0.729**	**0.758**	**0.578**

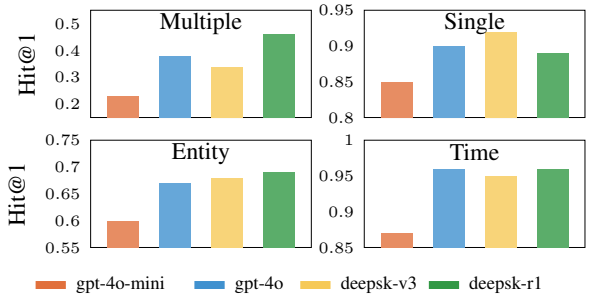

Figure 4: Hits@1 results with different LLMs.

On medium-complexity questions, RTQA scores 0.218 v.s. RAG's 0.092 (137% improvement); for complex questions, it reaches 0.135 vs. RAG's 0.009, marking a 1400% gain. These results highlight RTQA's strong capability in complex temporal reasoning, especially on multi-hop questions and those involving intricate time constraints, where traditional methods struggle.

5.3 Ablation Studies

To validate the effectiveness of different components in our proposed RTQA model, we conducted a series of ablation studies on the MultiTQ dataset. Table 3 presents the performance of various ablated versions of RTQA, where "w/o" indicates the removal of specific modules.

Impact of Question Decomposition We remove the temporal question decomposer module, processing questions directly without decomposition. As shown in Table 3, the result drops significantly, with the overall Hits@1 decreasing from 0.765 to 0.709. The impact is particularly pronounced for `Multiple` questions, where performance drops dramatically from 0.424 to 0.214 (a 49.5% reduction). This substantial decrease confirms that recursive decomposition is crucial for handling complex temporal reasoning that involves multiple hops or combined temporal constraints.

Table 5: Characteristics of $\mathcal{T}$ and API efficiency.

	MultiTQ	TimelineKGQA
Avg Depth	1.37	1.57
Avg Branch	1.60	1.81
Avg API Call	3.96	5.38

Impact of Multi-Answer Strategy We eliminated the answer aggregator module and evaluated the variant *w/o multi-answer*, which relies solely on answers derived from sub-questions without incorporating alternative sources. The results show an overall performance drop of 1.7%, with more pronounced declines for `Multiple` questions, which decreased by 19.6%, and `Entity` answers, which dropped by 3.6%. These findings highlight the effectiveness of the multi-answer module in reducing error propagation by offering alternative reasoning paths when sub-question inference fails or yields inaccurate results.

Impact of Fact Retrieval We examined the variant "w/o fact retrieval" that removes external knowledge from TKG, The results reveal a catastrophic performance degradation, with overall Hits@1 plummeting from 0.765 to a mere 0.070. The magnitude of this performance collapse underscores the fundamental importance of accurate fact retrieval in TKGQA. Without access to reliable factual information, even the most sophisticated reasoning frameworks cannot produce accurate answers, as they lack the necessary evidence base for their inferences.

5.4 Further Experimental Analysis

Multi-Granular Time Analysis To verify the effectiveness of the model on multi-granularity temporal reasoning, we compared RTQA's performance across different time granularities (day, month, year) and temporal question type (Equal, Before/After, Equal Multi).[7] Table 4 demonstrates

[7]Baseline results are sourced from (Chen et al., 2023).

that RTQA consistently outperforms all baseline models across all temporal granularities and reasoning types. The consistent superior performance demonstrates that our recursive question decomposition approach and multi-answer strategy work effectively regardless of temporal scale, making RTQA a robust solution for diverse temporal reasoning applications.

Generalizability across different LLMs To evaluate the adaptability of RTQA across different LLMs, we conducted experiments using `gpt-4o-mini`, `gpt-4o`, `deepseek-v3`, and `deepseek-r1`. Given the large size of the test set, we randomly sampled 1,000 questions for this study. To ensure consistent input across models, we fixed the question decomposition outputs by using `gpt-4o-mini` for all decomposition steps, and applied different LLMs only in the *Recursive Solver* stage. As shown in Figure 4, models with stronger inherent reasoning abilities achieve significantly better results, particularly on complex temporal questions. These results demonstrate the strong generalizability of RTQA, which can effectively integrate with various LLMs in a plug-and-play manner, consistently outperforming baseline models across multiple dimensions.

Efficiency Analysis We evaluate RTQA efficiency using test questions from MultiTQ and TimelineKGQA, measuring `Avg Depth`, `Avg Branch`, and `Avg API Call`. As shown in Table 5, MultiTQ questions are simpler (depth: 1.37, branch: 1.60) than those in TimelineKGQA (depth: 1.57, branch: 1.81). MultiTQ requires 3.96 API calls on average, while TimelineKGQA requires 5.38 due to more extensive answer aggregation. These results show that RTQA operates efficiently with low overhead across different question complexities.

Effect of Context Limits We analyzed the impact of context length on answer accuracy and completeness. The hyperparameter `n` controls the number of top-ranked facts retrieved as context. As shown in Table 6, Recall@n increases monotonically with larger `n`. However, Hits@1 first improves and then drops, as excessive irrelevant information introduces noise and impedes LLM reasoning. Given the nearly half a million candidate facts, a sufficiently large context length is necessary. In practice, we observed that setting `n`=50 achieves the best accuracy.

Table 6: Effect of Context Limits on Answer Quality

Context Length	Hits@1	Recall@n
n=10	70.4%	53.45%
n=20	73.8%	62.42%
n=30	76.0%	66.44%
n=40	76.2%	69.54%
n=50	77.8%	71.78%
n=60	76.0%	73.95%

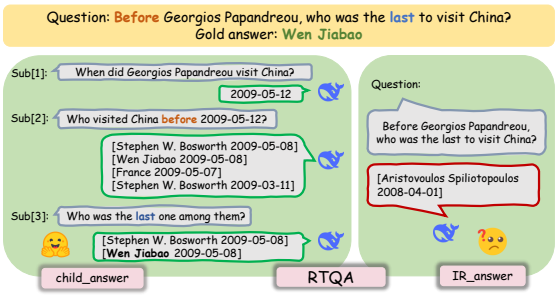

Figure 5: Case study of RTQA.

5.5 Case Study

Figure 5 compares two reasoning strategies of RTQA on the same question: solving via direct reasoning and solving via recursive sub-question decomposition. The comparison highlights the critical role of the question decomposition module in helping the model understand complex temporal constraints and generate reliable reasoning paths. On the right side of Figure 5, RTQA fails to handle temporal constraints such as *"before"* and *"last"*, resulting in hallucinated answers (highlighted with red boxes). In contrast, the left side demonstrates how RTQA decomposes the question into three sub-questions and recursively solves them step by step, ultimately arriving at the correct answer.

Error Analysis Our error analysis highlights five key issues affecting performance: (1) **Evaluation errors**, where predictions using aliases of the gold answer are incorrectly marked as wrong despite normalization; (2) **Annotation errors**, caused by gold answers annotated as `None`, rendering evaluations invalid; (3) **Retrieval errors**, due to irrelevant or missing facts, leading to reasoning failures; (4) **Temporal reasoning failures**, where complex constraints cause LLM hallucinations; and (5) **Decomposition errors**, resulting from illogical or unexecutable sub-question formats. These issues underscore the need to improve evaluation protocols, annotation quality, retrieval precision, temporal reasoning, and question decomposition.

6 Conclusion

We present **RTQA**, a training-free TKGQA framework that tackles complex temporal queries by recursively decomposing questions into sub-questions and reasoning bottom-up with TKG knowledge. Its multi-path answer aggregation mitigates error propagation, ensuring robust performance. Experiments on MultiTQ and TimelineKGQA benchmarks demonstrate significant Hits@1 improvements in "*Multiple*" and "*Complex*" categories, outperforming state-of-the-art methods. RTQA's plug-and-play design enhances compatibility with various LLMs, offering broad applicability. Future work will explore optimized decomposition and extensions to other knowledge graph domains, advancing efficient temporal question answering.

Limitations

Despite the strong performance of RTQA, several limitations remain that warrant further improvement. Firstly, the effectiveness of question decomposition heavily depends on the capabilities of the underlying LLM. Smaller models may struggle to generate high-quality sub-questions, thereby constraining the performance of the recursive solving process. Secondly, RTQA relies on a robust retriever to gather relevant TKG facts. Failure to retrieve key information can significantly reduce the reasoning accuracy of the LLM. Lastly, our method is primarily tailored for complex temporal knowledge graph question answering, and its applicability to other QA domains has yet to be thoroughly validated. Future work should focus on enhancing model adaptability across different LLMs and question domains, improving retrieval performance, and extending the framework to broader QA tasks.

Ethics Statement

In this paper, we investigate temporal knowledge graph question answering (TKGQA), focusing on complex reasoning over structured temporal data. Our method is developed and evaluated using publicly available and widely used datasets, including MultiTQ and TimelineKGQA. These datasets are constructed from open sources and do not contain any sensitive or personally identifiable information. Therefore, we believe that our work does not pose any ethical concerns.

Acknowledgements

This work is founded by National Natural Science Foundation of China (NSFC62306276/NSFCU23B2055/NSFCU19B2027), Zhejiang Provincial Natural Science Foundation of China (No. LQ23F020017), Yongjiang Talent Introduction Programme (2022A-238-G), and Fundamental Research Funds for the Central Universities (226-2023-00138). This work was supported by Ant Group.

References

James F. Allen. 1984. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154.

Amos Azaria, Rina Azoulay, and Shulamit Reches. 2024. Chatgpt is a remarkable tool—for experts. *Data Intelligence*, 6(1):240–296.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *ACL (1)*, pages 6101–6119. Association for Computational Linguistics.

Shulin Cao, Jiajie Zhang, Jiaxin Shi, Xin Lv, Zijun Yao, Qi Tian, Lei Hou, and Juanzi Li. 2023. Probabilistic tree-of-thought reasoning for answering knowledge-intensive complex questions. In *EMNLP (Findings)*, pages 12541–12560. Association for Computational Linguistics.

Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. *Data Intelligence*, 6(3):587–620.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *CoRR*, abs/2402.03216.

Zhuo Chen, Zhao Zhang, Zixuan Li, Fei Wang, Yutao Zeng, Xiaolong Jin, and Yongjun Xu. 2024b. Self-improvement programming for temporal knowledge graph question answering. In *LREC/COLING*, pages 14579–14594. ELRA and ICCL.

Ziyang Chen, Dongfang Li, Xiang Zhao, Baotian Hu, and Min Zhang. 2024c. Temporal knowledge question answering via abstract reasoning induction. In *ACL (1)*, pages 4872–4889. Association for Computational Linguistics.

Ziyang Chen, Jinzhi Liao, and Xiang Zhao. 2023. Multi-granularity temporal question answering over knowledge graphs. In *ACL (1)*, pages 11378–11392. Association for Computational Linguistics.

Ziyang Chen, Xiang Zhao, Jinzhi Liao, Xinyi Li, and Evangelos Kanoulas. 2022. Temporal knowledge graph question answering via subgraph reasoning. *Knowl. Based Syst.*, 251:109134.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Wentao Ding, Hao Chen, Huayu Li, and Yuzhong Qu. 2022. Semantic framework based query generation for temporal question answering over knowledge graphs. In *EMNLP*, pages 1867–1877. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *CoRR*, abs/2401.08281.

Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. 2024. Two-stage generative question answering on temporal knowledge graph using large language models. In *ACL (Findings)*, pages 6719–6734. Association for Computational Linguistics.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*, pages 4816–4821. Association for Computational Linguistics.

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. TEQUILA: temporal question answering over knowledge bases. In *CIKM*, pages 1807–1810. ACM.

Zhen Jia, Philipp Christmann, and Gerhard Weikum. 2024. Faithful temporal question answering over heterogeneous sources. In *WWW*, pages 2052–2063. ACM.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *ICLR*. OpenReview.net.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Yonghao Liu, Di Liang, Mengyu Li, Fausto Giunchiglia, Ximing Li, Sirui Wang, Wei Wu, Lan Huang, Xiaoyue Feng, and Renchu Guan. 2023. Local and global: Temporal question answering via information fusion. In *IJCAI*, pages 5141–5149. ijcai.org.

Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N. Ioannidis, Adesoji Adeshina, Phillip Ryan Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. 2022. Tempoqr: Temporal question reasoning over knowledge graphs. In *AAAI*, pages 5825–5833. AAAI Press.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh K. Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, G. P. Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, and 6 others. 2021. SYGMA: system for generalizable modular question answering overknowledge bases. *CoRR*, abs/2109.13430.

Xinying Qian, Ying Zhang, Yu Zhao, Baohang Zhou, Xuhui Sui, Li Zhang, and Kehui Song. 2024. Timer4 : Time-aware retrieval-augmented large language models for temporal knowledge graph question answering. In *EMNLP*, pages 6942–6952. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Apoorv Saxena, Soumen Chakrabarti, and Partha P. Talukdar. 2021. Question answering over temporal knowledge graphs. In *ACL/IJCNLP (1)*, pages 6663–6676. Association for Computational Linguistics.

Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL*, pages 4498–4507. Association for Computational Linguistics.

Aditya Sharma, Apoorv Saxena, Chitrank Gupta, Seyed Mehran Kazemi, Partha P. Talukdar, and Soumen Chakrabarti. 2023. Twirgcn: Temporally weighted graph convolution for question answering over temporal knowledge graphs. In *EACL*, pages 2041–2052. Association for Computational Linguistics.

Qiang Sun, Sirui Li, Du Huynh, Mark Reynolds, and Wei Liu. 2025. Timelinekgqa: A comprehensive question-answer pair generator for temporal knowledge graphs. *CoRR*, abs/2501.04343.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Suifeng Zhao, Tong Zhou, Zhuoran Jin, Hongbang Yuan, Yubo Chen, Kang Liu, and Sujian Li. 2024. Awecita: Generating answer with appropriate and well-grained citations using llms. *Data Intelligence*, 6(4):1134–1157.

A Dataset Details

A.1 MULTITQ

MultiTQ is the largest known TKGQA dataset, constructed from the ICEWS05-15 dataset (García-Durán et al., 2018), and contains 500K unique question-answer pairs. In addition, MULTITQ features multiple temporal granularities, including years, months, and days, with questions spanning over 3,600 days. The distribution of questions across categories is shown in Table 7.

A.2 TIMELINEKGQA

TimelineKGQA is an open-source automated QA pair generator for temporal knowledge graphs. Using TimelineKGQA, (Sun et al., 2025) creates two benchmark datasets from the ICEWS Coded Event Data (Boschee et al., 2015)(Time Range) and CronQuestion knowledge graph(Time Point) for demonstrating the question difficulty aligns with complexity categorization. The distribution of questions across categories is shown in Table 8.

Table 7: Statistics of question categories in MULTITQ.

Category		Train	Dev	Test
Single	Equal	135,890	18,983	17,311
	Before/After	75,340	11,655	11,073
	First/Last	72,252	11,097	10,480
Multiple	Equal Multi	16,893	3,213	3,207
	After First	43,305	6,499	6,266
	Before Last	43,107	6,532	6,247
	Total	386,787	587,979	54,584

Table 8: Statistics of question categories in TimelineKGQA.

Source KG		Train	Val	Test
CronQuestionKG	Simple	7,200	2,400	2,400
	Medium	8,252	2,751	2,751
	Complex	9,580	3,193	3,193
	Total	25,032	8,344	8,344

B Case Study Details

To illustrate how RTQA decomposes a complex temporal question and recursively solves it to obtain the correct answer, we analyze the question: "Before Georgios Papandreou, who was the last to visit China?" The process involves breaking the question into sub-questions, solving each recursively, and aggregating the results. Tables 9, 10, 11, 12 detail the reasoning process for each sub-question and the root node.

C Prompts

C.1 Prompts for Temporal Question Decomposer

In the MultiTQ dataset, temporal questions are divided into **simple** and **multiple** categories based on complexity. The **simple** category includes *equal*, *first_last*, and *before_after*, while **multiple** comprises *equal multi*, *before_last*, and *after_first*. We designed category-specific prompts to guide the LLM in effective question decomposition. Figure 6 presents the prompts for *Simple*, including instructions and examples. Figure 7 shows the prompt for *Multiple*. Following the decomposition guidelines in (Cao et al., 2023), we adapted prompts for temporal scenarios, using manually crafted question-answer pairs from validation set.

C.2 Prompts for Recursive Solver

Figure 8 shows the prompt used in the initial step of the recursive solving process. The prompt provides the large language model (LLM) with the original complex temporal question and historical facts retrieved from the temporal knowledge graph (TKG). The LLM is tasked with reasoning over these facts to either decompose the question into sub-questions or directly provide an answer if the question is simple enough. For example, for the question "Who was the president of the United States when Barack Obama became a senator?", the prompt includes historical facts such as Barack Obama's timeline (e.g., "Barack Obama became a senator in 2005") and U.S. presidential terms (e.g., "George W. Bush was president from 2001 to 2009"), enabling the LLM to perform temporal reasoning.

Figure 9 depicts the prompt used in a subsequent step of the recursive solving process. The prompt supplies the LLM with the original question (or a sub-question) and relevant facts determined from the previous sub-question's answer, asking the LLM to make the most accurate choice for the current step. This builds on the recursive decomposition by leveraging prior answers to resolve temporal dependencies. For instance,

Table 9: Reasoning process for sub-question idx 0.

Field	Content
idx	0
question_text	When did Georgios Papandreou visit China?
fa	3
question	When did Georgios Papandreou visit China?
IR_answer	2009-05-12
Historical Facts	Georgios Papandreou made a visit to China on 2009-05-12. China hosted a visit from Georgios Papandreou on 2009-05-12. Georgios Papandreou expressed intent to meet or negotiate with China on 2009-05-12. Georgios Papandreou made a visit to France on 2005-02-11. Georgios Papandreou made a visit to France on 2010-03-05. Georgios Papandreou made a visit to France on 2011-05-28. Georgios Papandreou made a visit to France on 2011-03-19. Georgios Papandreou made a visit to France on 2010-02-10.
answer	2009-05-12

for the question "Which country was the last one among them?", if the previous sub-question "List the countries and their independence dates: [France: 1789, Germany: 1871, Japan: 1945]" yields these facts, the prompt provides this data, and the LLM returns "Japan" as the country with the latest independence date (1945).

C.3 Prompt for Answer Aggregator

Figure 10 shows the instruction and examples for answer aggregation.

Table 10: Reasoning process for sub-question idx 1.

Field	Content
idx	1
question_text	Who visited China before #1?
fa	3
question	Who visited China before 2009-05-12?
IR_answer	[Stephen W. Bosworth 2009-05-08], [Wen Jiabao 2009-05-08], [France 2009-05-07], [Stephen W. Bosworth 2009-03-11]
Historical Facts	South Korea hosted a visit from China on 2009-05-12. Stephen W. Bosworth made a visit to China on 2009-05-13. Lawrence Cannon made a visit to China on 2009-05-12. China made a visit to South Korea on 2009-05-12. Abdullah Gül hosted a visit from China on 2009-05-12. Stephen W. Bosworth made a visit to China on 2009-05-08. Kuomintang made a visit to China on 2009-05-27. Lawrence Cannon made a visit to China on 2009-05-13. Stephen W. Bosworth made a visit to China on 2009-05-15. Wen Jiabao made a visit to China on 2009-05-09. Kuomintang made a visit to China on 2009-05-26. China hosted a visit from Iran on 2009-10-15. Wen Jiabao made a visit to China on 2009-05-08. China hosted a visit from France on 2009-05-07. Ma Biao made a visit to China on 2009-07-05. Georgios Papandreou made a visit to China on 2009-05-12. China made a visit to Kazakhstan on 2009-06-12. Abhisit Vejjajiva made a visit to China on 2009-06-15. Wen Jiabao made a visit to China on 2009-05-28. Wu Po-hsiung made a visit to China on 2009-05-25. Eric Chu made a visit to China on 2009-05-17. Barack Obama made a visit to China on 2009-10-12. Abhisit Vejjajiva made a visit to China on 2009-06-25. Xi Jinping made a visit to China on 2009-06-22. China hosted a visit from the Russian military on 2009-07-11.
answer	[Stephen W. Bosworth 2009-05-08], [Wen Jiabao 2009-05-08], [France 2009-05-07], [Stephen W. Bosworth 2009-03-11]

Table 11: Reasoning process for sub-question idx 2.

Field	Content
idx	2
question_text	Who was the last one among them?
fa	3
question	Who was the last one among them?
IR_answer	[Stephen W. Bosworth 2009-05-08], [Wen Jiabao 2009-05-08]
Relevant Facts	[Stephen W. Bosworth 2009-05-08] [Wen Jiabao 2009-05-08] [France 2009-05-07] [Stephen W. Bosworth 2009-03-11]
answer	[Stephen W. Bosworth 2009-05-08], [Wen Jiabao 2009-05-08]

Table 12: Reasoning process for the root node (original question).

Field	Content
idx	3
question_text	Before Georgios Papandreou, who was the last to visit China?
sons	0,1,2
gold_answer	Wen Jiabao
question	Before Georgios Papandreou, who was the last to visit China?
IR_answer	[Aristovoulos Spiliotopoulos 2008-04-01]
child_answer	[Stephen W. Bosworth 2009-05-08], [Wen Jiabao 2009-05-08]
Historical Facts	Georgios Papandreou made a visit to China on 2009-05-12. China hosted a visit from Georgios Papandreou on 2009-05-12. Georgios Papandreou expressed intent to meet or negotiate with China on 2009-05-12. Wen Jiabao made a visit to Georgios Papandreou on 2010-10-10. Georgios Papandreou hosted a visit from Wen Jiabao on 2010-10-10. Georgios Papandreou made a visit to France on 2010-03-05. Georgios Papandreou made a visit to France on 2005-02-11. Georgios Papandreou made a visit to France on 2011-05-28. Georgios Papandreou made a visit to France on 2011-03-19. Georgios Papandreou made a visit to France on 2010-03-04. Georgios Papandreou made a visit to France on 2010-02-10. France hosted a visit from Georgios Papandreou on 2005-02-11. Georgios Papandreou made a visit to France on 2010-03-07. Georgios Papandreou made a visit to France on 2010-03-06. Georgios Papandreou made a visit to France on 2006-11-26. Georgios Papandreou made a visit to France on 2009-04-24. Georgios Papandreou made a visit to France on 2011-11-01. Middle East made a visit to Georgios Papandreou on 2008-07-01. France hosted a visit from Georgios Papandreou on 2010-03-05. France hosted a visit from Georgios Papandreou on 2010-02-10. Antanas Valionis made a visit to China on 2006-04-20. Georgios Papandreou made a visit to Iran on 2006-06-30. Aristovoulos Spiliotopoulos made a visit to China on 2008-04-01. Nicos Anastasiades made a visit to China on 2015-10-19. Nicos Anastasiades made a visit to China on 2015-10-18.
answer	[Stephen W. Bosworth 2009-05-08], [Wen Jiabao 2009-05-08]

Instruction:
Convert the following question into a JSON object where the question is the key and the value is an empty list. Do not include any explanation or extra text. Just return the JSON. Just return the modified question in JSON format with an empty list as its value.

Here are a few examples:

Q: Who visited France in 2009-05?
A: {"Who visited France in 2009-05?": []}

Q: When did Qatar pay a visit to Barack Obama?
A: {"When did Qatar pay a visit to Barack Obama?": []}

Q: Who applied for Iran in January 2010?
A: {"Who applied for Iran in 2010-01?": []}

Q: Which country negotiated with Japan on 19 April 2005?
A: {"Which country negotiated with Japan on 2002-04-19?": []}

Q: Who visited Japan in April 2012?
A: {"Who visited Japan in 2012-04?": []}

Q: In May 2009, who signed an agreement with Iran?
A: {"In 2009-05, who signed an agreement with Iran?": []}

Q: Who accused Iran in 2015?
A: {"Who accused Iran in 2015?": []}

Q: On 19 March 2006, who threatened Iran?
A: {"On 2006-03-19, who threatened Iran?": []}

Q: Who visited Guatemala on 7 July 2007?
A: {"Who visited Guatemala on 2007-07-07?": []}

Remaining examples ...

Q:

A:

Figure 6: Prompt example of RTQA for Temporal Question Decomposition, the category is **Simple** in MultiTQ.

Instruction:
You are an expert specializing in dealing with problems containing the keywords "before/after". You need to read the question carefully.
1.If the problem involves a situation like "before December 13, 2005" with a "before+ timestamp", there is no need to decompose the original problem. Just convert the question into a JSON object where the question is the key and the value is an empty list.
2.If the problem involves the situation of a "before+ entity" like "before Japan", the original problem needs to be decomposed into sub-problems. First, generate an explicit sub-question to determine the time (e.g., "When did Iran. . . ?"). When a sub-question is logically depends on the answer to a previous one, use placeholders (e.g., #1) to refer to that answer. Return a valid JSON object representing the question tree. Each key is a parent question, and its value is a list of sub-questions.

Here are a few examples:

Q: Who rejected Iran before the citizens of State Actor did?
A: {"Who rejected Iran before the citizens of State Actor did?": ["When did the citizens of State Actor reject Iran?", "Who rejected Iran before #1?"]}

Q: After Japan, who made South Korea suffer from conventional military forces?
A: {"After Japan, who made South Korea suffer from conventional military forces?": ["When did Japan make South Korea suffer from conventional military forces?", "Who make South Korea suffer from conventional military forces after #1?"]}

Q: Which country did Qatar appeal to after April 2011?
A: {"Which country did Qatar appeal to after 2011-04?": []}

Q: Before 14 October 2015, who made Burundi suffer from conventional military forces?
A: {"Before 2015-10-14, who made Burundi suffer from conventional military forces?": []}

Q: Who had a telephone conversation with Japan after November 2005?
A: {"Who had a telephone conversation with Japan after 2005-11?": []}

Q: Who negotiated with Colombia before 22 December 2010?
A: {"Who negotiated with Colombia before 2010-12-22?": []}

Q: With which country did Qatar sign formal agreements before 15 January 2008?
A: {"With which country did Qatar sign formal agreements before 2008-01-15?": []}

Q: After November 2007, who wanted to engage in diplomatic cooperation with Timor-Leste?
A: {"After 2007-11, who wanted to engage in diplomatic cooperation with Timor-Leste?": []}

Q: Before 24 January 2005, who wanted to establish diplomatic cooperation with the Kuomintang?
A: {"Before 2005-01-24, who wanted to establish diplomatic cooperation with the Kuomintang?": []}

Q: Who negotiated with Bolivia after June 2007?
A: {"Who negotiated with Bolivia after 2007-06?": []}

Remaining examples ...

Q:

A:

Figure 7: Prompt example of RTQA for Temporal Question Decomposition, the category is **Multiple** in MultiTQ.

Instruction:
Based on the historical facts, please answer the given question clearly in the following format: ...So the answer is: <final concise answer>.
1.If the question asks for a specific year (e.g., "Which year", "In which year", "the exact year", etc.), then return the answer in "yyyy" format. Just return the most appropriate timestamp as the answer.
2.If the question asks for a specific month (e.g., "Which month", "In what month", "the exact month", etc.), then return the answer in "yyyy-mm" format, including the year and the month. Just return the most appropriate timestamp as the answer.
3.If the question asks for a specific date (e.g., contains keywords like "When", "What day", "the exact date", etc.), return the answer in "yyyy-mm-dd" format. Just return the most appropriate timestamp as the answer.
4.If the question asks for a set of entities (e.g., contains keywords like "who", "which country", etc.), and multiple sources in the context offer valid answers, return the union of all correct, non-duplicate entities and attached timestamp in a list format.

Here are a few examples:

Historical facts: Barack Obama Reject Party Member (United Kingdom) 2008-09-23.
Barack Obama Reject Party Member (United Kingdom) 2008-09-23.
Barack Obama Make statement Party Member (United Kingdom) 2008-11-08.
Barack Obama Make statement Party Member (United Kingdom) 2008-11-08.
Barack Obama Express intent to meet or negotiate Party Member (United Kingdom) 2009-03-10.
Zawahiri Reject Barack Obama 2009-08-04.
Question: In which year did Barack Obama reject the party member of United Kingdom?
Answer: The rejection event occurred on 2008-09-23, so the year is 2008. So the answer is: 2008.

Historical facts: Media Personnel (Somalia) Praise or endorse Cabinet / Council of Ministers / Advisors (Somalia) 2012-11-27.
Media Personnel (Somalia) Praise or endorse Cabinet / Council of Ministers / Advisors (Somalia) 2015-01-12.
Media Personnel (Somalia) Make statement African Union 2007-10-08.
Media Personnel (Somalia) Make statement African Union 2007-01-17.
Media Personnel (Somalia) Make statement African Union 2012-11-18.
Cabinet / Council of Ministers / Advisors (Somalia) Praise or endorse Media Personnel (Somalia) 2011-11-28.
Media Personnel (Somalia) Make statement African Union 2007-06-15.
Question: When did Somalia's media personnel first commend Somalia's council of ministers?
Answer: We are asked to find the first time Somalia's media personnel commended (i.e., praised or endorsed) Somalia's Cabinet / Council of Ministers / Advisors. From the historical facts: Media Personnel (Somalia) Praise or endorse Cabinet / Council of Ministers / Advisors (Somalia) on: 2012-11-27 and 2015-01-12. Among these, the earliest instance is 2012-11-27. So the answer is: 2012-11-27.

Historical facts: Agence France-Presse Demand China in 2010-05-26.
Agence France-Presse Make an appeal or request China in 2007-01-08.
China Appeal for military aid Agence France-Presse in 2008-03-26.
France Make an appeal or request China in 2012-06-05.
France Demand China in 2008-06-11.
Question: Could you tell me the exact month when Agence France-Presse appealed to China?
Answer: So the answer is: 2007-01

Remaining examples ...

Historical facts:

Question:

Answer:

Figure 8: Prompt example of RTQA for Recursive Solver. This prompt utilizes the triples in the TKG retrieved by the retriever as external knowledge to assist the LLM in solving the problem.

Instruction:
Based on the Relevant facts, please answer the given question clearly in the following format: ...So the answer is: <final concise answer>.
Each question provides a series of relevant facts, including "entity + timestamp" pairs. You need to choose the earliest or latest entity as the answer based on the order in which the events occurred.

Here are a few examples:

Relevant facts: ["China 2006-01-20", "China 2006-10-30", "Vietnam 2008-04-30"]
Question: Which country was the last one among them?
Answer: The last country among the relevant facts, based on the timestamps, is Vietnam. So the answer is: Vietnam 2008-04-30.

Remaining examples ...

Relevant facts:

Question:

Answer:

Figure 9: Prompt example of RTQA for Recursive Solver. This prompt is mainly used to solve the problem of choosing the best first/last solution among multiple candidate answers.

Instruction:
You are given a question and multiple candidate answers from sources A, B, and C.
Follow these strict rules to choose the best answer: If only sources A and B are available, prefer B's answer unless it is "Unknown" or "Error", in which case choose A. If all three sources A, B, and C are available, prefer C's answer unless it is "Unknown" or "Error", then fall back to B, and if B is also invalid, fall back to A.

Here are a few examples:

Question: When did the citizens of Africa express their intention to establish diplomatic cooperation with Vietnam?
Candidate answer:
source A: 2012-09-04
source B: 2012-09-04
Source C: Unknown
Output: So the answer is: 2012-09-04

Question: Who was the first to praise Juan Carlos I after 2006-02-22?
Candidate answer:
source A: Jorge Briz Abularach
source B: Unknown
Source C: House of Representatives (Uruguay)
Output: So the answer is: House of Representatives (Uruguay)

Question: Who rejected the Prime Minister of India after 2012-01-03?
Candidate answer:
source A: Sri Lanka
source B: China
Output: So the answer is: China

Remaining examples ...

Question:

Candidate answer:

Output:

Figure 10: Prompt example of RTQA for Answer Aggregator.