2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS) | 979-8-3315-1596-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICPADS63350.2024.00042

Enhancing Large Language Models with Knowledge Graphs for Robust Question Answering

Zhui Zhu*, Guangpeng Qi†, Guangyong Shang†, Qingfeng He‡, Weichen Zhang§,
Ningbo Li§, Yunzhi Chen†, Lijun Hu†, Wenqiang Zhang†, Fan Dang§✉
*Department of Automation and BNRist, Tsinghua University, †Inspur Yunzhou Industrial Internet Co., Ltd
‡Department of Computer Science and Technology, Tsinghua University, §Global Innovation Exchange, Tsinghua University
{z-zhu22,heqf21,weic_zhang23,lnb23}@mails.tsinghua.edu.cn
{qigp,shangguangyong,chenyunzhi,hljun,zhangwq01}@inspur.com
dangfan@tsinghua.edu.cn

***Abstract*—In recent years, large language models (LLMs) have shown rapid development, becoming one of the most popular topics in the field of artificial intelligence. LLMs have demonstrated powerful generalization and learning capabilities, and their performance on various language tasks has been remarkable. Despite their successes, LLMs face significant challenges, particularly in domain-specific tasks that require structured knowledge, often leading to issues such as hallucinations. To mitigate these challenges, we propose a novel system, SynaptiQA, which integrates LLMs with Knowledge Graphs (KGs) to answer more questions about knowledge. Our approach leverages the generative capabilities of LLMs to create and optimize KG queries, thereby improving the accuracy and contextual relevance of responses. Experimental results in an industrial data set demonstrate that SynaptiQA outperforms baseline models and naive retrieval-augmented generation (RAG) systems, demonstrating improved accuracy and reduced hallucinations. This integration of KGs with LLMs paves the way for more reliable and interpretable domain-specific question answering systems.**

***Index Terms*—Artificial Intelligence, Knowledge Graph, Large Language Model**

I. Introduction

With the rapid development of natural language processing (NLP) technology, large language models (LLMs) such as GPT-3 [1] and BERT [2] have demonstrated outstanding performance in various language tasks. These models, through pre-training and fine-tuning, are capable of achieving impressive results in tasks such as text generation, question answering, and language translation [1]–[3]. Leveraging massive pre-training data and complex deep learning architectures, these large language models can capture complex linguistic patterns, making them perform better in many NLP tasks compared with other models [4], [5].

However, despite the impressive performance of large language models across a wide range of tasks, they still face several significant challenges. One notable challenge is that large language models primarily rely on large-scale text data for training, which result in suboptimal performance when dealing with domain-specific task [5]. large language models lack a deep understanding of structured knowledge and they perform not good for the questions which require accurate and reliable domain knowledge [6]. Large language models primarily rely on statistical learning to generate answers, rather than being based on explicit knowledge bases or external information sources [7]. This leads the problem of nonsensical or unfaithful answer, also called "hallucinations", ultimately affecting their practical application [6], [8].

Hallucinations typically refer to the phenomenon where generated content appears absurd or inconsistent with the provided source material [6]. This issue has widespread implications across various applications, particularly in fields that demand high accuracy of results, such as medicine and industry.

The utilization of retrieval-augmented generation (RAG) techniques presents a solution for reducing hallucination. RAG techniques retrieve information from data sources to assist large language models in generating answers, enabling the models to access external sources of information and produce more accurate responses [7]. In existing RAG methods, external information is typically encoded as vector data. And semantic similarity is leveraged to retrieve relevant document passages from external knowledge repositories. This process enhances the performance of the large language model by incorporating the retrieved information [7].

However, when data is processed and stored as vectors, the relationship between the context is often lost, and relationships between data points can become particularly ambiguous [9]. Additionally, the effectiveness of the RAG methods heavily relies on the segmentation of the document chunks, as incorrect or inappropriate document segmentation can significantly impact its effectiveness [9].

Knowledge Graph (KG), as a structured knowledge representation method, is capable of effectively organizing and storing a large amount of domain knowledge [10]. By representing entities and their relationships in the form of nodes and edges, KGs serves as an intuitive and efficient knowledge management tool, like Wikidata [11] and YAGO [12]. KGs not only capture complex relationships and attributes of the data but also enable reasoning and querying through graph algorithms, providing more accurate and reliable knowledge support.

The integration of KG with LLM holds the potential to

✉Fan Dang is the corresponding author.

979-8-3315-1596-6/24/$31.00 ©2024 IEEE
DOI 10.1109/ICPADS63350.2024.00042

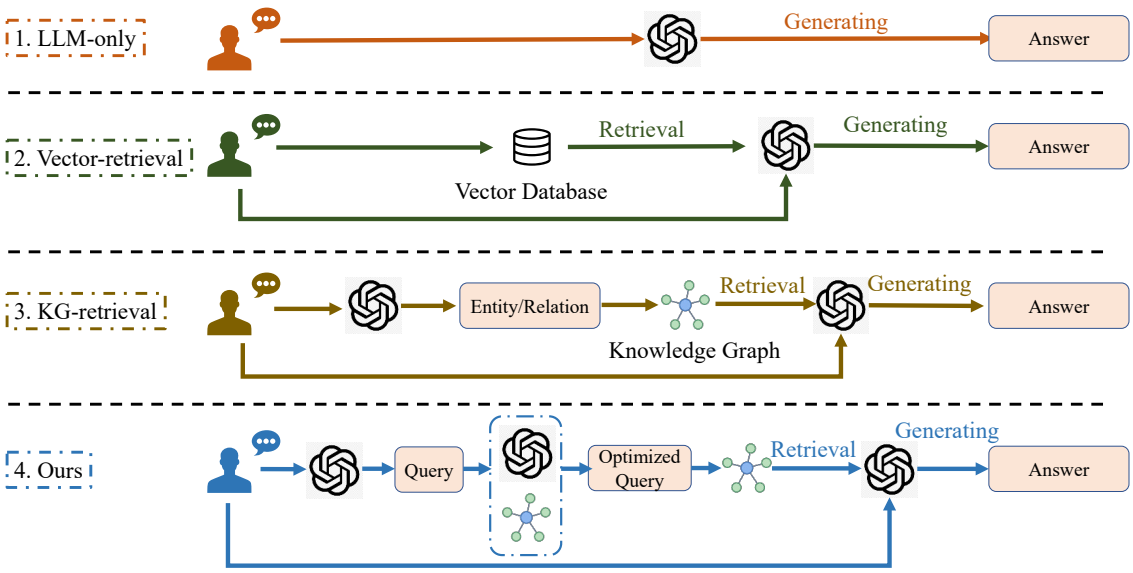

Fig. 1. A comparison between our system and other baseline system

enhance the knowledge usage of the models, improve their understanding of contextual information of the data and generate interpretable results. Due to the advantages of KG, researchers are attempting to integrate them in order to compensate for certain limitations of LLMs [10]. For example, ERNIE [13] attempts to use the KG triples to improve the training process of LLMs and KAPING [14] retrieves facts related to the input question from the KG and inject them to the prompts of LLMs to generate better output.

However, existing approaches to enhance LLMs with KGs still have some limitations. These limitations primarily manifest in the inefficient utilization of knowledge within the graph and the high complexity of querying the knowledge graph.

A straightforward approach is to directly utilize LLMs to query KGs. This is feasible because LLMs have robust generative and generalization capabilities, making such a method viable. However, we observed that during the generation process, the sentences produced by LLMs are highly unstable and not suitable for direct querying. Moreover, knowledge graphs are highly sensitive to the input queries, making precision in query formulation crucial.

To address above challenges, we propose SynaptiQA, a novel system that combines LLMs and KGs for collaborative knowledge question answering. We have repeatedly used large models and knowledge graphs, and innovatively leverage the capabilities of LLMs to generate knowledge graph queries and employ word embedding techniques to optimize the queries. SynaptiQA fully harnesses the strong generalization and comprehension abilities of LLMs, along with the accuracy and contextual information advantages of KGs.

We conducted experiments on an industrial dataset, which includes a knowledge graph with over 3,000 nodes and more than 10,000 relationships. Our system demonstrated superior performance in terms of BERTScore compared to the naive RAG system and large models without knowledge enhancement. Specifically, our system achieved accuracy improvements of 0.26 and 0.03 over the large model without knowledge enhancement and the naive RAG system, respectively.

In conclusion, we propose SynaptiQA to improve the perfomance of KGQA problem. SynaptiQA fully leverages the generation capability of LLMs and use word embedding to enhance the efficiency and quality of knowledge graph queries. Our system overperforms the baseline system and reduce the hallucinations in domain-specific KGQA problem.

II. Background

A. Large Language Model

Large Language Models (LLMs) are complex natural language processing models based on deep learning techniques, particularly the Transformer [15] architecture. Models like GPT-3 [1], LLAMA [3], [16], and BERT [2] have been trained on large-scale text data and are capable of generating high-quality natural language and understanding complex language tasks. By learning the relationships between words, these models perform exceptionally well on various natural language processing tasks. However, despite their impressive performance in language generation and understanding, LLMs still face challenges such as accuracy of results and limitations in domain-specific questions.

B. Prompt Engineering

Prompt engineering has emerged as a pivotal technology as the development of LLMs. It refers to the process of designing and optimizing input prompts to guide language models in generating desired outputs [17], [18]. This technique plays a

significant role in enhancing the performance and application scope of language models.

The core of prompt engineering lies in constructing effective prompts that can fully leverage the knowledge and capabilities embedded within pre-trained models. By meticulously crafting these prompts, researchers and engineers can significantly improve the performance of models on specific tasks without requiring additional training.

Researchers have proposed various strategies, including manually designing prompts [1], [19], automating prompt generation [20], and optimizing prompts for multiple tasks. These approaches not only enhance the practical utility of models but also uncover the intricate semantic structures and knowledge representation mechanisms within pre-trained models.

C. Knowledge-Augmented LLMs

For LLMs, it is currently not possible to store all the knowledge in the world, and they often suffer from "hallucination" problems when dealing with domain-specific questions [6], [8]. To address these limitations of LLMs, researchers have proposed Knowledge-Enhanced Language Models (KELMs) [21], [22]. These models enhance their semantic understanding and reasoning capabilities by incorporating external sources of knowledge. These external knowledge sources can include structured knowledge such as knowledge graphs or unstructured information like Wikipedia [23]. KELMs can provide more accurate answers to questions involving domain-specific knowledge and improve the credibility and consistency of their responses. Common approaches include embedding information from knowledge bases into the input representation of large language models or dynamically querying [24], [25] and utilizing external knowledge during the model's generation process [26], [27].

D. Knowledge Graph

Knowledge Graph (KG) is a graph data system used to represent and store structured knowledge. It often consists of nodes (representing entities) and edges (representing relationships between entities) and is often represented in the form of triplets to denote knowledge. In addition to entities and relationships, KG includes rich semantic information, hierarchical structures, and inference rules, enabling efficient querying and reasoning through knowledge. Compared to other knowledge storage methods, like vector databases, KGs can store complete contextual information and allow for querying using graph structures [10]. Leveraging these advantages, Knowledge Graphs have found widespread applications in fields such as search engines [28], [29], recommendation systems [30], and intelligent question answering [31], [32].

E. Knowledge Graph Question Answering

Knowledge Graph Question Answering (KGQA) refers to the technology that utilizes KGs to answer natural language questions [10], [33]. The core of a KGQA system lies in correctly mapping the natural language question to relevant entities and relationships within the Knowledge Graph, and retrieving accurate answers from it. KGQA systems typically involve steps such as natural language understanding, Knowledge Graph querying, and answer generation.

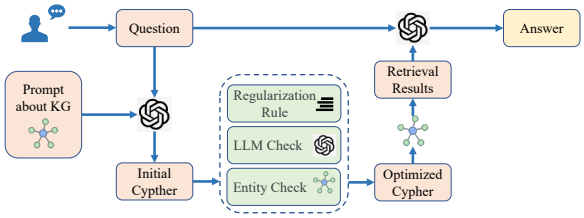

Fig. 2. Architecture of our System

Previous research has proposed various approaches to enhance the performance of KGQA, such as neural semantic parsing [34], [35], information retrieval [36], and differentiable knowledge graph [37].

The development of LLMs has provided new opportunities for KGQA research. LLMs possess powerful abilities in text comprehension and knowledge reasoning, which have greatly improved the performance of KGQA. ChatKBQA [38] utilizes large language models to transform textual questions into logical forms, which are then used for querying the knowledge graph. KAPING [14] embeds the information of knowledge graph triplets into prompts of LLMs.

However, existing methods have limitations in harnessing the full potential of large language models. They either rely on these models solely for answering questions without fully leveraging their capabilities. Additionally, current approaches often focus on utilizing only the triplet information of the knowledge graph, overlooking its rich hierarchical information such as attribute details and graph structure. Therefore, based on these observations, we propose the SynaptiQA model. SynaptiQA directly utilizes the large language model to generate graph query based on the input questions. It then optimizes the query statements using the graph information and extracts precise information from the knowledge graph to generate the final answer.

III. System

Our system consists of four main components: query generation, query optimization, KG querying, and QA result generation. In the first two steps, we leverage the LLMs' powerful generalization capabilities to directly generate and optimize the query statements for KG retrieval. Then, combining the retrieval results from the KG, we utilize the LLMs to obtain the final answer. The following sections describe each component in detail and how they interconnect to produce accurate QA results.

A. Query Generation

To fully leverage the generative capabilities of large models, we directly utilize them to generate query statements. During the generation process, in order to ensure that the generated

statements meet the requirements of KG queries, we integrate common querying methods of KGs and embed this information into the input of the large language model using prompt engineering.

We will extract common query statement formats and keyword formats from the related manual of KG. Additionally, we will retrieve the types of nodes and relationships, as well as potentially useful attribute keywords, from the KG we are using.

However, for a knowledge graph, there are numerous feasible query types and use cases, which can make it challenging for the model to determine how to utilize them effectively. Additionally, an excessive number of rules can degrade the model's performance. Therefore, we will search historical data to find problems similar to the current issue and use the corresponding query statements as few-shot data.

B. Query Optimization

The queries generated by the large model may not directly apply to the knowledge graph due to three main reasons: (1) The generated responses may include irrelevant content, such as "Here is the Cypher I generated." (2) The generated query statements may contain syntax errors. (3) The content being queried may not exist within the knowledge graph. To address these three issues, we implemented the following optimizations:

Statement Optimization: To tackle the redundancy in the answers generated by large models, we designed a regularization method to achieve the truly effective parts of the query statements. And we use LLM to clean the query statement one again. We then validated the effectiveness of these statements based on predefined rules. Furthermore, we utilized open-source code [39] to verify and correct the direction of the query statements.

Query Validity Verification: The KGs are highly organized and structured data model. This characteristic allows them to retain a vast amount of information effectively and allows them for the efficient utilization of its structure for searching. However, it also makes the KG itself challenging to manipulate. To address this, many researchers have delved into the field of knowledge graph embedding (KGE) [40], [41]. The key idea is to embed the components of the KG, including entities and relationships, into continuous vector spaces, thereby simplifying operations while preserving the original graph structure of the KG. Nevertheless, traditional word embedding systems are typically designed for direct querying and do not differentiate between relations and attributes, making them unsuitable for our task of optimizing query statements generated by LLMs.

Therefore, we have specifically improved KGE for our purposes. First, we embed all information from the knowledge graph, including all entities, relations, and attributes, using the m3e model. During querying, we analyze the query content to extract all relevant information. For each entity and relation, we utilize their embedding vectors for comparison and analysis. Unlike most embedding methods, we assign separate embedding vectors to each attribute, ensuring flexibility in selecting the most similar knowledge graph elements based on the query statements generated by the LLMs.

Based on the identified similar knowledge graph elements, we use the LLMs to select the most appropriate query option, replacing the original query content to generate the final query statement.

C. KG Query and Answer Generation

We utilize the optimized query statements to query KG. Compared to other methods that utilize knowledge graphs, such as querying entity neighborhood information [42], our approach offers greater flexibility in generating queries and retrieval requirements based on the specific context of the problem. For relatively simple questions, we generate straightforward query statements, like *MATCH (n1:Fault_name content:'Cold Trough')-[:relation]-(n2:Maintenance_action) RETURN n2.content*. And for more complex questions, we leverage the inferential capabilities of large models to directly generate intricate queries. For example, *MATCH (p1:Fault_name content:'Cold Trough'),(p2:Normal_status), p=shortestpath((p1)-[*..10]- (p2)), RETURN p*. This approach avoids the insufficiency of neighborhood information in addressing complex issues.

Subsequently SynaptiQA employs LLM to generate the final QA results. Apart from the template, the prompt includes all information retrieved from the knowledge graph and the original question. If the knowledge graph returns no results, we allow the large language model to provide an answer based on its own knowledge, along with a confidence level. By embedding both the results of the KG query and the original question into the output of the LLMs, we generate the final answer.

IV. Evaluation

A. Experiment Setting

1) Implementation: Our entire system is built using the LangChain [43] development framework, with the KG implemented using Neo4j [44]. The management and querying of the knowledge graph are performed using the Cypher language.

2) Data: For the document data, we utilized industrial data collected by ourselves and generated a KG by integrating LLMs with human expertise. This KG comprises 3,224 nodes and 13,312 relationships, with each node and relationship including additional attribute information. Regarding the question-answering aspect, we conducted experiments using 209 manually crafted questions. These questions and corresponding data pertain to operating procedures, production processes, and process mechanisms.

3) Model: The word embedding model we employed is the m3e-large [45] model, and the large-scale model was tested using Qwen-4B [46].

4) Metrics: We employ two evaluation metrics to assess our system:

BERTScore: BERTScore [47] evaluates the similarity between the output and the actual data source by using the cosine

TABLE I
MAIN RESULT

Method	Kimi Accuracy	BERTScore
Vector-retrieval	0.7033	0.7830
LLM-only	0.4737	0.6730
KG-retrieval	0.6077	0.7340
SynaptiQA	0.7368	0.8012

TABLE II
RESULTS IN DIFFERENT FIELDS

Method	**Dataset**		
	Operating Procedure	**Process Mechanism**	**Production Process**
Vector-retrieval	0.6990	0.8286	0.6479
LLM-only	0.4757	0.4571	0.4789
KG-retrieval	0.5340	0.77140	0.6338
SynaptiQA	0.7476	0.8857	0.6479

similarity of the embedding encodings generated by the BERT [2] model.

Kimi Accuracy: Kimi Accuracy leverages the Kimi [48] LLM to align the generated results with the actual data source, determining whether the generated answers fully match the actual data source.

BERTScore takes into account contextual and semantic information, allowing for a more accurate measurement of sentence similarity, especially for sentences that are semantically similar but expressed differently. However, BERTScore may still produce erroneous judgments for some results and lacks a precise characterization. Kimi Accuracy uses a LLM as an evaluation metric, which can provide a better assessment. Nevertheless, due to the model's sensitivity to prompts, it can only serve as a reference metric. By combining both metrics, we can achieve a more comprehensive evaluation of our system.

5) Baseline Model: Our system will be compared with the following systems:

Vector-retrieval: Utilizing a vector database to store document data. During the question-answering process, the question is first converted into a vector form, relevant documents are retrieved from the database, and finally, the LLM is used to generate the answer.

KG-retrieval: Employing a KG to store document data. When answering the question, the LLM is used to extract entities. The attributes of the entities, along with their one-hop relationships, are queried from the knowledge graph. All retrieved information is then combined with the LLM to generate the answer.

LLM-only: Directly utilizes a large language model without leveraging a knowledge graph or data base for answering questions.

B. Main Result

The primary results are presented in Table I. Comparatively, our method achieves the highest scores in both BERTScore and Kimi Accuracy metrics, with a Kimi Accuracy of 0.737 and a BERTScore of 0.801, outperforming the other four baseline methods. Specifically, the LLM-only method performs the worst, with an accuracy of only 0.474. This is mainly because, for industrial datasets, many pieces of knowledge are not included in the pre-training process of LLMs, such as knowledge encompasses details in industrial production and proprietary knowledge within enterprises. Therefore, knowledge augmentation for domain-specific problems has significant application value.

Among the various knowledge augmentation methods, the Vector-retrieval method performs second best, with an accuracy of 0.703 and a BERTScore of 0.783, whereas the performance of KG-retrieval methods is relatively less impressive. To explain this phenomenon, we conducted further observations. Although the KG-retrieval method can ensure the retrieval of relevant results, it can only fetch answers within a single hop and often returns a significant amount of confusing information. In contrast, our method effectively leverages both the generation capabilities of large models and the structured querying capability of knowledge graphs, thereby providing more accurate responses.

TABLE III
ABLATION STUDY RESULT

Method	Kimi Accuracy	BERTScore
LLM-only	0.4737	0.6730
KG-query	0.5407	0.7094
SynaptiQA	0.7368	0.8012

C. Results in different fields

Furthermore, we evaluated the results of questions across different domains. Our system consistently demonstrated optimal performance across various domains, highlighting both the stability of our system and the feasibility of large model enhancement based on knowledge graphs.

Among all the domains, knowledge augmentation showed the most significant improvement in the area of process mechanisms. This is likely because knowledge in process mechanisms demands the highest accuracy, and the documentation and graphs related to this area are most clearly described. Consequently, the knowledge augmentation approach significantly enhances the application capability of large models in this domain.

In every domain, the knowledge augmentation approach improved the efficiency of large models, underscoring the necessity of using knowledge augmentation in industrial contexts.

D. Ablation Study

To further verify our system design, we designed an ablation experiment with the KG-query method (Table III). **KG-query** uses a KG to store document data and employs the LLM to generate query statements. These query statements are not

optimized. The output from the KG is then combined with the LLM to generate the answer.

As discussed in Section 3.2, queries generated by large models are often unstable. When these generated queries are directly used for knowledge graph retrieval, they frequently return an empty set, meaning no results are found, leading to the poor performance of the KG-query method.

Meanwhile, our design effectively mitigates the instability of queries generated by large models. By integrating knowledge graphs and query rules, we enhance query effectiveness, thereby improving the overall system performance.

V. RELATED WORK

A. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is an approach that combines retrieval and generation techniques to enhance the performance of generative models by retrieving relevant information from external knowledge bases [7]. The core idea of the RAG method is to use a retrieval module to obtain relevant documents or fragments from a pre-constructed knowledge base, and then integrate these retrieved pieces of information with the input to the generative model to produce more accurate and information-rich responses [7], [49], [50]. The RAG method excels in tasks requiring external knowledge, such as open-domain question answering, dialogue systems, and knowledge-intensive text generation.

Typically, this approach follows a traditional workflow that includes indexing, retrieval, and generation steps [7], [49], [50], often described as the "retrieve-read" framework [51]. In RAG, the retrieval module is responsible for identifying documents or fragments relevant to the input query from a large knowledge base, commonly using vector databases (such as chroma [52] and Pinecone [53]) and vector-based retrieval methods.

Existing researchers are exploring ways to enhance the performance of RAG systems by improving the traditional retrieval methods through techniques such as pre-retrieval [51], [54], [55], post-retrieval [56], and modular RAG approaches [57], [58]. Additionally, some researchers are investigating how to more closely integrate large model fine-tuning with the RAG pipeline to enable the large model to acquire knowledge related to RAG retrieval during the training phase [59], [60].

B. Traditional KGQA Systems

[34] employs neural network-based models to parse and interpret the semantics of natural language questions. Subsequently, it decomposes the natural language question into multiple subtasks, and progressively constructs a query graph. Each stage generates a partial query graph, with these partial graphs ultimately combining to form a complete query graph. [35] also utilizes the same paradigm, employing neural networks to encode both the natural language questions and the knowledge graph.However, due to the lack of semantic representation capabilities in traditional deep neural networks, these methods have not achieved satisfactory accuracy on KGQA problems.

C. LLM Based KGQA Systems

The powerful capabilities of LLMs have given KGQA greater potential, so many researchers have begun to explore the combination of the two [14], [26], [38]. Among them, ChatKBQA [38] employs large language model to first generate logical form and then retrieve information from knowledge graph to assist question answering tasks. However, ChatKBQA's way of generating logical form may not fits knowledge graph's reasoning process, causing hallucination in large language model's final response. KAPING [14] offers a framework for identifying triples relevant to a given question, thereby extracting pertinent information from the knowledge graph to assist a large language model in generating responses. However, both methods fail to align the language model with knowledge graph's structure, thus may fail to retrieve truly correct information from knowledge graph.

VI. FUTURE WORK

For future work, we plan to investigate the possibility of combining knowledge graphs and vector databases to enhance the inference stage of large language models. While knowledge graphs have certain advantages over vector databases, the latter possess a vast capacity for information. Therefore, exploring the synergy between these two primary knowledge sources holds promise. Current work necessitates a fine-tuning process to align the model with the knowledge graph, and the quality of question-answering in large language models is closely associated with the quality of the knowledge graph. We intend to reconstruct the knowledge graph based on the feedback from large language models during the fine-tuning process, thereby obtaining knowledge that better aligns with the preferences of large language models.

On the other hand, our current application of large models remains relatively simple. Moving forward, we aim to leverage the strong reasoning capabilities of large models to further enhance the performance of KGQA. We can employ more advanced prompting methods, such as Chain of Thought (CoT) [19] and Tree of Thought (ToT) [61], to stimulate the reasoning processes of LLMs and reveal their inference paths based on the knowledge graph. Furthermore, we intend to explore the joint training of knowledge graphs and large models. By fine-tuning, we aim to integrate the knowledge and querying methods of the knowledge graph into the LLM, thereby enabling better synergy between large models and knowledge graphs.

VII. CONCLUSION

We propose SynaptiQA to optimize the effectiveness of end-to-end question answering tasks and reduce hallucinations. SynaptiQA leverages large language model and word embedding to enhance the efficiency and quality of knowledge graph queries. By incorporating the structured knowledge of knowledge graphs into the inference process of LLM, this approach mitigates hallucinations in domain-specific question answering task and improves the quality of responses. The system is test on an industrial dataset and overperform other

baseline methods on both BERTScore and Kimi Accuracy metrics.

We promote the alignment of large language models with structured knowledge graphs through word embedding and fine-tuning, demonstrating the vast potential of combining large language models with structured knowledge. Further, the success of SynaptiQA illustrates that structured knowledge graphs can better support the inference of large language models in complex tasks compared to vector knowledge databases.This may be attributed to the current limitations of large language models' reasoning capabilities for more complex tasks. The inherent logical chains of knowledge graphs can assist in reasoning, a capacity that vector knowledge databases lack.

REFERENCES

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

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[4] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.

[5] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[6] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *arXiv preprint arXiv:2311.05232*, 2023.

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

[8] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

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

[10] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[11] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[12] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[13] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," 2019. [Online]. Available: https://arxiv.org/abs/1905.07129

[14] J. Baek, A. F. Aji, and A. Saffari, "Knowledge-augmented language model prompting for zero-shot knowledge graph question answering," 2023. [Online]. Available: https://arxiv.org/abs/2306.04136

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[16] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[17] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021. [Online]. Available: https://arxiv.org/abs/2107.13586

[18] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, and P. Resnik, "The prompt report: A systematic survey of prompting techniques," 2024. [Online]. Available: https://arxiv.org/abs/2406.06608

[19] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903

[20] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," 2023. [Online]. Available: https://arxiv.org/abs/2211.01910

[21] L. Hu, Z. Liu, Z. Zhao, L. Hou, L. Nie, and J. Li, "A survey of knowledge enhanced pre-trained language models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 4, pp. 1413–1430, 2024.

[22] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2312.10997

[23] [Online]. Available: https://www.wikipedia.org/

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

[25] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," *arXiv preprint arXiv:1905.07129*, 2019.

[26] Y. Wen, Z. Wang, and J. Sun, "Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models," *arXiv preprint arXiv:2308.09729*, 2023.

[27] W. Su, Y. Tang, Q. Ai, Z. Wu, and Y. Liu, "Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models," *arXiv preprint arXiv:2403.10081*, 2024.

[28] P. Wang, H. Jiang, J. Xu, and Q. Zhang, "Knowledge graph construction and applications for web search and beyond," *Data Intelligence*, vol. 1, no. 4, pp. 333–349, 2019.

[29] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 1271–1279.

[30] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3549–3568, 2020.

[31] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 105–113.

[32] J. Jiang, K. Zhou, W. X. Zhao, and J.-R. Wen, "Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph," *arXiv preprint arXiv:2212.00959*, 2022.

[33] B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, and J. Sun, "A survey on complex question answering over knowledge base: Recent advances and challenges," *arXiv preprint arXiv:2007.13069*, 2020.

[34] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*, 2015.

[35] K. Luo, F. Lin, X. Luo, and K. Zhu, "Knowledge base question answering via encoding of complex query graphs," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 2185–2194.

[36] A. Saxena, A. Tripathi, and P. Talukdar, "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 4498–4507.
[37] P. Sen, A. Saffari, and A. Oliya, "Expanding end-to-end question answering on differentiable knowledge graphs with intersection," *arXiv preprint arXiv:2109.05808*, 2021.
[38] H. Luo, Z. Tang, S. Peng, Y. Guo, W. Zhang, C. Ma, G. Dong, M. Song, W. Lin *et al.*, "Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models," *arXiv preprint arXiv:2310.08975*, 2023.
[39] "cypher-direction-competition." [Online]. Available: https://github.com/sakusaku-rich/cypher-direction-competition/
[40] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 105–113.
[41] A. Saxena, A. Tripathi, and P. Talukdar, "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 4498–4507.
[42] "Enhancing rag-based application accuracy by constructing and leveraging knowledge graphs." [Online]. Available: https://blog.langchain.dev/enhancing-rag-based-applications-accuracy-by-constructing-and-leveraging-knowledge-graphs/
[43] "LangChain," 2022. [Online]. Available: https://github.com/langchain-ai/langchain
[44] "Neo4j." [Online]. Available: https://neo4j.com/
[45] H. s. Wang Yuxin, Sun Qingxuan, "M3e: Moka massive mixed embedding model," 2023.
[46] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.
[47] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.
[48] "Kimi," 2023. [Online]. Available: https://kimi.moonshot.cn/
[49] "What is rag (retrieval-augmented generation)?" [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation
[50] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
[51] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, "Query rewriting for retrieval-augmented large language models," *arXiv preprint arXiv:2305.14283*, 2023.
[52] "chroma." [Online]. Available: https://www.trychroma.com/
[53] "Pinecone." [Online]. Available: https://www.pinecone.io/
[54] W. Peng, G. Li, Y. Jiang, Z. Wang, D. Ou, X. Zeng, D. Xu, T. Xu, and E. Chen, "Large language model based long-tail query rewriting in taobao search," in *Companion Proceedings of the ACM on Web Conference 2024*, 2024, pp. 20–28.
[55] H. S. Zheng, S. Mishra, X. Chen, H.-T. Cheng, E. H. Chi, Q. V. Le, and D. Zhou, "Take a step back: Evoking reasoning via abstraction in large language models," *arXiv preprint arXiv:2310.06117*, 2023.
[56] V. Blagojevi, "Enhancing rag pipelines in haystack: Introducing diversityranker and lostinthemiddleranker," 2023.
[57] W. Yu, D. Iter, S. Wang, Y. Xu, M. Ju, S. Sanyal, C. Zhu, M. Zeng, and M. Jiang, "Generate rather than retrieve: Large language models are strong context generators," *arXiv preprint arXiv:2209.10063*, 2022.
[58] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," *arXiv preprint arXiv:2305.15294*, 2023.
[59] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "Raft: Adapting language model to domain specific rag," *arXiv preprint arXiv:2403.10131*, 2024.
[60] Z. Liu, W. Ping, R. Roy, P. Xu, M. Shoeybi, and B. Catanzaro, "Chatqa: Building gpt-4 level conversational qa models," *arXiv preprint arXiv:2401.10225*, 2024.
[61] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.