GoT-R: Enhancing Large Language Models for Complex Question Answering with Graph-of-Thought Guided Reasoning

Peixuan Huang[1], Bohan Li[1,2,3] ✉, Wenlong Wu[1], Haofen Wang[4] ✉, Mengfei Xu[1], Chen Chen[1], Lei Liang[5], and Meng Wang[4]

[1] College of Artificial Intelligence & Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2] Ministry of Industry and Information Technology, Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
[3] National Engineering Laboratory for Integrated Aero-Space-Ground Ocean Big Data Application Technology, Xi'an, China
[4] College of Design & Innovation, Tongji University, Shanghai, China
[5] Ant Group Knowledge Graph Team, Hangzhou, China
{peixuanh,bhli}@nuaa.edu.cn, carter.whfcarter@gmail.com

Abstract. Large language models (LLMs) have demonstrated significant potential in complex question answering (QA) by reasoning on knowledge graphs (KGs), which offer structured external knowledge. Existing methods for KG-augmented reasoning, including iterative and once paradigms, have enhanced LLMs' performance in complex QA and emphasized the significance of reasoning through KG relations. However, a significant challenge remains in constructing optimal reasoning plans from KG relations while maximizing their reasoning capabilities. To address this challenge, we propose **G**raph-**o**f-**T**hought Guided **R**easoning (GoT-R), which enhances LLMs for complex QA. GoT-R integrates three components: (i) atomic relations selection, which uses an encoder to identify relevant relations; (ii) graph-of-thought (GoT) construction, which builds a graph-of-thought combining KG-aligned relations and LLM's inherent knowledge; and (iii) GoT-guided hybrid retrieval and reasoning, which integrates relation-aware and relation-unaware retrieval to generate reliable and comprehensive reasoning evidence. By leveraging KG relations and LLM knowledge, GoT-R achieves faithful and interpretable reasoning while reducing reliance on high-quality QA datasets. Experimental results on four complex QA datasets demonstrate GoT-R's effectiveness and generalizability. The source code for this project is available at https://github.com/Peixuan-Huang/GoT-R.

Keywords: Complex question answering · Knowledge graph · Large language model.

1 Introduction

The emergence of large language models (LLMs) has marked a transformative milestone in natural language processing (NLP) due to their emergent ability

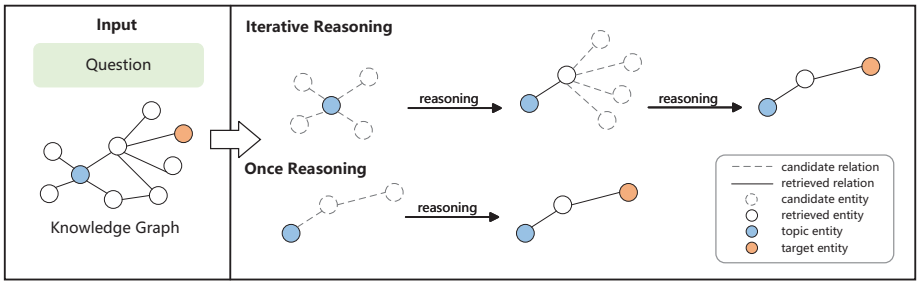

Fig. 1. The landscape of existing KG-augmented reasoning methods.

and generalizability [20]. Nevertheless, their limitations in complex question answering (QA), which requires advanced reasoning and domain knowledge, still remain due to their lack of domain-specific expertise, interpretability, and up-to-date knowledge.

Knowledge Graphs (KGs), with their structured and easy-to-update knowledge representation capabilities as well as minimal data redundancy, serve as effective solutions for overcoming LLMs' inherent limitations. KG-based retrieval-augmented methods[1,11] enhance LLM by retrieving triples from KGs but often overlook KG structure and introduce redundancy, limiting their effectiveness in complex QA. Enabling LLMs to reason on KGs to derive comprehensive and accurate knowledge is essential.

KG-augmented reasoning enables LLMs to simulate the human mind, offering a promising approach to address challenges of complex question answering. As illustrated in Fig. 1, the existing methods can be broadly categorized into iterative reasoning and once reasoning paradigms. Iterative reasoning facilitates dynamic interactions between LLMs and KGs [6,17,23,26]. Although it demonstrate impressive performance by iteratively retrieving knowledge and refining intermediate reasoning steps, it often suffers from high time complexity due to frequent LLM interactions and challenges in determining termination points. And mistakes in early iterations can cascade, leading to invalid subsequent reasoning and wasted resources. In contrast, once reasoning pre-plans reasoning paths or retrieves structured knowledge in a single query, reducing interaction costs [15,28]. Representative methods in iterative reasoning, such as ToG [23], utilize LLMs to select relations and entities from KGs, leveraging their inherent knowledge to synergistically enhance reasoning. Once reasoning methods like RoG [15] plans relation paths via model fine-tuning, enabling faithful and interpretable reasoning. These advanced approaches highlight the importance of the relations in KGs during reasoning and the prospect of harnessing the knowledge inherent in the LLMs. However, LLMs in ToG may favor certain relations, overlooking more relevant ones due to training data bias. RoG completely relies on relation paths for reasoning, limiting its handling of complex questions with implicit relations [14]. Moreover, it relies on fine-tuning with labeled QA datasets, reducing its adaptability in real-world scenarios with limited anno-

tations [9]. Therefore, constructing optimal reasoning plans from KG relations while maximizing their reasoning capabilities remains a significant challenge.

To address this challenge, we propose a novel framework for complex question answering called **G**raph-**o**f-**T**hought guided **R**easoning (GoT-R). GoT-R integrates three key components: (i) atomic relations selection, (ii) graph-of-thought construction and (iii) graph-of-thought guided hybrid retrieval and reasoning. The first component is an end-to-end module that uses LLMs to create an atomic question-relation training set, followed by training an encoder using prototype contrastive learning to select relevant atomic relations. Next, GoT-R prompts the LLM with in-context learning to construct a graph-of-thought, incorporating entities from the question and LLM's inherent knowledge, along with KG-aligned relations. Finally, this graph guides GoT-R in combining relation-aware and relation-unaware retrieval to generate reliable and comprehensive reasoning evidence, improving the LLM's reasoning accuracy. GoT-R framework adopts a once reasoning paradigm, reducing LLM calls. It pre-filters candidate atomic relations aligned with the question's intent, improving accuracy and flexibility for GoT construction. Guided by GoT, GoT-R can leverage both the structured knowledge from KGs and the inherent knowledge of the LLM to further enhance reasoning performance. Additionally, GoT-R eliminates the need for high-quality QA datasets, increasing the LLM's generalizability across different KGs.

The main contributions of this paper are as follows:

- We propose GoT-R, which combines relation-aware and relation-unaware retrieval and reasoning guided by graph-of-thought. It aligns with question intent and provides structured and comprehensive knowledge, thus enable more accurate reasoning, mitigate the factual hallucination of LLMs and improves interpretability.
- We introduce an atomic relations selection module, which leverages LLMs' language generation capability and prototype contrastive learning. This enables GoT-R to flexibly construct a more accurate GoT and integrate with different KGs, augmenting LLM with ontology reasoning capabilities to support question answering in various domains.
- Comprehensive experiments on four complex QA datasets demonstrate the effectiveness and generalization ability of GoT-R.

2 Related Work

2.1 Knowledge Graph-Augmented LLM Reasoning

Recent graph learning advances in contrastive frameworks [32], knowledge hypergraph completion [13], and dynamic recommendations [32] demonstrate structured data modeling potential. Existing approaches for knowledge graph-augmented reasoning in LLMs generally retrieve graphical elements [8] from external KGs to support reasoning and can be categorized into two paradigms: iterative reasoning and once reasoning. Iterative reasoning incrementally executes reasoning steps over the KG. For example, KD-CoT [26] incorporates KG knowledge to

validate and refine chain-of-thought (CoT) reasoning paths, alleviating hallucination and error propagation. StructGPT [6] introduces LLM-based agents that autonomously retrieve and reason on graphs via pre-defined interfaces. Similarly, ToG [23] iteratively prompts LLMs to explore related entities and relations until the question can be answered to extract the reasoning paths. ToG enhances LLM reasoning by integrating KGs and leveraging LLMs' inherent knowledge but risks overlooking relevant relations due to training data bias. Based on ToG, ToG2.0 [17] enhances efficiency by restricting the retrieval corpus using entities encountered during exploration. While iterative reasoning improves accuracy through incremental refinement, it often suffers from high time complexity, unclear stopping criteria, and error accumulation, leading to inefficient or invalid reasoning. In contrast, once reasoning pre-plans the reasoning process before interacting with the KG. For instance, MindMap [28] employs a prompting pipeline where LLMs reason over extracted KG subgraphs to generate answers aligned with KG-based reasoning paths. RoG [15] fine-tunes LLMs to generate relation paths and uses a BFS retriever to search KG reasoning paths. This paradigm offers lower complexity and faster response. While RoG shows impressive performance, its dependence on gold-standard annotations reduces its broader applicability [9]. In this work, GoT-R framework introduces atomic relation selection and graph-of-thought guided hybrid retrieval and reasoning module to make LLM reason more effectively by retrieving comprehensive reasoning evidence in a once reasoning paradigm and generalize across diverse KGs, thereby enhancing their ability to answer complex questions.

2.2 Graph-of-thought

Graph-of-Thought (GoT) models human thinking by representing thought units as nodes and their connections as edges, reflecting the non-sequential nature of thought [30]. [30] enhances deductive reasoning in LMs through a two-stage framework with a GoT encoder and gated fusion. GoT has also been applied to mind map elicitation and generating answers in graph form, and to model LLM-generated thoughts as graphs for cohesive outcomes [2,28]. ReX-GoT [33] further mimics human reasoning in a three-step process to find the optimal path. The structured form of GoT benefits knowledge retrieval and integration.

3 Methodology

3.1 Overview of GoT-R

As shown in Fig. 2, GoT-R is an end-to-end once reasoning framework for complex question answering, consisting of three components: atomic relations selection, graph-of-thought construction, and graph-of-thought guided hybrid retrieval and reasoning. First, the model begins with atomic relation selection, where an atomic question-relation training set is constructed utilizing an LLM. A specially designed encoder is then trained through prototype contrastive learning

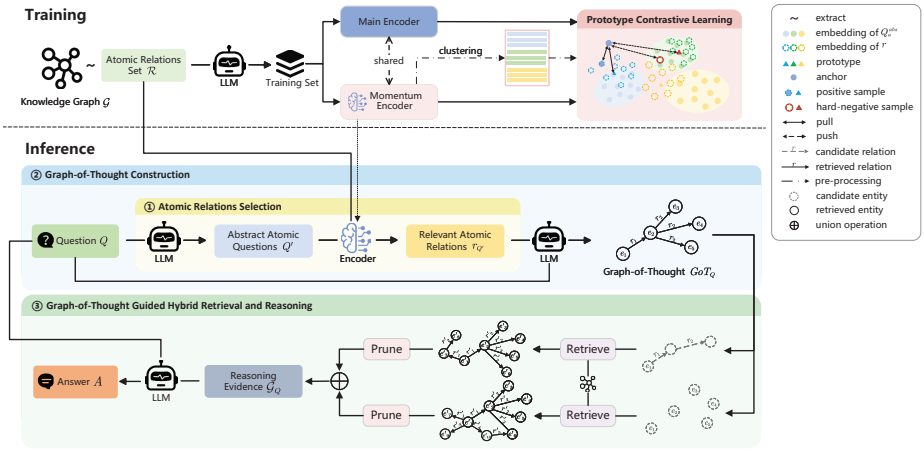

Fig. 2. The illustration of GoT-R framework.

with label-aware hard-negative sampling strategy to identify relevant relations for each atomic question, ensuring that the selected relations are closely aligned with the input question. Next, GoT-R flexibly constructs a graph-of-thought which represents the mindmap to solve the question based on the selected atomic relations via in-context learning. This graph incorporates both relations from the knowledge graph and entities from LLM's inherent knowledge, creating a practicable reasoning plan. Finally, GoT-R employs both relation-aware and relation-unaware retrieval strategies, retrieving relevant and comprehensive evidence from the KG for reasoning. GoT-R improves LLMs' reasoning performance with fewer LLM calls and adapts LLMs to various domains.

3.2 Atomic Relations Selection

Relations in a KG are reliable and can serve as sustainable guidance for reasoning [15], but using a LLM to select relevant relations from all relations at once is challenging without fine-tuning and partitioning. To address this, we propose a method for selecting atomic relations, using an encoder to identify relevant relations for each atomic question derived from question decomposition. These selected relations help the LLM build a more accurate graph-of-thought. This approach involves three steps: training data construction, encoder training and relations selection.

Training Data Construction GoT-R leverages LLMs to generate a training set that takes advantage of their robust generative capabilities. Given a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities and relations, respectively, and $|\mathcal{E}| = n, |\mathcal{R}| = m$. $\mathcal{T} = \{(e_s, r, e_o)|e_s, e_o \in \mathcal{E}, r \in \mathcal{R}\}$ represents the set of knowledge triples, each contains a head entity e_s, a tail

entity e_o, and a relation r. In general, the atomic relations set of $\mathcal{G}$ is $\mathcal{R}$, where $\mathcal{R} = \{r_1, r_2, ..., r_m\}$. GoT-R randomly samples k_1 triples for each atomic relation r_i(where $1 \leq i \leq m$). It then utilize LLMs to generate k_2 atomic questions Q_a based on each triple to ensure diverse expression. Furthermore, GoT-R replaces the subject(i.e., the head entity) of each atomic question with "<topic entity>" to produce abstract atomic questions, denoted as Q_a^{abs}, for questions without a specific subject can represent the semantic of relations better. For each abstract atomic question $q(q \in Q_a^{abs})$, the positive atomic relation r^+ is the relation in the triple from which q is generated. A template for deriving Q_a is as follows:

Prompt for rewriting triple to questions

Given a triple of the form [head entity, relation, tail entity], rewrite the triple as a question where the head entity is the subject entity in the question and the tail entity is the answer to the question. Note that tail entity cannot appear in this question. Give $\{k_2\}$ different statements of the question as a list.

Encoder Training We employ prototype contrastive learning to align atomic question representations with their corresponding relation representations [13], preserving semantic similarities between relations in the knowledge graph to facilitate robust reasoning processes.

During the training of the sentence encoder E, we encode the input abstract atomic question $q(q \in Q_a^{abs})$ and the positive atomic relation r^+ using E. The embeddings of q and r^+ are h_q and h_{r^+}, respectively.

$$h_q = E(q),\ h_{r^+} = E(r^+). \tag{1}$$

GoT-R first clusters h_q into K categories $C_1, C_2, ..., C_K$, whose centroids are denoted by $c_1, c_2, ..., c_K$. To better capture the relevance of each atomic relation, GoT-R calculates the average of h_{r^+} in each cluster as the prototype, denoted as c_i^r(where $1 \leq i \leq K$).

We then introduce a label-aware hard negative sampling strategy that combines label information with hard negative mining in supervised contrastive learning, since randomly sampled negatives can include false negatives, hindering accurate representation learning. Previous studies have emphasized the importance of hard negative examples, which are more difficult to differentiate and help improve discriminative feature learning and faster convergence [7].

Specifically, for each anchor q in a given cluster C_i(where $1 \leq i \leq K$), GoT-R selects an atomic relation from each of the top N nearest clusters to sample true negative atomic relations as the hard-negative samples. It yields N negative samples for each input question q. Similarly, these top N nearest clusters are considered as the hard-negative prototypes for each anchor prototype c_i(where $1 \leq i \leq K$). By randomly selecting negative samples from N nearest cluster, GoT-R reduces the computational costs of calculating distance while ensuring diverse hardness of hard-negative samples. This approach promotes

learning global representations without biasing the model in a specific direction. Similar to [10], our contrastive loss function is defined as following:

$$\mathcal{L} = \sum_i -(log\frac{exp(sim(h_{q_i}, h_{r^+i})/\tau)}{\sum_{j=0}^{N} exp(sim(h_{q_i}, h_{rj})/\tau)} + log\frac{exp(sim(h_{q_i}, {c^r}_i^+)/\phi_i)}{\sum_{j=0}^{N} exp(sim(h_{q_i}, c_j^r)/\phi_j)}), \tag{2}$$

where h_{q_i} is an anchor sample of the batch, and every anchor has one positive sample h_{r^+i}. h_r denotes both the positive and negative samples, τ is a temperature parameter, and sim is the cosine similarity function. Similarly, every anchor has one positive prototype ${c^r}_i^+$. c^r denotes both the positive and the negative prototypes. ϕ denotes the concentration estimation, where a smaller ϕ indicates larger concentration. For a cluster $C, C \in C_1, C_2, ..., C_K$, its concentration can be defined as:

$$\phi = \frac{\sum_{d=1}^{D} \|h_{q_d} - c\|_2}{Dlog(D + \alpha)}. \tag{3}$$

Here, D represents the number of samples in cluster C, and c is the centroid of C. h_{q_d} demotes the momentum feature of an atomic question. The parameter α serves as a smooth parameter to ensure that small clusters do not yield an excessively large value of ϕ. Additionally, ϕ is normalized for each set of clusters such that they have a mean of τ.

Relations Selection GoT-R employs LLMs to decompose the input natural language question Q into abstract atomic questions $Q' = \{q_1, q_2, ..., q_z\}$. For various QA scenarios, we can design different prompt templates to facilitate this decomposition. An example is provided below:

Prompt for Question Decomposition

Break the <question> down into <atomic questions> according to subject-predicate-object as a list. If the <question> is already atomic, it can not be split. Finally, replace the subject of each <atomic question> by <topic entity> to generate <abstract atomic questions>.

Then, GoT-R uses the trained encoder E to select the top k_q relevant atomic relations for each $q_i \in Q'$(where $1 \leq i \leq z$) based on their cosine similarity.

3.3 Graph-of-Thought Construction

Previous methods [4,23] have improved LLMs' performance by leveraging LLMs' inherent knowledge and reasoning ability. Inspired by this and aiming to enabling more comprehensive knowledge reasoning, we propose prompting the LLM to construct a graph-of-thought that mimics a knowledge graph [13], containing relation paths and auxiliary knowledge to address the input question.

We prompt LLMs to construct a graph-of-thought GoT_Q based on the input question Q and all selected atomic relations $r_{Q'}$, where $|r_{Q'}| = z \cdot k_q$. GoT_Q

is built using Cypher for LLMs are more inclined to use continuous language for responses and have decent code capabilities [14]. Additionally, we employ a few-shot prompting strategy, which has proven effective in previous studies [4]. This approach leverages the LLM's in-context learning capability, enabling it to better adapt to the graph-of-thought construction task. The generated Cypher query is then parsed into triples. Formally, this can be represented as:

$$GoT_Q = LLM(r_{Q'}||Q) = (\mathcal{E}_{got}, \mathcal{R}_{got}, \mathcal{T}_{got}), \tag{4}$$

where $||$ represents concatenation using prompt template, $\mathcal{E}_{got}$ represents the set of entities, $\mathcal{R}_{got}$ represents the set of relations, and $\mathcal{T}_{got}$ denotes the set of triples constructed from the question Q and the selected atomic relations $r_{Q'}$.

By constructing a graph-of-thought to address the problem as a whole, GoT-R reduces errors from question decomposition. By incorporating both KG-aligned relations and LLM-derived knowledge, it provides a more precise reasoning plan and auxiliary knowledge for problem-solving. To avoid high time complexity, we use LLMs to identify the critical relation path rather than directly extracting paths from the graph. The prompt for constructing the graph-of-thought is as follows:

Prompt for Graph-of-Thought Construction

You should answer the {Question} in the following steps:
<step 1> Find out what {Knowledge Planing} do you need to solve the {Question}.
<step 2> Using the atomic relationships in {Possible Atomic Relationships}, strictly fill the {Knowledge Planing} to construct the {Knowledge Graph} as complete as possible with (Cypher) with your own knowledge.
<step 3> Strictly complete the {Knowledge Graph} to construct {Completed Knowledge Graph} to include more detailed reasoning paths that solve the {Question}.
<step 4> Based on the {Completed Knowledge Graph}, generate valid {Relation Paths} that can be helpful for answering the {Question}.

3.4 Graph-of-Thought Guided Hybrid Retrieval and Reasoning

To fully leverage the potential of GoT, we propose a hybrid relation-aware and relation-unaware retrieval method for reasoning. Relation-aware retrieval considers the structured connections in GoT and the semantic of relations, ensuring precise and contextually relevant information. Relation-unaware retrieval directly construct the reasoning paths based on entities, disregarding the semantic of relation. It enriches reasoning with auxiliary knowledge. Therefore, we integrating these methods to provide comprehensive knowledge coverage and enhance reasoning accuracy and interpretability for LLMs in complex QA tasks.

Relation-Aware Retrieval and Pruning The relation paths in GoT_Q represent the logically reasoning plan based on KG-aligned relations. GoT-R traverses

these paths by beam search to retrieve relevant knowledge from the KG. Specifically, for each relation path $path_r = [e_s, r_1, r_2, .., r_L]$ in GoT_Q, GoT-R first retrieves the matched head entities ε_s for e_s. For each matched entity in ε_s, it then retrieves the corresponding relations σ_1 for r_1:

$$\varepsilon_s = argtopm_{1 \leq i \leq n} sim(f(e_s), f(e_i)), \tag{5}$$

$$\sigma_i = argtopm_{1 \leq j \leq z} sim(f(r_i), f(r_j)), 1 \leq i \leq L, \tag{6}$$

where $f(\cdot)$ represents an encoder function based on a pre-trained language model, and z denotes the number of relations associated with a matched entity. Next, the tail entities ε_o for the triple $(e_s, r_1, ?)$ are obtained. By setting ε_o as the new ε_s and repeating the previous steps, a set of reasoning paths is generated. Each reasoning path is denoted as $path_{rs} = [e'_s, r'_1, e'_{o_1}, r'_2, e'_{o_2}, .., r'_L, e'_{o_L}]$.

To prune the reasoning paths, we define the score for each path as follows:

$$score_p = \sum_{i=1}^{L} sim(f(r_i), f(r'_i)). \tag{7}$$

The pruning strategy involves sorting the reasoning paths by $score_p$ and retaining the top K_1 paths $\mathcal{G}^r_Q$.

For the relation paths $path_r$ are generated by LLMs, we fix the triples for GoT_Q to ensure that no relevant relations are ignored. The process follows the same procedure as relation-aware retrieval and pruning, with the relation path defined as $path^{fix}_r = [e_s, r_1]$. We then retain the top K_2 triples as the final results $\mathcal{G}^t_Q$.

Relation-Unaware Retrieval and Pruning Concurrently, GoT-R extracts the path-based evidence graph and neighbor-based evidence graph [28] guided by GoT_Q. We align the entities in GoT_Q with $\mathcal{E}$ in the KG to create an candidate entity set $\mathcal{E}_{cand} = \{e_1, e_2, ..., e_M | e_i \in \mathcal{E}, 1 \leq i \leq M\}$. Candidate entities are constructed in pairs, and the l-hop shortest paths between them are recorded to obtain the path-based evidence graph:

$$\mathcal{G}^{path}_Q = shortestpath(e_i, e_j, l), 1 \leq i, j \leq M, i \neq j. \tag{8}$$

The neighbor-based evidence graph provides additional question-related evidence for $\mathcal{G}^{path}_Q$. In this step, GoT-R expands each node $e_i \in E_{cand}$ by incorporating its 1-hop neighbors, thereby adding triples (e_i, r, e_o) to the set $\mathcal{G}^{nei}_Q$. To handle potential noise in the knowledge, we use a reranking model, $Reranker(x, y, topk)$, which assesses a segment set x and selects the $topk$ ranked elements from y. The filtering process is formalized as:

$$\mathcal{G}^e_Q = Reranker(Q, \mathcal{G}^{path}_Q \oplus \mathcal{G}^{nei}_Q, topk), \tag{9}$$

where $|\mathcal{G}^e_Q| = topk$, $\oplus$ denotes union operation. Relation-unaware retrieval makes the implicit knowledge of LLM interpretable and expands the scope of knowledge considered in the subsequent reasoning process.

Evidence Combination and Reasoning We combine and deduplicate these three sets of structured knowledge to form the reasoning evidence $\mathcal{G}_Q$ for reasoning.

$$\mathcal{G}_Q = \mathcal{G}_Q^r \oplus \mathcal{G}_Q^t \oplus \mathcal{G}_Q^e. \tag{10}$$

Finally, question Q and $\mathcal{G}_Q$ are fed into the LLMs to reason the answer A for Q. Formally, the output can be represented by:

$$A = LLM(\mathcal{G}_Q || Q). \tag{11}$$

4 Experiments

4.1 Experiment Setup

Datasets We evaluate our model on four complex question answering datasets that require advanced reasoning and domain knowledge:

- **WebQSP** [31]: 4,737 natural language questions based on Freebase [3].
- **CWQ** [24]: 34,689 complex questions in natural language based on Freebase.
- **GenMedGPT-5k** [12]: A 5K dataset of dialogues between patients and doctors, using EMCKG [28] as the supporting KG.
- **CMCQA** [29]: A large Chinese medical QA dataset, with CMCKG [28] as the supporting KG.

We use the full test sets of WebQSP and CWQ, along with a Freebase subgraph. For GenMedGPT-5k and CMCQA, we sample 714 and 495 dialogues, respectively, following MindMap's work [28]. Dataset statistics are shown in Table 1.

Table 1. Dataset Information

Datasets	Question	KG	Node	Triple	Relation
WebQSP	1639	Freebase	841614	2351824	5419
CWQ	3531	Freebase	722268	2335214	5097
GenMedGPT-5k	714	EMCKG	1123	5802	6
CMCQA	495	CMCKG	62282	528466	12

Baselines For WebQSP and CWQ, We compare GoT-R with KBQA baseline methods from three categories: 1) Embedding based methods: KV-Mem [18], NSM [5] and TransferNet [21]. 2) LLMs: Alpaca-7B [22], LLaMA2-Chat-7B [25], ChatGPT [19] and ChatGPT+CoT [27]. 3) LLMs+KGs methods: KD-CoT [26], RoG [15], StructGPT [6], ToG [23], and ToG2.0 [17]. For GenMedGPT-5k and CMCQA, we use MindMap as the KG-augmented reasoning baseline. And follow its work, we use ChatGPT and GPT-4 [19] for implicit knowledge inference and include three retrieval-augmented baselines: BM25 retriever, Text Embedding retriever, and KG retriever. ChatGPT is used as the backbone.

Table 2. Performance comparison on WebQSP and CWQ.

Type	Methods	WebQSP		CWQ	
		EM(%)	F1(%)	EM(%)	F1(%)
Embedding	KV-Mem	46.7	34.5	18.4	15.7
	NSM	68.7	62.8	47.6	42.4
	TransferNet	71.4	-	48.6	-
LLM	Alpaca-7B	51.8	-	27.4	-
	LLaMA2-Chat-7B	64.4	-	34.6	-
	ChatGPT	66.8	-	39.9	-
	ChatGPT+CoT	75.6	-	48.9	-
LLM+KG	KD-CoT+ChatGPT	68.6	52.5	55.7	-
	StructGPT+ChatGPT	72.6	-	-	-
	ToG+ChatGPT	76.2	-	57.1	-
	ToG-R+ChatGPT	75.8	-	58.9	-
	ToG+GPT-4	82.6	-	67.6	-
	ToG-R+GPT-4	81.9	-	69.5	-
	ToG2.0+ChatGPT	81.1	-	-	-
	RoG+LLaMA2-Chat-7B	85.7	70.8	62.6	56.2
	GoT-R+ChatGPT	82.4	71.7	53.8	51.7
	GoT-R+GPT-4	**86.2**	**74.6**	**70.6**	**64.9**

Evaluation Metrics For WebQSP and CWQ, we use exact match (EM) to evaluate the top-ranked answer and F1 score to assess answer coverage. For GenMedGPT-5k and CMCQA, we use BERTScore to measure semantic similarity between the generated answer and the ground truth. Additionally, we use GPT-4 to evaluates the accuracy of three answer components: Disease, Medication, and Test and ranks the answer [28]. A lower rank signifies better accuracy.

Implementation Details In our experiments, we use ChatGPT (gpt-3.5-turbo) and GPT-4 (gpt-4-turbo) as the backbone LLMs. During training, we set $k_1 = k_2 = 3$ for WebQSP and CWQ, and $k_1 = 10, k_2 = 5$ for GenMedGPT-5k and CMCQA to generate the training dataset and a pretrained DistilBert model as the encoder. K-means clustering was chosen for its efficiency, with the number of clusters K matching the number of atomic relations. We use AdamW as the optimizer, with the learning rate of 2×10^{-6}. During inference, we set $k_q = 10$ for WebQSP and CWQ, $k_q = 3$ for GenMedGPT-5k and CMCQA, with path length $l = 3$, retrieving $K_1 = K_2 = K_3 = 100$ reasoning evidence.

4.2 Overall Experimental Results

As shown in Table 2 and Table 3, GoT-R outperforms non-LLM+KG methods on WebQSP and CWQ, demonstrating the effectiveness of combining LLMs with knowledge graphs. Integrated with ChatGPT, GoT-R surpasses the prompting-based SOTA method ToG on WebQSP, showing better reasoning capability.

Table 3. Performance comparison on GenMedGPT-5k and CMCQA.

Datasets	Methods	BERT Score			GPT-4 Ranking (Average)
		Precision	Recall	F1	
GenMed GPT-5k	BM25 Retriever	0.7693	0.7981	0.7831	3.7213
	Embedding Retriever	0.7690	0.8038	0.7858	4.3908
	KG Retriever	0.7718	0.8030	0.7868	4.0280
	ChatGPT	0.7612	0.8004	0.7800	4.9678
	GPT-4	0.7690	0.7893	0.7787	5.6120
	MindMap+ChatGPT	**0.7936**	0.7978	0.7954	2.6443
	GoT-R+ChatGPT	0.7815	**0.8244**	**0.8020**	**2.6373**
CMCQA	BM25 Retriever	0.9364	0.9348	0.9355	5.5293
	Embedding Retriever	0.9357	0.9335	0.9345	5.0162
	KG Retriever	0.9317	0.9348	0.9332	2.8101
	ChatGPT	0.9383	**0.9360**	0.9371	4.0303
	GPT-4	0.9352	0.9358	0.9355	3.5677
	MindMap+ChatGPT	0.9386	0.9315	0.9349	4.2586
	GoT-R+ChatGPT	**0.9407**	0.9339	**0.9372**	**2.7879**

However, its performance on CWQ improves less for its's weaker GoT construction ability for multi-hop questions. With GPT-4, GoT-R significantly improves, outperforming SOTA methods like fine-tuned-based method RoG and prompting-based method ToG. Fig. 3 highlights GoT-R+GPT-4's superior performance for multi-hop questions due to GPT-4's reasoning and in-context learning ability. In the medical domain, GoT-R excels with top F1 scores and GPT-4 ranking, demonstrating its effective reasoning capabilities for different domains. Overall, GoT-R's performance, especially with GPT-4, proves its effectiveness in enhancing LLMs reasoning for complex question answering and adapting LLMs to various domain.

4.3 Ablation Study

We conduct the following studies on subsets of WebQSP and CWQ, each with 200 randomly sampled questions. We compare four variants: 1) w/o ARS (atomic relations selection), 2) w/o HRR (hybrid retrieval and reasoning), 3) w/o RUR (relation-unaware retrieval), and 4) w/o RAR (relation-aware retrieval). As shown in Table 4, each component significantly impacts performance. Removing ARS leads to degradation, emphasizing its role in constructing graph-of-thought. Removing HRR causes greater declines, especially in F1 scores, highlighting the significance to reasoning with external knowledge. Using RAR instead of RUR results in a slight EM drop (0.7% on WebQSP, 5.8% on CWQ). Removing RAR while keeping RUR causes a greater decline (EM drops by 13.4% and 11.7%), underscoring RAR's greater importance. It demonstrate that the relation paths in GoT is important. In conclusion, all components are crucial, with HRR hav-

Table 4. Ablation results.

Models	WebQSP		CWQ	
	EM(%)	F1(%)	EM(%)	F1(%)
GoT-R	**74.5**	**61.0**	**51.5**	**42.8**
w/o ARS	71.5	58.0	48.5	40.7
w/o HRR	46.0	28.0	43.0	18.2
w/o RUR	74.0	59.1	48.5	40.2
w/o RAR	64.5	48.6	45.5	41.7

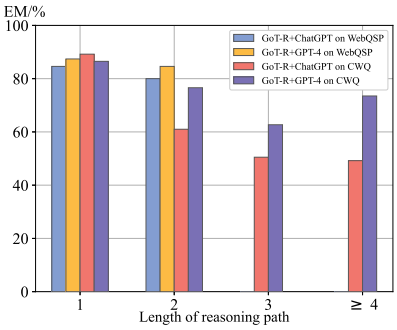

Fig. 3. Multi-hop QA analysis.

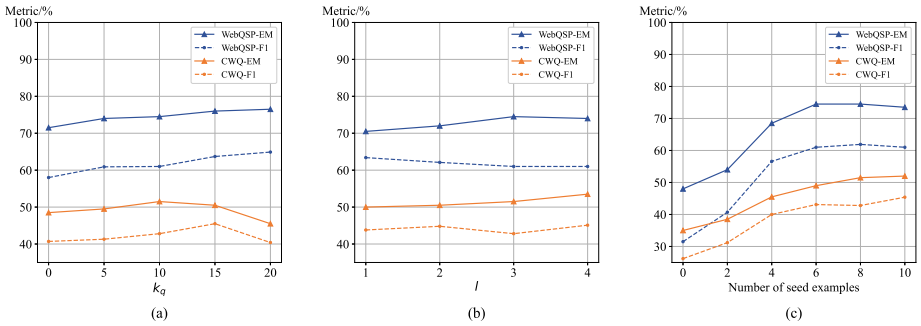

Fig. 4. Hyper-parameter analysis.

ing the greatest impact, RAR more important than RUR, and ARS essential for effective HRR.

4.4 Hyper-Parameter Analysis

We first evaluated GoT-R by varying two hyper-parameters: k_q (size of selected atomic relations) and l (path length of the path-based evidence graph). As shown in Fig. 4 (a) and (b), performance on WebQSP improved as k_q increased, while CWQ declined beyond $k_q = 10$ due to excess relations with interference, so we set $k_q = 10$. For l, WebQSP performed best at $l = 3$ for it only has 2-hop questions. While CWQ benefited from $l = 4$ for it has 4-hop questions, $l = 4$ increases computational costs, so we set $l = 3$ for balancing accuracy and computational efficiency.

Furthermore, in order to evaluate whether LLMs can construct graph-of-thought through few-shot in-context learning, we conducted a sensitivity analysis shown in Fig. 4 (c). As the number of examples increased, overall performance of GoT-R improved, with WebQSP peaking at 6-shot and CWQ slowing after 8-shot. Thus, few-shot sizes were set to 6 and 8, respectively. Variability was

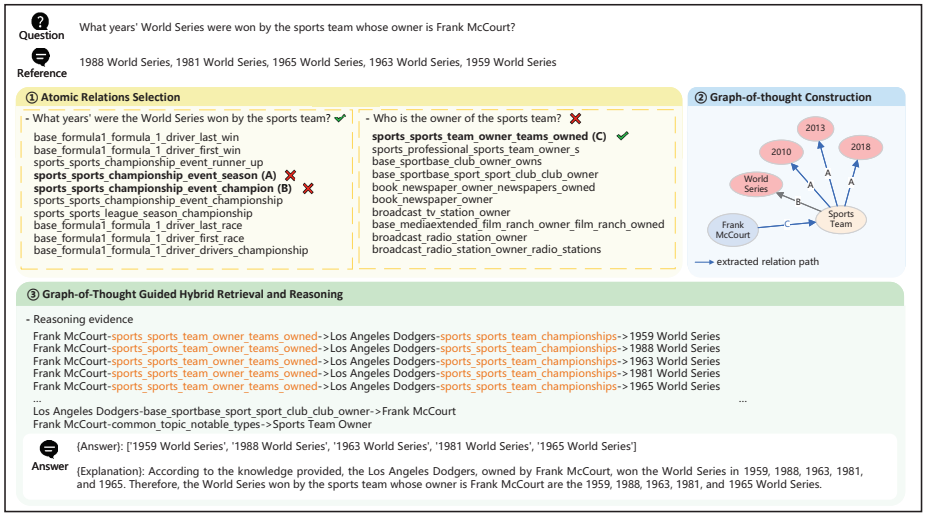

Fig. 5. Case study on the reasoning process on CWQ.

higher in WebQSP due to its simpler questions. Therefore, few-shot in-context learning is effective for graph-of-thought construction.

4.5 Case Study

We present a question answering case on CWQ in Fig. 5. This case demonstrates that even when the atomic question is incorrect, GoT-R can still construct a reasonable graph-of-thought, illustrating that enabling the LLM to think holistically can mitigate the impact of incorrect question decomposition. Moreover, although the atomic relation selection module does not always identify accurate relations, it can still guide GoT-R to retrieve the correct ones, highlighting its ability to select relations that are semantically highly relevant to the question. This example illustrates how our method leverages atomic relation selection to construct a graph-of-thought, guiding the reasoning process to achieve more accurate and reliable reasoning.

5 Conclusion

In this work, we introduce GoT-R, a once reasoning framework for answering complex question. GoT-R flexibly construct a graph-of-thought with atomic relations as a practicable reasoning plan to guide hybrid retrieval and reasoning, fully leveraging its reasoning potential. Experimental results on four complex QA datasets have demonstrated the effectiveness and broad applicability of our approach. Moreover, the success of GoT-R highlights the potential of graph-guided reasoning in overcoming LLM limitations.

Acknowledgments. This research is supported in part by the"14th Five-Year Plan" Civil Aerospace Pre-Research Project of China under Grant No. D020101, the Natural Science Foundation of China No. 62302213, Innovation Funding of Key Laboratory of Intelligent Decision and Digital Operations No. NJ2023027, Ministry of Industrial and Information Technology Project of Hebei Key Laboratory of Software Engineering, No. 22567637H, the Natural Science Foundation of Jiangsu Province under Grant No. BK20210280.

References

1. Baek, J., Aji, A. F., Saffari, A.: Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In: NLRSE, pp. 78–106 (2023)
2. Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., Hoefler, T.: Graph of Thoughts: Solving Elaborate Problems with Large Language Models. In: AAAI, Vol. 38, No. 16, pp. 17682-17690 (2024)
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
4. Guan, X., Liu, Y., Lin, H., Lu, Y., He, B., Han, X., Sun, L.: Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In: AAAI, Vol. 38, No. 16, pp. 18126-18134 (2024)
5. He, G., Lan, Y., Jiang, J., Zhao, W. X., Wen, J. R.: Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In: WSDM, pp. 553-561 (2021)
6. Jiang, J., Zhou, K., Dong, Z., Ye, K., Zhao, W. X., Wen, J. R.: Structgpt: A general framework for large language model to reason over structured data. In: EMNLP pp. 9237–9251 (2023)
7. Kim, J., Jin, S., Park, S., Park, S., Han, K.: Label-aware Hard Negative Sampling Strategies with Momentum Contrastive Learning for Implicit Hate Speech Detection. In: ACL, pp. 6311–6321 (2024)
8. Kou, Y., Shen, D., Snell, Q., Li, D., Nie, T., Yu, G., Ma, S.: Efficient team formation in social networks based on constrained pattern graph. In: ICDE, pp. 889-900 (2020)
9. Lan, Y., He, G., Jiang, J., Jiang, J., Zhao, W. X., Wen, J. R.: Complex knowledge base question answering: A survey. TKDE **35**(11), 11196-11215 (2022)
10. Li, J., Zhou, P., Xiong, C., Hoi, S. C.: Prototypical contrastive learning of unsupervised representations. In: ICLR (2021)
11. Li, S., Gao, Y., Jiang, H., Yin, Q., Li, Z., Yan, X., Zhang, C., Yin, B.: Graph Reasoning for Question Answering with Triplet Retrieval. In: ACL, pp. 3366–3375 (2023)
12. Li, Y., Li, Z., Zhang, K., Dan, R., Jiang, S., Zhang, Y.: Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. Cureus **15**(6) (2023)
13. Li, Z., Wang, C., Wang, X., Chen, Z., Li, J.: HJE: joint convolutional representation learning for knowledge hypergraph completion. TKDE **36**(8), 3879-3892 (2024)

14. Liu, J., Zhou, T., Chen, Y., Liu, K., Zhao, J.: Enhancing Large Language Models with Pseudo-and Multisource-Knowledge Graphs for Open-ended Question Answering. arXiv preprint arXiv:2402.09911. (2024)
15. Luo, L., Li, Y. F., Haffari, G., Pan, S.: Reasoning on graphs: Faithful and interpretable large language model reasoning. In: ICLR (2024)
16. Liu, Y., Xuan, H., Li, B., Wang, M., Chen, T., Yin, H.: Self-supervised dynamic hypergraph recommendation based on hyper-relational knowledge graph. In: CIKM, pp. 1617–1626 (2023)
17. Ma, S., Xu, C., Jiang, X., Li, M., Qu, H., Guo, J.: Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. arXiv preprint arXiv:2407.10805 (2024)
18. Miller, A., Fisch, A., Dodge, J., Karimi, A. H., Bordes, A., Weston, J.: Key-Value Memory Networks for Directly Reading Documents. In: EMNLP, pp. 1400–1409 (2016)
19. OpenAI, https://openai.com/
20. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying large language models and knowledge graphs: A roadmap. TKDE **36**(7), 3580-3599 (2024)
21. Shi, J., Cao, S., Hou, L., Li, J., Zhang, H.: TransferNet: An Effective and Transparent Framework for Multi-hop Question Answering over Relation Graph. In: EMNLP, pp. 4149–4158 (2021)
22. Stanford Alpaca, https://crfm.stanford.edu/2023/03/13/alpaca.html
23. Sun, J., Xu, C., Tang, L., Wang, S., Lin, C., Gong, Y., Ni, L. M., Shum H. Y., Guo, J.: Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. In: ICLR (2024)
24. Talmor, A., Berant, J.: The Web as a Knowledge-Base for Answering Complex Questions. In: NAACL, Vol. 1, pp. 641–651 (2018)
25. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
26. Wang, K., Duan, F., Wang, S., Li, P., Xian, Y., Yin, C., Rong, W., Xiong, Z.: Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. arXiv preprint arXiv:2308.13259 (2023)
27. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: NeurIPS, Vol. 35, pp. 24824-24837 (2022)
28. Wen, Y., Wang, Z., Sun, J.: MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models. In: ACL, Vol. 1, pp. 10370–10388 (2024)
29. Xia, F., Li, B., Weng, Y., He, S., Liu, S., Sun, B., Li, S., Zhao, J.: MedConQA: Medical Conversational Question Answering System based on Knowledge Graphs. In: EMNLP, pp. 148–158 (2022)
30. Yao, Y., Li, Z., Zhao, H.: GoT: Effective Graph-of-Thought Reasoning in Language Models. In: NAACL, pp. 2901-2921 (2024)
31. Yih, W., Richardson, M., Meek, C., Chang, M., Suh, J.: The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In: ACL, Vol. 2, pp. 201–206 (2016)
32. Zhang, J., Zeng, W., Tang, J., Zhao, X.: Hyperedge Graph Contrastive Learning. TKDE **36**(12), 8502-8514 (2024)
33. Zheng, L., Fei, H., Li, F., Li, B., Liao, L., Ji, D., Teng, C.: Reverse multi-choice dialogue commonsense inference with graph-of-thought. In: AAAI, Vol. 38, No. 17, pp. 19688-19696 (2024)