Fast Think-on-Graph: Wider, Deeper and Faster Reasoning of Large Language Model on Knowledge Graph

Xujian Liang[1,2], **Zhaoquan Gu**[2,3*]
[1]School of Cyberspace Security, Beijing University Of Posts And Telecommunications, China
[2]Department of New Networks, Peng Cheng Laboratory, Shenzhen, China
[3]School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China
xjliang@bupt.edu.cn, guzhaoquan@hit.edu.cn

Abstract

Graph Retrieval Augmented Generation (GRAG) is a novel paradigm that takes the naive RAG system a step further by integrating graph information, such as knowledge graph (KGs), into large-scale language models (LLMs) to mitigate hallucination. However, existing GRAG still encounter limitations: 1) simple paradigms usually fail with the complex problems due to the narrow and shallow correlations capture from KGs 2) methods of strong coupling with KGs tend to be high computation cost and time consuming if the graph is dense. In this paper, we propose the Fast Think-on-Graph (FastToG), an innovative paradigm for enabling LLMs to think "community by community" within KGs. To do this, FastToG employs community detection for deeper correlation capture and two stages community pruning - coarse and fine pruning for faster retrieval. Furthermore, we also develop two Community-to-Text methods to convert the graph structure of communities into textual form for better understanding by LLMs. Experimental results demonstrate the effectiveness of FastToG, showcasing higher accuracy, faster reasoning, and better explainability compared to the previous works.

Code — https://github.com/dosonleung/FastToG

Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al. 2020) is a cutting-edge technique for large-scale language models (LLMs) to generate accurate, reasonable, and explainable responses. The early RAG, known as Naive RAG (Gao et al. 2023), mostly works by indexing the documents into manageable chunks, retrieving relevant context based on vector similarity (Karpukhin et al. 2020) to a user query, integrating the context into prompts, and generating the response via LLMs. As the paradigm is simple and highly efficient for the basic tasks, Naive RAG is widely adopted in various applications e.g. Q&A (Chan et al. 2024), Recommendation System (Deldjoo et al. 2024), Dialogue System (Ni et al. 2023), etc. However, there remain such as low precision as the ambiguity or bias (Thurnbauer et al. 2023) exist in the embedding model, low recall when dealing with complex queries, and lack of explainability as the queries and documents are embedded in high-dimensional vector spaces.

[*]Corresponding Author

The Graph-based RAG (GRAG) is widely considered an advanced RAG by incorporating KGs as an external source for LLMs. Employing various graph algorithms within a graph database, GRAG empowers LLMs for complex operations (Liang et al. 2024) such as BFS and DFS querying, path exploration, and even community detection, etc, providing LLMs a wider and deeper understanding of the relations within the data, making it possible to execute sophisticated reasoning and the generate more precise responses. In this paper, GRAG is categorized into **n-w GRAG** and **n-d GRAG** (Fig. 1) based on the breadth and depth of retrieval. Particularly, there are still quite a few 1-w 1-d Graph RAG researches (Andrus et al. 2022; Baek, Aji, and Saffari 2023; Li et al. 2023). Similar to the "Retrieve one Read One" paradigm of Naive RAG, most of the 1-w 1-d research focuses on single entity or one-hop relationship queries, thereby inheriting the shortcomings of Naive RAG. To overcome these, researches (Jiang et al. 2023; Modarressi et al. 2023; Edge et al. 2024; Wang et al. 2024), categorized as n-w GRAG, aim to expand the context window of retrieval. Noteworthy works within the n-w GRAG category focus on treating the densely interconnected groups of nodes as the basic units for retrieval and reasoning. These groups are partitioned from the graph by community detection, allowing them to furnish LLMs with more contextual information compared to single node or random clustering methods. For example, given the query concerning the "climate of the area where Pennsylvania Convention Center is located", triples such as (Pennsylvania Convention Center, located in, Market East Section),...,(Pennsylvania, climate, humid subtropical) would be sufficient for answering as the entity *Pennsylvania Convention Center* of the query is linked to the relevant entity *humid subtropical* over a short distance (n-hop). Despite its simplicity, the retrieval may falter if the distance is too long between the entities (Fig 1 a,b). n-d GRAG (Sun et al. 2023; Wang et al. 2022a) are ideal for this predicament by broadening the depth of the context window. Analogous to "Let's think step by step" (Kojima et al. 2022), n-d GRAG takes entities and relations as intermediate steps in the chain of thought (CoT) to guide the generation of LLMs. As they are tightly coupled with KGs, these paradigms exhibit higher recall in retrieval. However, since LLMs are required to "think" at each step, longer chains result in heightened computational costs (Fig. 1, c). On the other hand, if

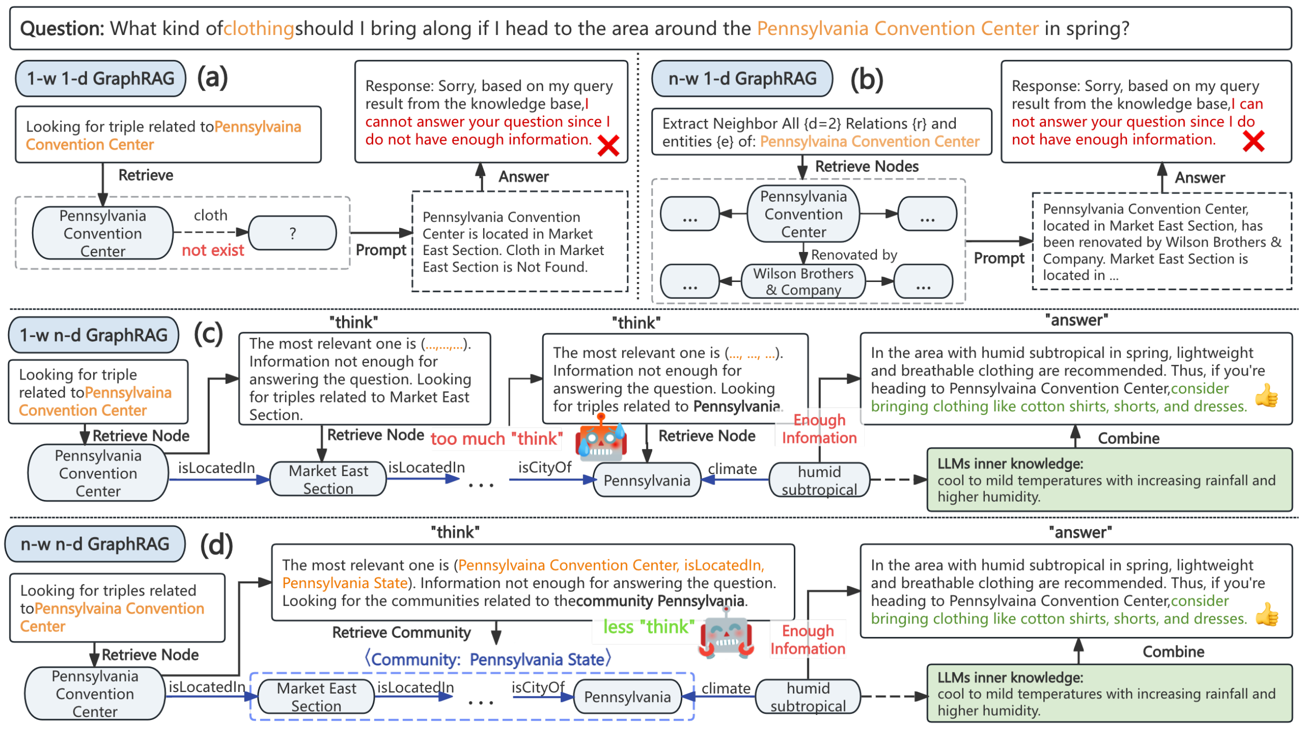
Question: What kind of clothing should I bring along if I head to the area around the Pennsylvania Convention Center in spring?
1-w 1-d GraphRAG (a)
Looking for triple related to Pennsylvaina Convention Center
Retrieve
Pennsylvania Convention Center
cloth
?
not exist
Prompt
Pennsylvania Convention Center is located in Market East Section. Cloth in Market East Section is Not Found.
Answer
Response: Sorry, based on my query result from the knowledge base, I cannot answer your question since I do not have enough information.
n-w 1-d GraphRAG (b)
Extract Neighbor All {d=2} Relations {r} and entities {e} of: Pennsylvaina Convention Center
Retrieve Nodes
Renovated by
Wilson Brothers & Company
Prompt
Pennsylvania Convention Center, located in Market East Section, has been renovated by Wilson Brothers & Company. Market East Section is located in ...
Answer
Response: Sorry, based on my query result from the knowledge base, I can not answer your question since I do not have enough information.
1-w n-d GraphRAG (c)
Looking for triple related to Pennsylvaina Convention Center
Retrieve Node
"think"
The most relevant one is (..., ..., ...). Information not enough for answering the question. Looking for triples related to Market East Section.
Retrieve Node
too much "think"
"think"
The most relevant one is (..., ..., ...). Information not enough for answering the question. Looking for triples related to Pennsylvania.
Retrieve Node
Enough Infomation
"answer"
In the area with humid subtropical in spring, lightweight and breathable clothing are recommended. Thus, if you're heading to Pennsylvaina Convention Center, consider bringing clothing like cotton shirts, shorts, and dresses.
Combine
LLMs inner knowledge: cool to mild temperatures with increasing rainfall and higher humidity.
isLocatedIn
Market East Section
isLocatedIn
isCityOf
Pennsylvania
climate
humid subtropical
n-w n-d GraphRAG (d)
Looking for triples related to Pennsylvaina Convention Center
Retrieve Node
"think"
The most relevant one is (Pennsylvaina Convention Center, isLocatedIn, Pennsylvaina State). Information not enough for answering the question. Looking for the communities related to the community Pennsylvania.
Retrieve Community
less "think"
Enough Infomation
⟨Community: Pennsylvania State⟩
"answer"
In the area with humid subtropical in spring, lightweight and breathable clothing are recommended. Thus, if you're heading to Pennsylvaina Convention Center, consider bringing clothing like cotton shirts, shorts, and dresses.
Combine
LLMs inner knowledge: cool to mild temperatures with increasing rainfall and higher humidity.

Figure 1: Comparison of 1-w 1-d Graph RAG (a), n-w Graph RAG (b), n-d Graph RAG (c) and n-w n-d GraphRAG (d)

the graph is dense, the multitude of paths between the source and target entities can become exceedingly vast, leading to a decreased recall of entity retrieval.

Building on the consideration of n-w or n-d methods, we propose a **n-d n-w Graph RAG** paradigm: Fast Think-on-Graph (FastToG). The key idea of FastToG is to guide the LLMs to **think "community by community"**. As illustrated in Fig. 1, though the path between *Pennsylvania Convention Center* and *humid subtropical* may be lengthy, it notably shortens when nodes are grouped. Accordingly, FastToG regards communities as the steps in the chain of thought, enabling LLMs to "think" wider, deeper, and faster. Concretely, FastToG leverages community detection on the graph to build the reasoning chains. Considering the time complexity of community detection, we introduce Local Community Search (LCS), aiming to literately detect the communities in a local scope. Given potential graph density concerns, LCS incorporates two stages of community pruning: modularity-based coarse pruning and LLMs-based fine pruning, aiming to enhance the efficiency and accuracy of retrieval. Furthermore, as the language models are trained on textual data, graph structures are incompatible with the format of input. In light of this, We explore two methods to convert community into text: Triple2Text and Graph2Text, which aim to provide better inputs for LLMs.

We conducted experiments on real-world datasets, and FastToG exhibited the following advantages:

1. Higher Accuracy: FastToG demonstrate its significant enhancement on the accuracy of LLMs generated content compared with the previous methods.
2. Faster Reasoning: Our experiments also show that community-based reasoning can notably shorten the reasoning chains, reducing the number of calls to the LLMs.
3. Better Explainability: The case study indicates that FastToG not only simplifies the retrieval for LLMs but also enhances the explainability for users.

Related Work

Algorithms of Community Detection

Community detection algorithms are used to grouping or partitioning nodes, as well as their tendency to strengthen or separate apart. These algorithms can be categorized into agglomerative and divisive types.

Agglomerative: These algorithms iteratively group smaller communities until a stop condition is met. One prominent method is hierarchical clustering, which progressively groups nodes or small communities into larger ones by similarity. This down-top grouping procedure is intuitive but computationally expensive, making it impractical for large-scale networks. Since the determination to stop the clustering is important, Louvain (Blondel et al. 2008) and Leiden (Traag, Waltman, and Van Eck 2019) algorithms introduce the modularity benefit function as a measure of community quality. Based on modularity, the optimization procedure can be explicitly stopped as the global modularity can no longer be improved given perturbations.

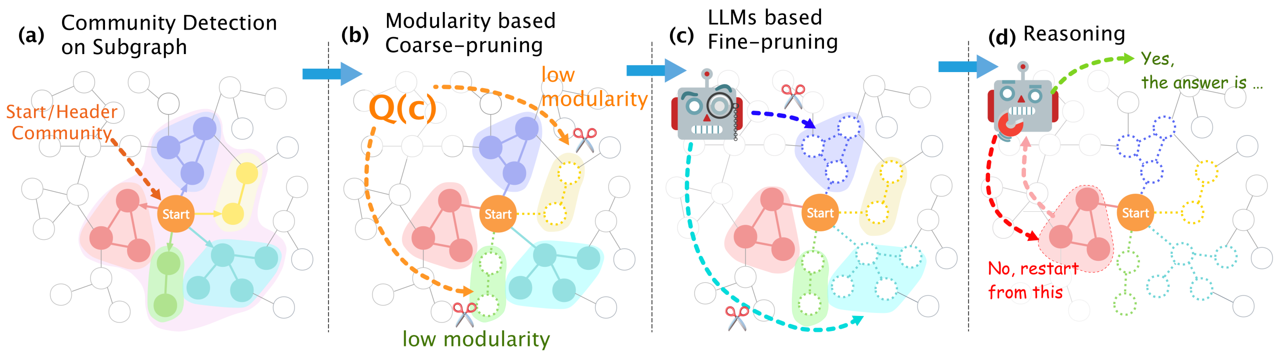

Figure 2: A general schema of the FastToG paradigm.

Divisive: In contrast, Divisive methods follow the up-bottom process of iteratively partitioning the original graph into small substructures. For example, the Girvan-Newman (GN) algorithm (Girvan and Newman 2002) removes edges with high betweenness centrality to foster more connected components as communities. Spectral clustering leverages the spectrum(eigenvalues) to embed nodes to k-dimensional vector space and group them based on the clustering method e.g. k-means.

Graph-based Retrieval Augmented Generation

Graph RAG empowers LLMs to capture and utilize the structure and semantics of KGs, enhancing the accuracy of response and explainability of reasoning process. According to the stage at which KGs participate in, GRAG are categorized into Pretrain/Fine-tuning based and Prompt-guided based methods.

Pretrain/Fine-tuning based: These methods enable language models to learn KGs by Input augmentation or model enhancement. (Xie et al. 2022) linearize graph structures to fine-tune language models, whereas K-BERT expands input structures with KGs information. (Zhang et al. 2019) embed the entities by incorporating knowledge encodes. (Wang et al. 2020) infuses knowledge into pretrained models via transformer adapters, and (Yamada et al. 2020) enhances transformers with an entity-aware self-attention mechanism. While these methods effectively integrate KGs into language models, they require additional supervised data, leading to increasing costs as the model scaling increases.

Prompt-guided based As the training/fine-tuning process is often expensive and time-consuming, recent RAG systems focus on prompt-guided methods. Based on the width and depth of retrieval, these works can be categorized into 1-w 1-d, n-w, and n-d methods. Examples of 1-w 1-d methods include Cok (Li et al. 2023), KAPING (Baek, Aji, and Saffari 2023) etc. Cok enhances the flexibility of queries with SPARQL. KAPING vectorizes triples for efficient retrieval. Representative works of n-w includes KGP (Wang et al. 2024), StructGPT (Jiang et al. 2023), RET-LLMs (Modarressi et al. 2023) and GraphRAG (Edge et al. 2024) etc. While the first three all randomly or fully capture the adjacent nodes for retrieval purposes, GraphRAG stands out by utilizing community detection to optimize retrieval for text summarization. KP-PLM (Wang et al. 2022a) and ToG (Sun et al. 2023) are the typical works of the n-d paradigm. KP-PLM firstly designs the $d > 1$ entity-relation chains for information retrieval. ToG, particularly relevant to our research, introduces enhancements such as pruning and beam search within chains via LLMs.

The Method

Overview

We begin by introducing the basic framework of Fast think on graph (FastToG). Inspired by the work (Kojima et al. 2022) "Let's think step by step", the basic idea of FastToG is enabling large models to "think" community by community on the KGs. Consequently, the core component of the FastToG (see our repository for the pseudocode) are the W reasoning chains P =[$p_1, p_2, ..., p_W$], where each chain $p_i \subseteq P$ is a linked list of communities i.e., p_i = [c_0^i, c_1^i, ..., c_{n-1}^i].

The FastToG framework comprises two main phases: the initial phase and the reasoning phase. The objective of the initial phase is to determine the start communities c_0 and the header community (Fig. 2a) c_0^i for each reasoning chain p_i. FastToG prompts LLMs to extract the subject entities of to query x as a single-node community c_0. After that, FastToG employs Local Community Search (LCS), which involves two key components: Community Detection on Subgraph (Fig. 2b) and Community Pruning (Fig. 2c), to identify neighbor communities with highest potential for solving the query x, serving as the head community c_0^i for each reasoning chain p_i.

Once the head communities are determined, the algorithm enters into the reasoning phase. For each p_i, FastToG continues to leverage LCS for local community detection and pruning, with the selected community being added to p_i as the newest element, termed **pruning**. After the update of all reasoning chains, Community2Text methods like Graph2Text (G2T) or Triple2Text (T2T) are utilized to convert all communities within chains into textual form as the input content of LLMs, which is called **reasoning** (Fig. 2d). If the gathered reasoning chains are deemed adequate by the LLMs for generating the answer, the algorithm will be terminated and return the answer. To mitigate time consumption, if no answer is returned within D_{max} iterations of reasoning, the algorithm will be degraded into the methods of answering the query by inner knowledge of LLMs e.g. IO/CoT/CoT-SC. We will illustrate all the details in the following sections.

Local Community Search

In this subsection, we focus on the LCS, which consist of there main part: community detection on the subgraph, pruning methods of coarse-pruning and fine-pruning.

Community Detection on Subgraph Due to the vast number of nodes and relations in the KGs, as well as the dynamic of queries, it is impossible to partition the KGs into communities entirely and frequently. To solve this, we propose community detection based on the local subgraphs for each pruning. Given KGs Ω, start community c_0 or header community c_0^i, FastToG firstly retrieves the subgraph g ($g \subset \Omega$) within n hops from c_0 or c_0^i. Considering the rapid growth of the number of neighbors in the dense graph and the degradation of semantics as the nodes move further apart, the algorithm randomly samples neighbor nodes at different hops with exponential decay ρ. The probability of selecting each node x at n-hop from c_0 or c_1^i is given by:

$$Pr(x = 1) = \rho^{n-1}$$

Once the local subgraph g is retrieved, the next step is to partition it into communities. Community detection algorithms are utilized to reveal the closely connected groups of nodes (referred to as communities) of the graph. This process enables the graph to be partitioned into subgroups with tight internal connectivity and sparse external connections, aiding LLMs in uncovering implicit patterns within the graph structure. In this study, we consider Louvain (Blondel et al. 2008), Girvan–Newman algorithm (Girvan and Newman 2002), Agglomerative Hierarchical Clustering, and Spectral Clustering as representative algorithms. To prevent the retrieval of repeated communities, the algorithm ensures that new communities discovered do not exist in the historical community set during each community detection iteration, and adds them to the historical community set after each pruning step.

The oversize communities may contain more redundant nodes, leading to the increased noise, prolonged community detection time, and reduced explainability. Thus, we introduce a constraint on the maximum size of community within the community detection. To do this, all detection algorithms will be backtracked when the stop condition is met. At each iteration from the end to the beginning of backtracking, the algorithm verifies whether the size of each community meets the condition size$(c_i) <= M$ (hyperparameter). Finally, the algorithm returns the partition status that first satisfies the size constraint.

Modularity-based Coarse-Pruning When dealing with too many candidate communities, existing methods relying on Large Language Models (LLMs) for community selection encounter difficulties, resulting in either high time consumption or selection bias (Zheng et al. 2023). To tackle this, current methods frequently incorporate random sampling to diminish the number of candidate communities or nodes. However, these methods inevitably ignore the network structure of the graph, resulting in the loss of structural information from the candidate communities.

Based on the above, we propose modularity-based coarse-pruning. The modularity (Blondel et al. 2008) of a partitioned subgraph g is:

$$Q = \frac{1}{2m}\sum_{ij}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) \tag{1}$$

Where m is the number of edges, A_{ij} is an element in the adjacency matrix A, k_i and k_j are the degrees of nodes i and j, respectively. The function $\delta(c_i, c_j) = 1$ indicates that nodes i and j belong to the same community, otherwise, it returns 0. In the weighted graph, m is the sum of graph weights, $\frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i, c_j)$ represent the average weight of the given community, while $\frac{k_i k_j}{2m}\delta(c_i, c_j)$ denotes the average weight of the given community under random conditions. The larger the difference between them, the higher the quality of community partitioning. For simplicity, we do not consider the weights or directions of edges in our work. The expression1 of modularity can be rewritten as:

$$Q = \frac{1}{2m}[\sum_{ij} A_{ij} - \frac{\sum_i k_i \sum_j k_j}{2m}]\delta(c_i, c_j) \tag{2}$$

$$= \frac{1}{2m}\sum_{c}[\sum \text{in} - \frac{(\sum \text{tot})^2}{2m}] \tag{3}$$

where $\sum$in is the number of edges in community c and $\sum$tot is the number of edges connected to c. Thus, the modularity of community c can be:

$$Q(c) = \sum \text{in} - \frac{(\sum \text{tot})^2}{2m} \tag{4}$$

Upon calculating the modularity of each candidate community $c \subseteq C$, the communities $C^{'}$ with low modularity will be pruned, while the remaining will serve as the refined set of candidate communities for the next stage.

$$C^{'} := \text{argtopk}_{c \subseteq C} Q(c) \tag{5}$$

LLMs-based Fine-Pruning After coarse pruning, a smaller set of candidate communities $C^{'}$ that are more compact in structure is identified. Subsequently, FastToG prompts the LLMs to make the final selection $C^{''}$:

$$C^{''} = \text{fine\_pruning}(x, C^{'}, \Pi, k) \tag{6}$$

where x is the query, Π is the instance of LLMs, $k = 1$ or W is for the single or multiple choice.

To simplify the pruning process, FastToG no longer considers scoring-based pruning (Sun et al. 2023). Instead, it prompts the LLMs to directly choose either the best community or top W communities. In the initial phase, LLMs are guided through multiple-choice prompts to retrieve W communities, which will act as the header communities c_0^1,...,c_0^W for the reasoning chains p_1,...,p_W, respectively. During the reasoning phase, single-choice prompts are employed for each reasoning chain to recall the best community c_j^i, which is then appended to its chain p_i. Not that the length of each chain p may not be the same if LLMs insist that none of the candidate communities are relevant to the query. In such cases, the exploration of such chains will be discontinued.

Reasoning

Once of all the reasoning chains $p_i \subset P$ are updated, LLMs will be prompted to integrate and generate the answers from all the chains. The returned results could be a clear answer if the chains are adequate for answering, or "Unknown" if not. In cases where "Unknown" is returned, the algorithm proceeds to the next iteration until either reaching the maximum depth $D = D_{max}$ or obtaining a definitive answer. If the maximum iterations are exhausted without a conclusive response, FastToG will be degraded to generate the answer by the inner knowledge of LLMs itself. Consistent with ToG (Sun et al. 2023), the entire process of FastToG paradigm involves 1 round of pruning and reasoning in the initial phase, and $2WD_{max}$ pruning and D_{max} reasoning in the second. Consequently, the worst condition needs $2WD_{max} + D_{max} + 2$ calls to the LLMs.

Community-to-Text

Knowledge graphs are organized and stored in the form of RDF triples. Thus, a community also consisted of triples e.g. c = [(Philadelphia, isCityOf, Pennsylvania), (Pennsylvania, climate, Humid Subtropical)]. To input this structure into LLMs, it needs to be converted into text format. To do this, we propose two methods: Triple-to-Text (T2T) and Graph-to-Text (G2T). For T2T, triples are directly converted into text by rule-based scripts e.g. T2T(c) = "Philadelphia isCityOf Pennsylvania, Pennsylvania climate Humid Subtropical". For G2T, triple will be converted into human language like G2T(c) = "Philadelphia, located in the state of Pennsylvania, features a Humid Subtropical climate."

T2T may result in redundancy. For instance, a T2T result of [(Allentown, isCityOf, Pennsylvania), (New Castle, isCityof, Pennsylvania), (Philadelphia, isCityOf, Pennsylvania)] is "Allentown isCityOf Pennsylvania, New Castle isCityof Pennsylvania, Philadelphia isCityOf Pennsylvania", which can be summarized as: "Allentown, New Castle, and Philadelphia are the cities of Pennsylvania". Therefore, G2T also undertakes the role of the text summary. To do this, we fine-tune the smaller language models (like T5-base, 220M) on the outputs from T2T for the conversion.

Since G2T leverages a base model with less parameters compared to existing LLMs (e.g., llama-3-8b has 36 times more parameters than T5-base), the impact of time efficiency is tiny. Apart from intra-community relationships, there also exist inter-community paths such as $c_1 - E_{1,2} - c_2$. Therefore, it is necessary to perform text transform on the $E_{1,2}$. This study excludes candidate communities with distance larger than 1 hop from the current community, meaning the paths between the current community and candidate communities do not contain any intermediate nodes.

Experiments

Experimental Setup

Dataset and Evaluation Metric We evaluated FastToG on 6 real-world datasets, which include datasets of multi-hop KBQA: CWQ (Talmor and Berant 2018), WebQSP (Yih et al. 2016), and QALD (Perevalov et al. 2022), slot filling: Zero-Shot RE (abbr. ZSRE) (Petroni et al. 2021) and TREx (Elsahar et al. 2018), and common-sense reasoning Creak (Onoe et al. 2021). To ensure comparability with prior works (Li et al. 2023; Sun et al. 2023; Edge et al. 2024), we employed exact match (hit@1) as the evaluation metric. Considering the computational cost and time consumption, we randomly sampled 1k examples from the datasets containing more than a thousand data. Furthermore, the initial entities provided by (Sun et al. 2023) were used as the starting community for the evaluations.

Language Model To reduce the computational cost, we employ two LLMs: gpt-4o-mini[1] and Llama-3-70b-instruct (Dubey et al. 2024) without quantization. To keep the pruning steady, the temperature is configured to 0.4 during pruning and 0.1 during reasoning. Considering the text descriptions of community structure are longer, the maximum length of output is increased to 1024.

Graph2Text Model The Graph2Text module is developed by fine-tuning T5-base (Raffel et al. 2020) using dataset WebNLG (Auer et al. 2007) and synthetic data generated by GPT-4. To build the synthetic data, we prompt GPT-4 to generate text descriptions of given communities, which are the byproduct of the experiments.

Knowledge Graph We utilize Wikidata (Vrandečić and Krötzsch 2014), which is a free and structured knowledge base, as source of KGs. The version of Wikidata is 20240101 and only the triples in English are considered. Following extraction, 84.6 million entities and 303.6 million relationships are stored in Neo4j[2].

Performance on Accuracy

Accuracy is one of the most important criteria for RAG systems. In this experiment, we evaluate FastToG and compared methods on the datasets and settings mentioned above.

Compared Methods We consider two categories of comparative methods: Inner-knowledge-based or KGs-retrieval-based methods. The former methods include: 1) IO prompt (Brown et al. 2020): prompting the model to directly generate the result. 2) CoT (Wei et al. 2022): guiding the model to "think" step by step before answering. 3) CoT-SC (Wang et al. 2022b): ensembling all the CoTs to obtain more consistent predictions. The KGs-retrieval-based methods include: 1) 1-d 1-w methods, which represent the methods like CoK

[1]https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence

[2]https://neo4j.com/

Method	CWQ	WebQSP	QALD	ZSRE	TREx	Creak
Inner-knowledge based Methods						
IO	31.2	49.6	38.6	26.4	46.4	90.2
CoT	35.1	60.8	51.8	35.6	52.0	94.6
CoT-SC	36.3	61.2	52.4	35.8	52.0	95.0
KGs-retrieval based Methods						
1-d 1-w	35.5	59.2	50.7	39.4	56.1	92.0
1-d n-w	42.3	64.4	54.8	46.1	58.8	92.8
n-d 1-w	42.9	63.6	54.9	54.0	64.2	95.4
Ours(t2t)	43.8	65.2	**56.1**	**54.4**	67.3	95.6
Ours(g2t)	**45.0**	**65.8**	55.9	54.2	**68.6**	**96.0**

Table 1: Accuracy (%) for different datasets by gpt-4o-mini.

Method	CWQ	WebQSP	QALD	ZSRE	TREx	Creak
Inner-knowledge based Methods						
IO	32.9	55.8	44.3	30.7	52.8	91.0
CoT	34.2	58.8	45.0	32.6	55.6	91.2
CoT-SC	34.8	60.6	46.8	34.9	57.0	91.8
KGs-retrieval based Methods						
1-d 1-w	35.0	57.6	44.4	34.4	56.0	91.3
1-d n-w	39.8	64.0	46.6	57.1	60.4	91.9
n-d 1-w	40.3	62.4	51.6	64.8	61.2	93.3
Ours(t2t)	42.1	65.9	**54.9**	67.7	63.5	92.2
Ours(g2t)	**46.2**	**66.4**	54.3	**67.9**	**64.7**	**94.5**

Table 2: Accuracy (%) for different datasets by llama-3-70b.

(Li et al. 2023), KAPING (Baek, Aji, and Saffari 2023), etc. 2) 1-d n-w methods for KGP (Wang et al. 2024) 3. n-d 1-w method for ToG (Sun et al. 2023). To keep consistency with previous works, we keep a $W = 3$ for the number of chains and $D_{max} = 5$ for the maximum iterations. For n-w methods like 1-d n-w, Ours(t2t), and Ours(g2t), the maximum size of community is all set at 4 and the algorithm for community detection are Louvain Method.

Compared Result Tab. 1 and Tab. 2 respectively display the accuracy achieved by gpt-4o-mini and llama3-70b-instruct across the datasets. Overall, FastToG, which includes t2t and g2t mode, outperforms all previous methods. In particular, Ours(g2t) surpasses n-d 1-w (ToG) by 4.4% in Tab. 1 and 5.9% in Tab. 2.

For two community-to-text conversions, g2t methods show higher accuracy on most datasets, aligning well with the idea for our proposed Graph2Text in the previous section. However, we note that the improvement of g2t is tiny (mostly $< 1\%$), and even slightly underperforms to t2t on QALD dataset. To find out the explanation for these counterintuitive results, we carried out extensive checks (see our repository for details) and found that the hallucination from the base model of Graph2Text is the main reason for g2t falling short of t2t.

Performance on Efficiency

Efficiency is another key criterion of our work. To verify it, we consider the number of calls to the LLMs as the evaluation criterion. As outlined in Section Reasoning, for each question, the number of calls to the LLMs is estimated as $2WD + D + 2$. Since all the settings of $n-d$ methods have

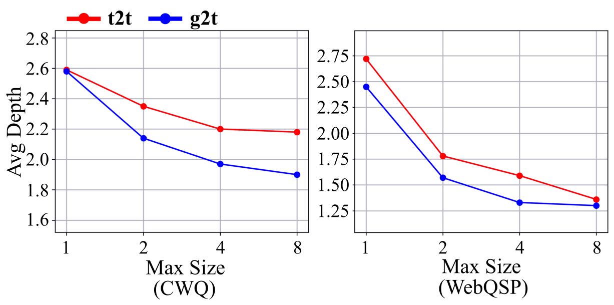

Figure 3: Average Depth versus Max size of community

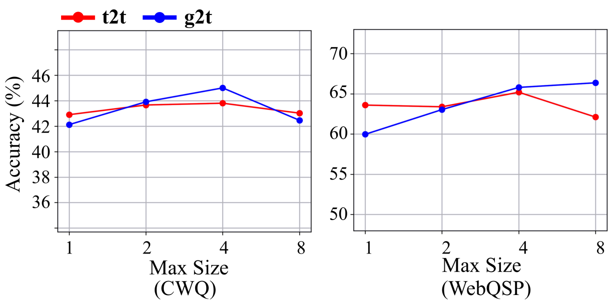

Figure 4: Accuracy versus Max size of community

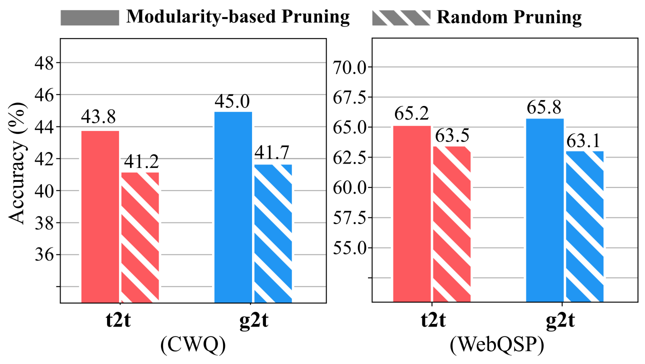

Figure 5: Accuracy (%) of two pruning methods

the same W, we only need to compare D to for the evaluation as efficiency $\epsilon \propto D$. Methods IO, CoT, CoT-SC, and 1-d are not considered because the value of D is fixed at 1.

Fig. 3 illustrates the relationship between the max size of the community and average depth on the CWQ and WebQSP datasets, where $MaxSize = 1$ refers to the previous n-d 1-w works like ToG. It is evident that FastToG ($MaxSize > 1$) outperforms the n-d 1-w methods. Even with the communities with $MaxSize = 2$, there can be a significant reduction in the $AvgDepth$ of the reasoning chains. For example, the $AvgDepth$ on CWQ is reduced by approximately 0.2 and 0.4 for t2t/g2t modes, respectively. On WebQSP, these reductions increase to about 1.0 for both modes. Moreover, as the $MaxSize$ increases, the decrease in depth becomes less substantial.

		Rand	Louvain	GN	Hier	Spectral
CWQ	t2t	40.4	43.8	44.0	44.5	44.0
	g2t	42.6	45.0	45.7	46.0	45.3
WebQSP	t2t	60.6	65.2	66.8	66.0	65.1
	g2t	62.5	65.8	66.9	66.1	66.7

Table 3: Accuracy (%) of Community Detection Algorithms

Ablation Study

Trade-off between Accuracy and Efficiency While larger communities can reduce the average depth of reasoning chains, such large communities may also bring more noise, potentially negatively impacting the reasoning effectiveness of LLMs. Fig. 4 illustrates the relationship between the $MaxSize$ and accuracy. Overall, most of the cases achieve higher accuracy when $MaxSize$ is set to 4. However, at $MaxSize = 8$, results show a decrease in accuracy. Therefore, setting a larger size of the community does not necessarily result in higher gain from accuracy.

Comparison on Pruning Methods To validate the effectiveness of our proposed Modularity-based Coarse Pruning, we compared it with Random Pruning, a method widely used by previous works. Fig. 5 depicts the accuracy comparison between Modularity-based coarse pruning and random pruning on the CWQ and WebQSP datasets for 2 modes, with all cases using $MaxSize = 4$. Overall, Modularity-based Coarse Pruning outperforms Random Pruning in all cases. Particularly, we observed that cases based on g2t mode are more sensitive in modularity-based pruning, indicating that communities of densely connected structure are preferable for the conversion between graphs and text.

Comparison on different Community Detection Community detection is a crucial step in FastToG. Tab. 3 compares the impact of different detection algorithms — including Louvain, Girvan-Newman (abbr. GN), hierarchical clustering (abbr. Hier), and Spectral Clustering (abbr. Spectral) on accuracy. Additionally, we consider random community detection (abbr. Rand), which randomly partitions nodes into different groups. Comparing the 4 non-random algorithms, the impact of different community detection on FastToG is very small ($< 1\%$ on average). On the other hand, when comparing algorithms between random and non-random, the latter outperforms the former ($> 3\%$ on average), demonstrating that community detection does help FastToG.

Case Study

For the query "Of the 7 countries in Central America, which consider Spanish an official language?" from dataset CWQ, we visualize the retrieval process of FastToG. Fig. 6 displays a snapshot of LLMs pruning with start community *Central America*. Note that the node *Central America* has 267 one-hop neighbors, making it hard to visualize. Tab. 4 shows the corresponding Graph2Text output of the communities or nodes. The left column shows part of the pruning with communities as the units to be selected, while the right column shows the nodes. As we can see, community as the basic

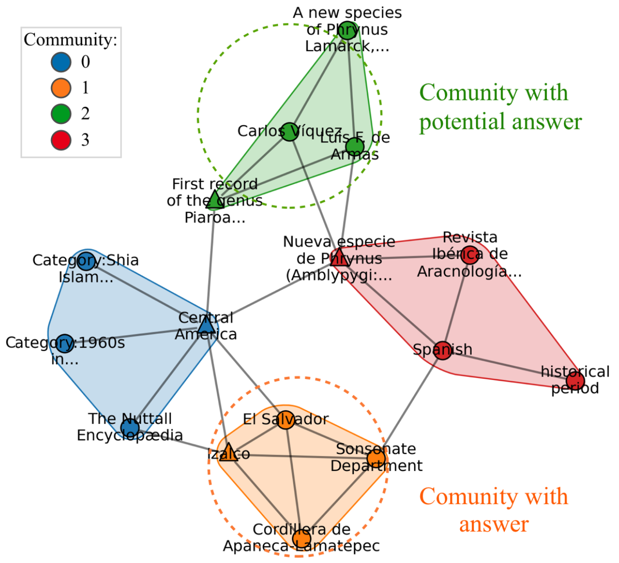

Figure 6: Visualization of the retrieval by FastToG

Think on Community	Think on Node
A. The Sonsonate Department is located in El Salvador, which is part of Central America. Spanish is the language used in the Sonsonate Department ...	A. El Salvador is belong to Central America
B. The Nuttall Encyclopdia describes Mexico City as a city in Central America, which is part of North America. The Centralist Republic of Mexico ...	B. Nueva especie de Phrynus is main subject in Central America.
C. The first record of genus Piaroa Villarreal, Giupponi & Tourinho, 2008, is from Central America. Carlos Vquez, a Panamanian author, has written about ...	... more than 20 options Z. The South and Central American Club is held in Central America.

Table 4: Think on Community versus Think on Node

unity for pruning greatly reduces the number of unit. In addition, Community as the unit can increase the likelihood of solving this problem. For example, comparing two option A, the community associated with the option A in left column has a path connecting to the entity *Spanish*. Thus, we can sure that *EI Salvador* is one of the answers. In contrast, when using Node as the unit, it requires more exploration to reach this entity. Hence, "think" on the community not only simplifies the pruning process for LLMs but also enhances clarity for users in understanding the reasoning process.

Conclusion

We introduced a novel GraphRAG paradigm - FastToG. By enabling large models to think "community by community" on the knowledge graphs, FastToG not only improves the accuracy of answers generated by LLMs but also enhances the efficiency of the RAG system. We also identified areas for further improvement, such as incorporating semantics and syntax with community detection, introducing multi-level hierarchical communities to enhance retrieval efficiency, and developing better community-to-text conversion.

Acknowledgments

This work was supported in part by the Major Key Project of PCL (No. PCL2024A05), National Natural Science Foundation of China (Grant No. 62372137), Shenzhen Science and Technology Program (No. KJZD20231023094701003), and the opening project of science and technology on communication networks laboratory (No. FFX24641X007).

References

Andrus, B. R.; Nasiri, Y.; Cui, S.; Cullen, B.; and Fulda, N. 2022. Enhanced story comprehension for large language models through dynamic document-based knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10436–10444.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, 722–735. Springer.

Baek, J.; Aji, A. F.; and Saffari, A. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10): P10008.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chan, C.-M.; Xu, C.; Yuan, R.; Luo, H.; Xue, W.; Guo, Y.; and Fu, J. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Deldjoo, Y.; He, Z.; McAuley, J.; Korikov, A.; Sanner, S.; Ramisa, A.; Vidal, R.; Sathiamoorthy, M.; Kasirzadeh, A.; and Milano, S. 2024. A Review of Modern Recommender Systems Using Generative Models (Gen-RecSys). *arXiv preprint arXiv:2404.00579*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; and Larson, J. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Elsahar, H.; Vougiouklis, P.; Remaci, A.; Gravier, C.; Hare, J.; Laforest, F.; and Simperl, E. 2018. T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In chair), N. C. C.; Choukri, K.; Cieri, C.; Declerck, T.; Goggi, S.; Hasida, K.; Isahara, H.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; Piperidis, S.; and Tokunaga, T., eds., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; and Wang, H. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Girvan, M.; and Newman, M. E. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12): 7821–7826.

Jiang, J.; Zhou, K.; Dong, Z.; Ye, K.; Zhao, W. X.; and Wen, J.-R. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Karpukhin, V.; Oğuz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, X.; Zhao, R.; Chia, Y. K.; Ding, B.; Bing, L.; Joty, S.; and Poria, S. 2023. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*, 3.

Liang, K.; Meng, L.; Liu, M.; Liu, Y.; Tu, W.; Wang, S.; Zhou, S.; Liu, X.; Sun, F.; and He, K. 2024. A Survey of Knowledge Graph Reasoning on Graph Types: Static, Dynamic, and Multi-Modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 9456–9478.

Modarressi, A.; Imani, A.; Fayyaz, M.; and Schütze, H. 2023. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*.

Ni, J.; Young, T.; Pandelea, V.; Xue, F.; and Cambria, E. 2023. Recent advances in deep learning based dialogue systems: A systematic survey. *Artificial intelligence review*, 56(4): 3055–3155.

Onoe, Y.; Zhang, M. J.; Choi, E.; and Durrett, G. 2021. Creak: A dataset for commonsense reasoning over entity knowledge. *arXiv preprint arXiv:2109.01653*.

Perevalov, A.; Diefenbach, D.; Usbeck, R.; and Both, A. 2022. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, 229–234. IEEE.

Petroni, F.; Piktus, A.; Fan, A.; Lewis, P.; Yazdani, M.; De Cao, N.; Thorne, J.; Jernite, Y.; Karpukhin, V.; Maillard, J.; Plachouras, V.; Rocktäschel, T.; and Riedel, S. 2021. KILT: a Benchmark for Knowledge Intensive Language Tasks. In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tur, D.; Beltagy, I.; Bethard, S.; Cotterell, R.;

Chakraborty, T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2523–2544. Online: Association for Computational Linguistics.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Sun, J.; Xu, C.; Tang, L.; Wang, S.; Lin, C.; Gong, Y.; Shum, H.-Y.; and Guo, J. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Talmor, A.; and Berant, J. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In Walker, M.; Ji, H.; and Stent, A., eds., *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 641–651. New Orleans, Louisiana: Association for Computational Linguistics.

Thurnbauer, M.; Reisinger, J.; Goller, C.; and Fischer, A. 2023. Towards Resolving Word Ambiguity with Word Embeddings. *arXiv preprint arXiv:2307.13417*.

Traag, V. A.; Waltman, L.; and Van Eck, N. J. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1): 1–12.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10): 78–85.

Wang, J.; Huang, W.; Shi, Q.; Wang, H.; Qiu, M.; Li, X.; and Gao, M. 2022a. Knowledge prompting in pre-trained language model for natural language understanding. *arXiv preprint arXiv:2210.08536*.

Wang, R.; Tang, D.; Duan, N.; Wei, Z.; Huang, X.; Cao, G.; Jiang, D.; Zhou, M.; et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Y.; Lipka, N.; Rossi, R. A.; Siu, A.; Zhang, R.; and Derr, T. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19206–19214.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xie, T.; Wu, C. H.; Shi, P.; Zhong, R.; Scholak, T.; Yasunaga, M.; Wu, C.-S.; Zhong, M.; Yin, P.; Wang, S. I.; et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Yamada, I.; Asai, A.; Shindo, H.; Takeda, H.; and Matsumoto, Y. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. *arXiv preprint arXiv:2010.01057*.

Yih, W.-t.; Richardson, M.; Meek, C.; Chang, M.-W.; and Suh, J. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In Erk, K.; and Smith, N. A., eds., *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 201–206. Berlin, Germany: Association for Computational Linguistics.

Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; and Liu, Q. 2019. ERNIE: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

Zheng, C.; Zhou, H.; Meng, F.; Zhou, J.; and Huang, M. 2023. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.