MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models

Yilin Wen[*], **Zifeng Wang**[1,*], **Jimeng Sun**[1],

[1] University of Illinois Urbana Champaign

Correspondence: jimeng@illinois.edu

Abstract

Large language models (LLMs) have achieved remarkable performance in natural language understanding and generation tasks. However, they often suffer from limitations such as difficulty in incorporating new knowledge, generating hallucinations, and explaining their reasoning process. To address these challenges, we propose a novel prompting pipeline, named MindMap, that leverages knowledge graphs (KGs) to enhance LLMs' inference and transparency. Our method enables LLMs to comprehend KG inputs and infer with a combination of implicit and external knowledge. Moreover, our method elicits the mind map of LLMs, which reveals their reasoning pathways based on the ontology of knowledge. We evaluate our method on diverse question & answering tasks, especially in medical domains, and show significant improvements over baselines. We also introduce a new hallucination evaluation benchmark and analyze the effects of different components of our method. Our results demonstrate the effectiveness and robustness of our method in merging knowledge from LLMs and KGs for combined inference. To reproduce our results and extend the framework further, we make our codebase available at https://github.com/wyl-willing/MindMap.

1 Introduction

Scaling large language models (LLMs) to billions of parameters and a training corpus of trillion words was proved to induce surprising performance in various tasks (Brown et al., 2020; Chowdhery et al., 2022). Pre-trained LLMs can be adapted to domain tasks with further fine-tuning (Singhal et al., 2023) or be aligned with human preferences with instruction-tuning (Ouyang et al., 2022). Nonetheless, several hurdles lie in the front of steering LLMs in production:

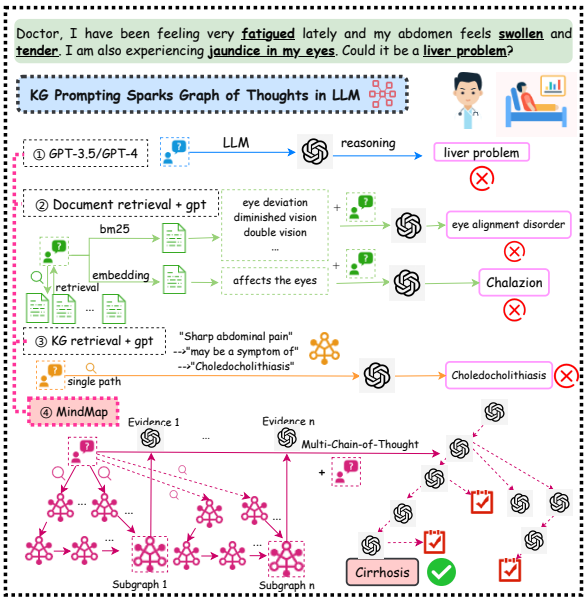

Figure 1: A conceptual comparison between our method and the other prompting baselines: LLM-only, document retrieval + LLM, and KG retrieval + LLM.

- **Inflexibility**. The pre-trained LLMs possess outdated knowledge and are inflexible to parameter updating. Fine-tuning LLMs can be tricky because either collecting high-quality instruction data and building the training pipeline can be costly (Cao et al., 2023), or continually fine-tuning LLMs renders a risk of catastrophic forgetting (Razdaibiedina et al., 2022).

- **Hallucination**. LLMs are notoriously known to produce hallucinations with plausible-sounding but wrong outputs (Ji et al., 2023), which causes serious concerns for high-stake applications such as medical diagnosis.

- **Transparency**. LLMs are also criticized for their lack of transparency due to the black-box nature (Danilevsky et al., 2020). The knowledge is implicitly stored in LLM's parameters, thus infeasible to be validated. Also, the inference process in deep neural networks remains elusive to be

[*]These authors contributed equally to this work.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10370–10388

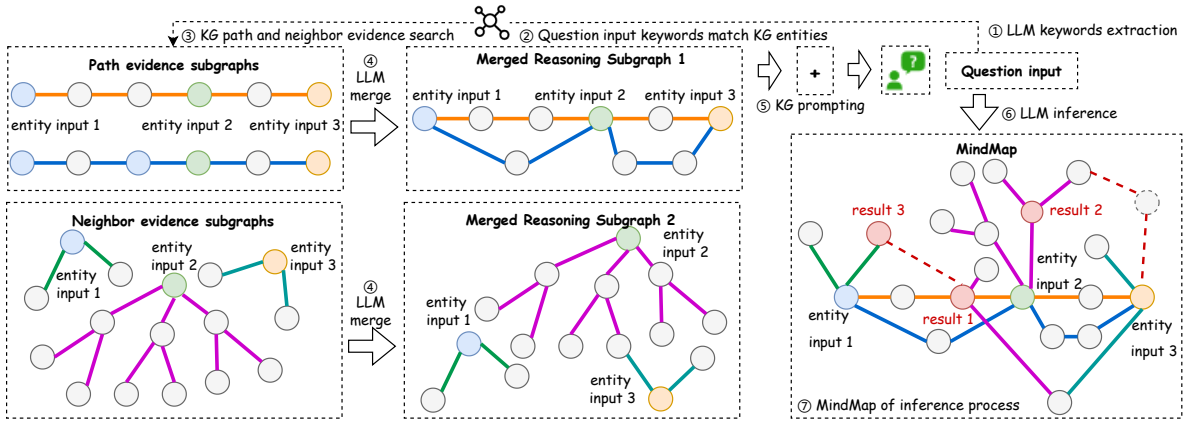

Figure 2: A conceptual demonstration of evidence query sub-graphs, merged reasoning sub-graphs, and mind map. The entity inputs $\mathcal{V}_q$ is identified from the input. Lines and circles of the same color indicate that they correspond. The red dashed lines in the `MindMap` box illustrate the augmentation operation based on the knowledge of LLM.

interpretable.

As a classic way to build large-scale structural knowledge bases, knowledge graphs (KG) are established by the triples of entities and relations, i.e., $\{\texttt{head}, \texttt{relation}, \texttt{tail}\}$. They can provide explicit knowledge representation and interpretable reasoning paths. Besides, KGs are amenable to continual modifications to debug the existing knowledge or add new knowledge. Due to their flexibility, preciseness, and interpretability, KGs emerged as a promising complement to the drawbacks of LLMs (Pan et al., 2023). For instance, KG triples were added to the training of LLMs (Zhang et al., 2019b) or KG encoders were entangled with LLM layers for joint inference and optimization on graph and text data (Zhang et al., 2022). By contrast, our work pivots on the synergistic inference of KGs and fixed LLMs, which is applicable to powerful pre-trained LLMs, such as commercial LLM-as-service APIs. In general, the prior arts in this venue can be categorized into two genres:

- **Retrieval-Augmented LLM Inference**. Researchers tried to retrieve documents to augment LLM inference (Lewis et al., 2020) while suffering from inaccurate retrieval and lengthy documents (Liu et al., 2023a). Recently, several attempts were made to incorporate extracted KG triples into the prompt to LLMs to answer KG-related questions (Baek et al., 2023). However, this approach treats KG inputs as plain text and ignores their graphical structure, which causes the generated response to be hard to validate and vulnerable to hallucinations.

- **Graph Mining with LLMs**. There were also attempts to prompt LLMs to comprehend graphical inputs, while they primarily experimented with graph mining tasks, e.g., edge detection and graph summarization (Guo et al., 2023; Chen et al., 2023). It was rarely explored in text generation tasks that require complex reasoning across multiple evidence graphs grounded on KGs.

The goal of this work is to build a plug-and-play prompting approach to elicit the graph-of-thoughts reasoning capability in LLMs. We call our method `MindMap` because it enables LLMs to comprehend graphical inputs to build their own mind map that supports evidence-grounded generation. A conceptual demonstration of our framework is in Figure 2. Specifically, `MindMap` sparks the graph of thoughts of LLMs that (1) consolidates the retrieved facts from KGs and the implicit knowledge from LLMs, (2) discovers new patterns in input KGs, and (3) reasons over the mind map to yield final outputs. We conducted experiments on three datasets to illustrate that `MindMap` outperforms a series of prompting approaches by a large margin. This work underscores how LLM can learn to conduct synergistic inference with KG. By integrating both implicit and explicit knowledge, LLMs can achieve transparent and dependable inference, adapting to different levels of correctness in additional KG information.

2 Related Work

Prompt Engineering. The "pre-train, prompt, and predict" paradigm has become the best practice for natural language processing in few-shot or zero-shot manners (Liu et al., 2023b). The core insight is LLMs are able to adapt to new tasks following

the input context and instructions via in-context learning (Brown et al., 2020), especially with instruction tuning (Wei et al., 2022a) and alignment (Ouyang et al., 2022). Retrieval-augmented generation emerged as a way to dynamically inject additional evidence for LLM inference (Lewis et al., 2020). The common practice is to query a dense embedding database to find the relevant document pieces to the input user questions, then put the retrieved corpus back to the prompt input. However, documents can be lengthy, thus not fitting into the context length limit of LLM. It was also identified even though we can build long documents as prompts, LLMs usually fail to capture information in the middle of the prompt and produce hallucinations (Liu et al., 2023a). Another line of research looks to prompt to elicit the intermediate reasoning steps of LLMs in chains (Wei et al., 2023) and trees (Yao et al., 2023a), while these approaches all focus on eliciting the implicit knowledge from LLMs. Nonetheless, our work explores sparking the reasoning of LLMs on graph inputs, with an emphasis on joint reasoning with implicit and external explicit knowledge.

Knowledge Graph Augmented LLM. Researchers have explored using knowledge graphs (KGs) to enhance LLMs in two main directions: (1) integrating KGs into LLM pre-training and (2) injecting KGs into LLM inference. For (1), it is a common practice to design knowledge-aware training objectives by either putting KG entities and relations into the training data (Zhang et al., 2019b; Sun et al., 2021) or applying KG prediction tasks, e.g., link prediction, as additional supervision (Yasunaga et al., 2022). However, when scaling the pre-training data to a web-scale corpus with trillion words, it is intractable to find or create KGs with approximate scale. More importantly, although these methods directly compress KG knowledge into LLM's parameters via supervision, they do not mitigate the fundamental limits of LLMs in flexibility, reliability, and transparency.

For (2), the early efforts were centered around fusing KG triples into the inputs of LLMs via attention (Liu et al., 2020; Sun et al., 2020) or attaching graph encoders to LLM encoders to process KG inputs (Wang et al., 2019). The follow-ups further adopted graph neural networks in parallel to LLMs for joint reasoning (Yasunaga et al., 2021) and added interactions between text tokens and KG entities in the intermediate layers of LLMs (Zhang et al., 2022; Yao et al., 2023b). Witnessing the recent success of pre-trained LLMs, the research paradigm is shifting to prompting fixed pre-trained LLMs with graphical inputs. Some of this line of research includes prompting LLMs for KG entity linking prediction (Choudhary and Reddy, 2023; Sun et al., 2023), graph mining (Guo et al., 2023). While these approaches permit LLMs to comprehend graph inputs, they take graphs as a text and lose the graph index information. Some more usually focus on targets KG tasks, like KG question answering (Baek et al., 2023). Most importantly, these methods often rely heavily on the factual correctness of the KG and ignore the situation where the KG does not match the question.

3 Method

We show the framework of `MindMap` in Figure 5, which comprises three main components:

1. **Evidence graph mining**: We begin by identifying the set of entities $\mathcal{V}_q$ from the raw input and query the source KG $\mathcal{G}$ to build multiple *evidence sub-graphs* $\mathcal{G}_q$.

2. **Evidence graph aggregation**: Next, LLMs are prompted to comprehend and aggregate the retrieved evidence sub-graphs to build the *reasoning graphs* $\mathcal{G}_m$.

3. **LLM reasoning on the mind map**: Last, we prompt LLMs to consolidate the built reasoning graph and their implicit knowledge to generate the answer and build a *mind map* explaining the reasoning process.

3.1 Step I: Evidence Graph Mining

Discovering the relevant evidence sub-graphs $\mathcal{G}_q$ from the external KG breaks down into two main stages.

3.1.1 Entity Recognition

We first use LLM to identify key entities from the question query Q. Specifically, we use a prompt that consists of three parts: the question to be analyzed, the template phrase "The extra entities are", and two examples. The full prompt is given in Table 9 of Appendix D. We then apply BERT similarity to match entities and keywords. Specifically, we encode all the keyword entities M extracted by LLM and all the entities $\mathcal{G}$ from the external knowledge graph into dense embeddings H_M and $H_\mathcal{G}$ respectively, and then compute the cosine similarity matrix between them. For each keyword,

```
SystemMessage(content= """You are an excellent AI doctor, and you can diagnose diseases and recommend medications based on the symptoms in the conversation."""),
HumanMessage(content="""Patient input:"""+ Question),
AIMessage(content=f """Combine the knowledge you already have, you have some extra medical knowledge information in the following:\n\n ### """+ path_reasoning_graph + """\n\n###""" + neighbor_reasoning_path),
HumanMessage(content="""What disease does the patient have? What tests should patient take to confirm the diagnosis? What recommened medications can cure the disease? Think step by step.\n\n\n
Output1: The answer includes disease and tests and recommened medications.\n\n
Output2: Show me inference process as a string about extract what knowledge from which Path-based Evidence or Neighor-based Evidence, and in the end infer what result. \n Transport the inference process into the following format:\n Path-based Evidence number('entity name'->'relation name'->...)->Path-based Evidence number('entity name'->'relation name'->...)->Neighbor-based Evidence number('entity name'->'relation name'->...)->Neighbor-based Evidence number('entity name'->'relation name'->...)->result number('entity name')->Path-based Evidence number('entity name'->'relation name'->...)->Neighbor-based Evidence number('entity name'->'relation name'->...). \n\n
Output3: Draw a decision tree. The entity or relation in single quotes in the inference process is added as a node with the source of evidence, which is followed by the entity in parenthesis.\n\n
There is a sample:\n ... """)
```

Figure 3: The prompt template for final input to LLM. Its input is the question and reasoning graphs.

we obtain the entity set $\mathcal{V}_q$ with the highest similarity scores, which we use to build the evidence subgraph in the next step.

3.1.2 Evidence Sub-graphs Exploration

We define the extra source knowledge graph by $\mathcal{G} = \{\langle u, r, o\rangle \,|u \in \psi, r \in \varphi, o \in \mathcal{L}\}$, where ψ_q, φ_q, and $\mathcal{L}_q$ represent the entity set, relation set, and textual set, respectively. The objective of this stage is to build the evidence sub-graphs $\mathcal{G}_q = \{\mathcal{G}_q^{path}, \mathcal{G}_q^{nei}\}$ based on the extracted entities $\mathcal{V}_q$. An evidence sub-graph is defined by $\mathcal{G}_q^* = \left(\mathcal{N}_q^*, \mathcal{E}_q{}^*, \psi_q^*, \varphi_q^*, \mathcal{L}_q^*\right)$, where $\mathcal{N}_q^*$ is the node set, $\mathcal{E}_q^*$ is the edge set where each edge $e = \langle n, n'\rangle$, $n, n' \in \mathcal{N}_q^*$.

As shown in Figure 5, we use two approaches to build the evidence sub-graph set $\mathcal{G}_q$ from the source knowledge graph. **(1)** Path-based exploration traces intermediary paths within $\mathcal{G}$ to connect important entities from the query. We form path segments by exploring connected nodes from a chosen node in $\mathcal{V}_q^0$ for at most k hops. The process continues until all segments are connected, creating a set of sub-graphs stored in $\mathcal{G}_q^{\text{path}}$. **(2)** Neighbor-based exploration adds related knowledge by expanding each node n in $\mathcal{N}_q$ by 1-hop to its neighbors, adding triples $\{(n, e, n')\}$ to $\mathcal{G}_q^{\text{nei}}$. This approach incorporates additional query-related evidence into $\mathcal{G}_q$. After exploration, we update $\mathcal{V}_q$ with newly added intermediate nodes from bridging pathways. To manage information overhead and maintain diversity, we prune $\mathcal{G}_q^{\text{path}}$ and $\mathcal{G}_q^{\text{nei}}$ by clustering and sampling sub-graphs based on their head entities. These pruning steps result in the final evidence graph $\mathcal{G}_q$, optimizing information while preserving diversity. Specific details are shown in Appendix E. We show the hallucination influence of results using path-based exploration and neighbor-based exploration components in the experiment part.

3.2 Step II: Evidence Graph Aggregation

In this phase, LLM is instructed to consolidate the diverse evidence sub-graphs $\mathcal{G}_q^*$ into a unified *reasoning graph* $\mathcal{G}_m$. This reasoning graph $\mathcal{G}_m$, upon completion, serves as an external augmented graph input for Step III, providing a holistic perspective on all evidence sub-graphs to enhance the output generation process.

To generate the final additional knowledge subgraph input, we first extracted at least k path-based and k neighbor-based evidence subgraphs from the previous part, each representing a possible connection between the query entities. Then, we formatted each subgraph as an entity chain, such as *"(Fatigue, Nausea) - IsSymptomOf - LiverProblem"*, and assigned a sequence number, such as P_1, P_2, N_1 , N_2. Next, we prompted LLM to convert each entity chain into a natural language description, using a template that can be found in Table 10 of Appendix D, and defined them as reasoning graph $\mathcal{G}_m$. This design had two benefits: **(a)** It simplified the subgraphs into a concise and consistent format that captured the essential information. **(b)** It leveraged LLM's natural language understanding and generation abilities to unify semantically similar entities and resolve potential ambiguities.

3.3 Step III: LLM Reasoning with Mind Map

In this step, LLMs are prompted with two reasoning graphs $\mathcal{G}_m^{\text{path}}$ and $\mathcal{G}_m^{\text{nei}}$ in Step II to produce the final outputs.

3.3.1 Prompting for Graph Reasoning

To generate a mind map and find final results, we provide LLMs with a prompt that has five components: a system instruction, a question, evidence graphs $\mathcal{G}_m$, a graph-of-thought instruction, and exemplars. The graph-of-thought instruction uses the Langchain technique to guide LLMs to comprehend and enhance the input, build their own mind map for reasoning, and index the knowledge sources of the mind map. The prompt template is detailed in Figure 3. The final answers consist of a summary answer, an inference process, and a mind map that shows the graph reasoning pathways. The entities in the mind map are from the evidence graphs $\mathcal{G}_m$ and LLM's own retrieval enhancement, as shown in the right red box in Figure 5 in Appendix D.

Table 1: The statistics of the used datasets.

Dataset	GenMedGPT-5k	CMCQA	ExplainCPE
Domain	English Clinical Q&A	Chinese Long Dialogue	5-way Choice Question
Multi-task	Disease, Drug, Test	Disease, Drug, Test, Food	Option, Explanation
KG dataset	EMCKG	CMCKG	**CMCKG**
Question	714	468	400
Node	1122	62282	62282
Triple	5802	506490	506490
Relationship	6	12	12

Table 2: The BERTScore and GPT4 ranking of all methods for **GenMedGPT-5k**.

	BERT Score			GPT4 Ranking	Hallucination
	Precision	Recall	F1 Score	(Average)	Quantify
MindMap	**0.7936**	0.7977	**0.7954**	**1.8725**	**0.6070**
GPT-3.5	0.7612	0.8003	0.7800	4.8571	0.5563
Tree-of-thought(TOT)	0.7202	0.7949	0.7554	-	0.5483
GPT4	0.7689	0.7893	0.7786	4.1764	0.5577
BM25 Retriever	0.7693	0.7981	0.7831	3.5546	0.5834
Embedding Retriever	0.7690	0.8038	0.7857	3.1232	0.5886
KG Retriever	0.7717	0.8030	0.7868	3.4159	0.5871

3.3.2 Synergistic Inference with LLM and KG Knowledge

We find that previous retrieval-augmented LLMs tend to rephrase the retrieved facts without exploiting the knowledge of LLM itself. However, MindMap enables LLM to synergistically infer from both the retrieved evidence graphs and its own knowledge. We attribute this ability to three aspects: **(1)** *Language Understanding*, as LLM can comprehend and extract the knowledge from $\mathcal{G}_m$ and the query in natural language. We show an example of entity disambiguation in Section 4.6.3 where nodes like 'vaginitis' and 'atrophic vaginitis' have the same meaning but appear in different evidence sub-graphs (Figure 7 in Appendix F). **(2)** *Knowledge Reasoning*, as LLM can produce the final answer based on the mind map constructed from $\mathcal{G}_m$. We show the final result of a CMCQA question in Section 4.6.4 (Figure 8 in Appendix F). **(3)** *Knowledge Enhancement*, as LLM can leverage its implicit knowledge to expand, connect, and improve the information relevant to the query. This ability is especially valuable when the external knowledge input is inaccurate. We illustrate an example of wrong external knowledge in Section 4.6.2 and 4.6.5, where the question in Figure 6 (Appendix F) contains misleading symptom facts, such as 'jaundice in my eyes', that lead baseline models to retrieve irrelevant knowledge linked to 'eye'.

4 Experiments

We evaluate our method for a suite of question & answering tasks that require sophisticated reasoning and domain knowledge and compare it with retrieval-based baselines.

4.1 Experimental Setup

We evaluate the utilization of external knowledge graphs by MindMap in complex question-answering tasks across three medical Q&A datasets: *GenMedGPT-5k*, *CMCQA*, and *ExplainCPE*. These datasets cover patient-doctor dialogues, multi-round clinical dialogues, and multiple-choice questions from the Chinese National Licensed Pharmacist Examination, respectively. To support KG-enhanced methods, we construct two knowledge graphs (*EMCKG* and *CMCKG*) containing entities and relationships related to medical concepts. The *ExplainCPE* dataset utilizes *CMCKG* with knowledge mismatches to assess the impact of incorrect retrieval knowledge on model performance. We compare MindMap's ability to integrate implicit and explicit knowledge with various baselines, including vanilla GPT-3.5 and GPT-4, as well as the tree-of-thought method (TOT) (Yao et al., 2023a), which uses a tree structure for reasoning. Additionally, we consider three retrieval-augmented baselines: *BM25 retriever*, *Text Embedding retriever*, and *KG retriever*, see instruction details in Appendix D. These baselines leverage different methods and sources for evidence retrieval, with gpt-3.5-turbo-0613 as the backbone for all retrieval-based methods. Detailed descriptions of these baselines are provided in Appendix C.

4.2 Medical Question Answering

We used GenMedGPT-5K to test how LLMs deal with question-answering in the medical domain, where LLMs need to answer with disease diagnosis, drug recommendation, and test recommendation.

4.2.1 Evaluation Metrics

We used two metrics, *BERTScore*(Zhang et al., 2019a) and *GPT-4 Rating*, for quantitative evaluation. *BERTScore* measures semantic similarity between the generated and reference answers. GPT-4 was employed to (1) rank answer quality against ground truth and (2) compare pairs of answers on four criteria: *response diversity and integrity*, *overall factual correctness*, *correctness of disease diagnosis*, and *correctness of drug recommendation*. In addition, we introduce a new metric for hallucination quantification, which estimates the degree of deviation from the facts in the generated answers (Liang et al., 2023). To compute this metric, we

Table 3: The pair-wise comparison by GPT-4 on the winning rate of MindMap v.s. baselines on diversity & integrity score (%), fact total match score (%), and disease diagnosis (%), on **GenMedGPT-5k**.

MindMap vs Baseline	GPT-3.5			BM25 Retriever			Embedding Retriver			KG Reriever			GPT-4			TOT		
Metries	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose
Diversity & integrity	100	-	-	100	-	-	100	-	-	100	-	-	100	-	-	-	-	-
Total factualness	**80.11**	-	19.89	66.67	-	33.33	76.05	-	23.95	73.53	-	26.47	75.77	-	24.23	78.5	-	21.5
Disease diagnosis	84.73	0.14	15.13	75.91	1.26	22.83	77.03	1.96	21.01	66.67	2.94	30.39	73.11	1.40	25.49	75	24.6	0.3
Drug recommendation	88	5	7	87	8	5	72	13	15	74	19	7	83	8	9	87	5	8
Average	**88.21**	1.285	10.505	**82.395**	2.315	15.29	**81.27**	3.74	14.99	**78.55**	5.485	15.965	**82.97**	2.35	14.68	**80.17**	14.8	9.93

Table 4: The BERTScore and GPT-4 ranking of all methods for **CMCQA** dataset.

	BERT Score			GPT-4 Ranking
	Precision	Recall	F1 Score	(Average)
MindMap	**0.9415**	0.9321	0.9367	**2.3**
GPT-3.5	0.9385	0.9361	0.9372	3.4
GPT-4	0.9355	0.9358	0.9356	3.6
BM25 Retriever	0.9365	0.9348	0.9356	3.7
Embedding Retriever	0.9357	0.9334	0.9345	5.4
KG Retriever	0.9318	0.9348	0.9332	2.3

first use the question-extra entities data generated by Step I and train a keyword extraction model (NER-MT5) based on mT5-large. Then, we input the outputs of MindMap, other baselines, and labels into the NER-MT5 model to obtain the lists of keywords for each answer. Finally, we concatenate the keywords with commas as ner-sentences, and calculate the tfidf similarity score between the ner-sentences of different outputs. A lower score indicates more hallucination in the answer.

4.2.2 Results

In Table 2, various methods are evaluated based on BERTScore, GPT-4 ranking scores, and hallucination quantification scores.

While BERTScore shows similar results among methods, MindMap exhibits a slight improvement, possibly due to the shared tone in medical responses. However, for medical questions, comprehensive domain knowledge is crucial, not well-captured by BERTScore. GPT-4 ranking scores and hallucination quantification reveal that MindMap significantly outperforms others, with an average GPT-4 ranking of 1.8725 and low hallucination scores. This underscores MindMap's ability to generate evidence-grounded, plausible, and accurate answers compared to baseline models like GPT-3.5 and GPT-4, which may produce incorrect responses due to reliance on implicit knowledge. Additionally, Table 3 demonstrates MindMap's consistent superiority over other methods, emphasizing the value of integrating external knowledge to mitigate LLM hallucinations and provide accurate answers.

4.3 Long Dialogue Question Answering

In our experiments on the CMCQA dataset, characterized by lengthy dialogues requiring complex reasoning, Table 4 showcases MindMap consistently ranking favorably compared to most baselines, albeit similar to KG Retriever. Additionally, in Table 5, MindMap consistently outperforms baselines in pairwise winning rates as judged by GPT-4. Despite a narrower performance gap compared to GenMedGPT-5K, attributed to the inadequacy of the knowledge graph (KG) in covering all necessary facts for CMCQA questions, MindMap still outshines all retrieval-based methods, including KG Retriever. This suggests previous retrieval-based approaches might overly rely on retrieved external knowledge, compromising the language model's (LLM) ability to grasp intricate logic and dialogue nuances using its implicit knowledge. Conversely, MindMap leverages both external and implicit knowledge in graph reasoning, yielding more accurate answers.

4.4 Generate with Mismatch Knowledge from KG

In addressing the robustness of MindMap concerning the factual correctness of KG, we leverage the identical KG dataset employed in the second dataset - *ExplainPE*. Consequently, the knowledge retrieved may tend to be redundant or devoid of accurate information. This aspect is particularly crucial since it mirrors a common scenario in production, where LLM often needs to generate answers by amalgamating both its implicit knowledge and the knowledge retrieved from external sources.

4.4.1 Evaluation Metrics

We evaluate all methods based on the accuracy of the generated choice and the quality of the explanations. For assessing explanation quality, we use BERTScore and GPT-4 ranking. We specifically

Table 5: The pair-wise comparison by GPT-4 on the winning rate of MindMap v.s. baselines on disease diagnosis and drug recommendation on CMCQA.

MindMap vs Baseline	GPT-3.5			BM25 Retriever			Embedding Retriver			KG Reriever			GPT-4		
Metrics	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose
Disease diagnosis	35.68	39.96	24.36	30.98	50.21	18.80	37.18	42.74	20.08	34.40	45.51	20.09	27.99	47.22	24.79
Drug recommendation	47.32	30.62	22.06	47.11	29.34	23.55	44.97	32.12	22.91	44.33	31.26	24.41	44.11	29.76	26.12
Average	41.5	35.29	23.21	39.045	39.775	21.175	41.075	37.43	21.495	39.365	38.385	22.25	36.05	38.49	25.455

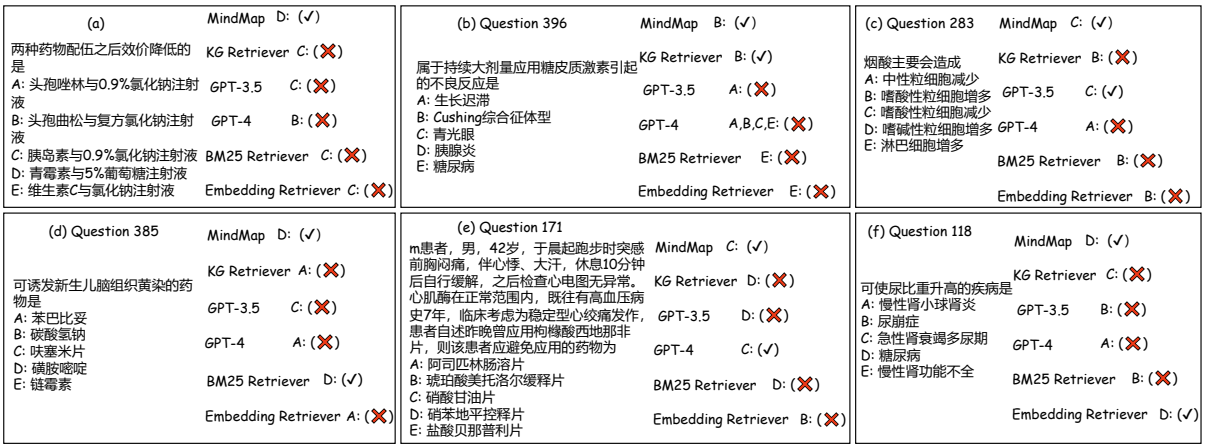

Figure 4: **Case examples of multi-choice in ExplainCPE**, comparing predictions by Baselines and MindMap.

Table 6: The accuracy scores for **ExplainCPE**. We calculate the rates of correct, wrong, and failed responses.

Method	Accuracy Rate(%)		
	Correct	Wrong	Failed
GPT-3.5	52.2	47.2	0.5
BM25 Retriever	50	44.2	5.7
Embedding Retriever	54.2	45.2	0.5
KG Retriever	42	44	14
GPT-4	72	27.7	0.2
MindMap	**61.7**	**37.7**	**0.5**
w/o prompt template $\mathbf{p}_1$	53.5	46	0.5

Table 7: Quantitative comparison with BERTScore and GPT-4 preference ranking between MindMap and baselines in ExplainCPE dataset.

	BERT Score			GPT-4 Ranking
	Precision	Recall	F1 Score	(Average)
MindMap	0.9335	0.9376	0.9354	**2.98**
GPT-3.5	0.9449	0.9487	0.9467	3.0425
GPT-4	0.9487	0.9529	0.9507	3.0075
BM25 Retriever	0.9413	0.9411	0.9411	3.6675
Embedding Retriever	0.9440	0.9459	0.9449	4.3175
KG Retriever	0.9354	0.9373	0.9362	3.985

instruct the GPT-4 rater to prioritize the correctness of the explanation over its helpfulness or integrity.

4.4.2 Results

In Table 6, our method (MindMap) demonstrates superior accuracy compared to various baselines, affirming its effectiveness over document retrieval prompting techniques. Interestingly, we observed that directly incorporating retrieved knowledge into prompts sometimes degrades answer quality, as seen with KG Retriever and BM25 Retriever performing worse than the vanilla GPT-3.5 model. This discrepancy arises from mismatched external knowledge, leading to misleading effects on the language model (LLM). The model tends to rely on retrieved knowledge, and when inaccurate, the LLM may generate errors. Ablation analysis on instruction prompts revealed that prompting the LLM to "combine with the knowledge you already have" ($\mathbf{p}_1$) improved performance by 8.2%. Moreover, Table 7 highlights MindMap's ability to generate rationales for answers, earning a ranking of 2.98 by GPT-4.

Table 8: The BERTScore and hallucination qualification of different component for GenMedGPT-5k.

	Tokens	BERT Score			Hallucination
	(Average)	Precision	Recall	F1 Score	Quantify
Path-only	1028	0.6310	0.7885	0.7002	0.3854
Neighbor-only	1236	0.6393	0.7930	0.7072	0.3894
MindMap	1431	**0.7938**	**0.7987**	**0.7960**	**0.5890**
Improved-path	+403	+0.1628	+0.0102	+0.0957	+0.2036
Improved-neigh	+195	+0.1545	+0.0057	+0.0888	+0.1996

4.5 Ablation Study

In our study, we compared our method (MindMap) with two variants: Neighbor-only and Path-only. Neighbor-only focuses on neighbor-based evidence exploration, while Path-only concentrates on path-based evidence exploration. Despite using additional tokens, MindMap showed significant improvements in hallucination quantification compared to both Neighbor-only and Path-only methods. This highlights the importance of combining both path-based and neighbor-based approaches to reduce hallucinations. Notably, the neighbor-based method proved more effective in enhancing factual accuracy compared to the path-based method. For tasks involving medical inquiries, path-based methods are better at finding relevant external information, though they struggle with multi-hop answers such as medication and test recommendations.

4.6 In-depth Analysis

We further conducted an in-depth analysis of the cases by MindMap, focusing on the discussion of the following aspects.

4.6.1 How does MindMap perform without correct KG knowledge?

In Figure 4(c) (Appendix F), when faced with a question where GPT-3.5 is accurate but KG Retriever errs, MindMap achieves an accuracy rate of 55%. We attribute the low accuracy of the KG Retriever to its inability to retrieve the necessary knowledge for problem-solving. MindMap effectively addresses such instances by leveraging the LLM inherent knowledge, identifying pertinent external explicit knowledge, and seamlessly integrating it into a unified graph structure.

4.6.2 How robust is MindMap to unmatched fact queries?

The question in Figure 6 (Appendix F) contains misleading symptom facts, such as *'jaundice in my eyes'* leading baseline models to retrieve irrelevant knowledge linked to *'eye'*. This results in failure to identify the correct disease, with recommended drugs and tests unrelated to liver disease. In contrast, our model MindMap accurately identifies *cirrhosis'* and recommends the relevant *'blood test'* showcasing its robustness.

4.6.3 How does MindMap aggregate evidence graphs considering entity semantics?

In Figure 7 of Appendix F, nodes like *'vaginitis'* and *'atrophic vaginitis'* are present in different evidence sub-graphs but share a semantic identity. MindMap allows LLMs to disambiguate and merge these diverse evidence graphs for more effective reasoning. The resulting mind maps also map entities back to the input evidence graphs. Additionally, Figure 7 illustrates the GPT-4 rater's preference for total factual correctness and disease diagnosis factual correctness across methods. Notably, MindMap is highlighted for providing more specific disease diagnosis results compared to the baseline, which offers vague mentions and lacks treatment options. In terms of disease diagnosis factual correctness, the GPT-4 rater observes that MindMap aligns better with the ground truth.

4.6.4 How does MindMap visualize the inference process and evidence sources?

Figure 8 in Appendix F presents a comprehensive response to a CMCQA question. It includes a summary, an inference process, and a mind map. The summary extracts the accurate result from the mind map, while the inference process displays multiple reasoning chains from the entities on the evidence graph $\mathcal{G}_m$. The mind map combines all the inference chains into a reasoning graph, providing an intuitive understanding of knowledge connections in each step and the sources of evidence sub-graphs.

4.6.5 How does MindMap leverage LLM knowledge for various tasks?

Figure 4 in Appendix F illustrates MindMap's performance on diverse question types. For drug-related questions (a) and (d), which demand in-depth knowledge, MindMap outperforms other methods. Disease-related questions (b) and (f) show comparable results between retrieval methods and MindMap, indicating that incorporating external knowledge mitigates errors in language model outputs. Notably, for general knowledge questions (c), LLMs like GPT-3.5 perform better, while retrieval methods lag. This suggests that retrieval methods may overlook the knowledge embedded in LLMs. Conversely, MindMap performs as well as GPT-3.5 in handling general knowledge questions, highlighting its effectiveness in synergizing LLM and KG knowledge for adaptable inference across datasets with varying KG fact accuracies.

5 Conclusion

This paper introduced knowledge graph (KG) prompting that 1) endows LLMs with the capability of comprehending KG inputs and 2) facilitates LLMs inferring with a combined implicit knowledge and the retrieved external knowledge. We then investigate eliciting the mind map, where LLMs perform the reasoning and generate the answers with rationales represented in graphs. Through extensive experiments on three question & answering datasets, we demonstrated that our approach, `MindMap`, achieves remarkable empirical gains over vanilla LLMs and retrieval-augmented generation methods, and is robust to mismatched retrieval knowledge. We envision this work opens the door to fulfilling reliable and transparent LLM inference in production.

References

Rohaid Ali, Oliver Y Tang, Ian D Connolly, Jared S Fridley, John H Shin, Patricia L Zadnik Sullivan, Deus Cielo, Adetokunbo A Oyelese, Curtis E Doberstein, Albert E Telfeian, et al. 2022. Performance of chatgpt, gpt-4, and google bard on a neurosurgery oral boards preparation question bank. *Neurosurgery*, pages 10–1227.

Samy Ateia and Udo Kruschwitz. 2023. Is chatgpt a biomedical expert?–exploring the zero-shot performance of current gpt models in biomedical tasks. *arXiv preprint arXiv:2306.16108*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290*.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*.

Nurendra Choudhary and Chandan K Reddy. 2023. Complex logical reasoning over knowledge graphs using large language models. *arXiv preprint arXiv:2305.01157*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Falcon Z. Dai. 2020. Word2vec conjecture and a limitative result.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable ai for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. GPT4Graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 792–802.

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

Xun Liang, Shichao Song, Simin Niu, Zhiyu Li, Feiyu Xiong, Bo Tang, Zhaohui Wy, Dawei He, Peng Cheng, Zhonghao Wang, and Haiying Deng. 2023. Uhgeval: Benchmarking the hallucination of chinese large language models via unconstrained generation.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2022. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Anil Sharma and Suresh Kumar. 2023. Ontology-based semantic retrieval of documents using word2vec model. *Data & Knowledge Engineering*, 144:102110.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, pages 1–9.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-Graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuan-Jing Huang, and Zheng Zhang. 2020. CoLAKE: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3660–3670.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

Cunxiang Wang, Sirui Cheng, Zhikun Xu, Bowen Ding, Yidong Wang, and Yue Zhang. 2023. Evaluating open question answering evaluation. *arXiv preprint arXiv:2305.12421*.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7208–7215.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Yao Yao, Zuchao Li, and Hai Zhao. 2023b. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019a. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning,

and Jure Leskovec. 2022. GreaseLM: Graph reasoning enhanced language models. In *International Conference on Learning Representations*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019b. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

A Construction of Datasets

- **GenMedGPT-5k** is a 5K generated dialogue between patients and GPT-3.5 grounded on a disease database[1]. The question describes the symptoms of the patient during the consultation, which comes from the iCliniq database. Based on the database, the generated answers cover the diagnosis, symptoms, recommended treatments, and medical tests. We sampled 714 dialogues as the test set.

- **CMCQA** contains multi-round dialogues between patients and doctors in Chinese. It covers materials from 45 clinical departments such as andrology, gynecology, and obstetrics and gynecology. We simplified the setup by merging the patient's questions and the clinician's answers to build the one-round Q&A. We sampled 468 from all to build the test set.

- **ExplainCPE** is a 5-way choice question dataset from the Chinese National Licensed Pharmacist Examination. Answering the questions requires a series of capabilities, including logical reasoning, drug knowledge, scenario analysis, mathematical calculation, disease knowledge, and general knowledge. The answers include the correct options and the explanations. We extracted 400 samples related to disease diagnosis and treatment recommendations from the original dataset for testing.

B Implementation of Knowledge Graph

- **EMCKG** We utilized a disease database[2] to build the KG from scratch to support the knowledge source for the inference on GenMedGPT-5k. This database encompasses a diverse set of diseases and the corresponding symptoms, medical tests, treatments, etc. The entities in the EMCKG include disease, symptom, drug recommendation, and test recommendation. The relationships in the EMCKG include *'possible_disease'*, *'need_medical_test'*, *'need_medication'*, *'has_symptom'*, *'can_check_disease'*, *'possible_cure_disease'*. In total, the yielded KG contains of 1122 nodes and 5802 triples.

- **CMCKG** We established a KG based on *QASystemOnMedicalKG*[3] to support KG-augmented inference on CMCQA and ExplainCPE. The CMCKG includes various entities such as disease, symptom, syndrome, recommendation drugs, recommendation tests, recommendation food, and forbidden food. The relationships in the CMCKG include *'has_symptom'*, *'possible_disease'*, *'need_medical_test'*, *'has_syndrome'*, *'need_recipe'*, *'possible_cure_disease'*, *'recipe__is_good_for_disease'*, *'food__is_good_for_disease'*, *'food__is_bad_for_disease'*, *'need_medication'*, *'need_food'*, and *'forbid_food'*. In total, the KG contains 62282 nodes, 12 relationships, and 506490 triples.

C Implementation of Baselines

- **GPT-3.5 & GPT-4** We evaluate the performance of the recent dominant LLM models as two baselines, using *gpt-3.5-turbo* (Wang et al., 2023; Ateia and Kruschwitz, 2023) and *gpt-4*[4] (Ali et al., 2022; Guo et al., 2023) API respectively.

- **BM25 document retriever + GPT-3.5** We compare with existing BM25 document retriever methods (Roberts et al., 2020; Peng et al., 2023), which use BM25 retrieval scores (Robertson et al., 2009) as logits when calculating $p(z|x)$. For fair comparisons, we use the same KG database as our method to generate different document files. Specifically, we use the GPT-3.5 API to convert all knowledge data centered on one disease into natural language text as the content of a document. For GenMedGPT-5k, we make 99 documents based on English medical KG $\mathcal{G}_{English}$. For CMCQA and ExplainCPE, we make 8808 documents based on Chinese medical KG $\mathcal{G}_{Chinese}$. For each question query, we retrieve the top k gold document contexts based on bm25 scores.

- **Text embedding document retrieval + GPT-3.5** Same as BM25 document retriever methods, text embedding document retrieval methods (Sharma and Kumar, 2023; Lewis et al., 2020) retrieve the top k documents for each question query. The difference is that in this method we train a word2vec

[1] https://github.com/Kent0n-Li/ChatDoctor

[2] https://github.com/Kent0n-Li/ChatDoctor/blob/main/format_dataset.csv

[3] https://github.com/liuhuanyong/QASystemOnMedicalKG/blob/master/data/medical.json

[4] https://openai.com/gpt-4

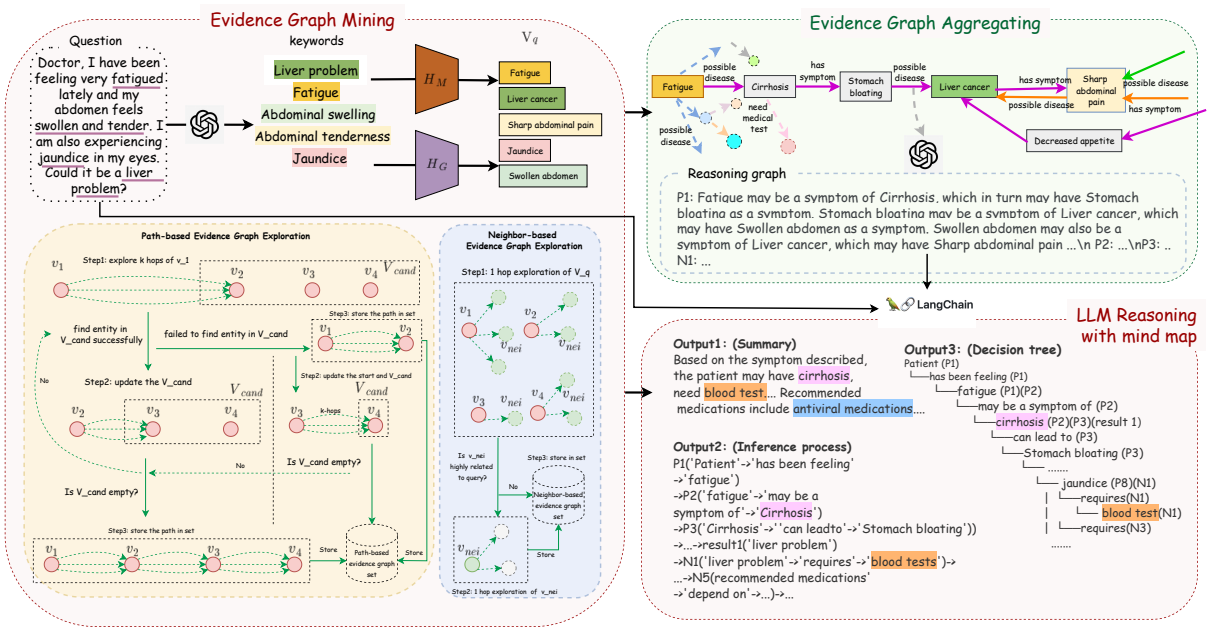

Figure 5: An overview of the architecture of our proposed MindMap. The left part illustrates the components of *evidence graph mining*, while the right part shows the *evidence graph aggregation* and *LLM reasoning with mind map*.

embedding (Dai, 2020) on the document corpus as the evidence source for document ranking.

- **KG retrieval + GPT-3.5** We compare with existing KG retrieval methods (Jia et al., 2021; Sun et al., 2023), which aim to find the shortest KG path between every pair of question entities. The final prompt is then retrieved from KG to guide GPT-3.5 model to answer questions. For fair comparisons, we use the same preliminary process as our method to recognize the entities in question query. The key differences between MindMap and these are that they do not think on multiple evidence KG sub-graphs with multi-thought in LLM, and without backtracking evidence sources.

- **Tree-of-thought (TOT)** We compare TOT as a typical Chain-of-thought (Wei et al., 2022b) baseline with MindMap. TOT is a method that uses a tree structure to solve complex problems(Yao et al., 2023a). By extending one inference path into multiple inference paths, the model can synthesize the results of multiple inference paths to obtain the final conclusion.

D Prompt Engine

- **The instructions of MindMap components.** Table 9 shows the instruction of Step I: entity recognition, which aims to identify and label medical entities in the user query. Table 10 shows the templates of Step II (Evidence Graph Aggregation), which generates natural language sentences from the evidence graph nodes and edges.

- **The instructions of baseline methods:** Table 11 shows the prompt template of two document retrieval methods (BM25 Retrieval and Embedding Retrieval). The input is the question and the most related document context.

- **The instructions of evaluation:** Figure 3 presents the final prompt used by MindMap for generating results and constructing a mind map. The prompt consists of a system message acknowledging the AI's expertise as a doctor, a user message representing the patient's input, and an AI message incorporating knowledge obtained from an external KG. The Langchain technique is employed to create the prompt, which guides the generation of step-by-step solutions. The response consists of a summary answer to the query, the inference process, and a mind map. Table 12 illustrates an example of the pairwise ranking evaluation using the GPT-4 rater, which compares the quality of different responses based on various criteria.

E Evidence Subgraphs Exploration

We provide more details on the path-based and neighbor-based exploration methods as follows:

- **Path-based Evidence Graph set $\mathcal{G}_q^{\text{path}}$ Exploration** connects entities in $\mathcal{V}_q$ by tracing their intermediary pathways within $\mathcal{G}$: **(a)** Choose one node in $\mathcal{N}_q^0$ as the start node n_1. Place the remaining nodes in a candidate node set $\mathcal{N}_{cand}$. Explore at most k hops from n_1 to find the next node n_2, where $n_2 \in \mathcal{N}_{cand}$. If n_2 is successfully reached within k hops, update the start node as n_2 and remove n_2 from $\mathcal{N}_{cand}$. If n_2 cannot be found within k hops, connect the segments of paths obtained so far and store them in $\mathcal{G}_q^{\text{path}}$. Then choose another node $n_1{}'$ from $\mathcal{N}_{cand}$ as the new start node, and remove both n_1 and n_2 from $\mathcal{N}_{cand}$. **(b)** Check if $\mathcal{N}_{cand}$ is empty. If it is not empty, iterate step 1 to find the next segment of the path. If it is empty, connect all segments to build a set of sub-graphs and put them into $\mathcal{G}_q^{\text{path}}$.

- **Neighbor-based Evidence Graph set $\mathcal{G}_q$ Exploration** aims to incorporate more query-related evidence into $\boldsymbol{G}_q$. It has two steps: **(a)** Expand for each node $n \in \mathcal{V}_q$ by 1-hop to their neighbors $\{n\prime\}$ to add triples $\{(n, e, n')\}$ to $\mathcal{G}_q^{\text{nei}}$. **(b)** For each v', check if it is semantically related to the question. If so, further expand the 1-hop neighbors of n', adding triples (n_{nei}, e', n') to $\mathcal{G}_q^{\text{nei}}$.

F In-depth Analysis

We select four examples for in-depth analysis, as shown in Figure 6, 7, 8, and 4.

- Figure 6 presents an example from GenMedGPT-5k. It includes the question, reference response, the response generated by `MindMap`, responses from baselines, and the factual correctness preference determined by the GPT-4 rater. This example is used to discuss the robustness of `MindMap` in handling mismatched facts.

- Figure 7 illustrates another example from GenMedGPT-5k. It displays the question query, reference response, summary responses from both `MindMap` and baseline models, a mind map generated by `MindMap`, and specific preferences in terms of factual correctness and sub-task disease fact match determined by the GPT-4 rater. This example shows the ability of `MindMap` to aggregate evidence graphs.

- Figure 8 showcases an example from CMCQA. It includes the question query, a summary answer, the inference process, and the generated mind map by `MindMap`. This example provides insights into the visualization of the final output produced by `MindMap`.

- Figure 4 demonstrates an example from ExplainCPE. It consists of six questions categorized into three different question types and evaluates the accuracy of `MindMap` and baseline models. This example allows us to examine the performance of `MindMap` across various tasks.

G Pairwise Ranking Evaluation

For each pair of answers, as an example in Table 12, raters were asked to select the preferred response or indicate a tie along the following axes (with exact instruction text in quotes):

- **Diversity and integrity:** "According to the result in reference output, which output is better."

- **Total factual correctness:** "According to the facts of disease diagnosis and drug and tests recommendation in reference output, which output is better match."

- **Disease diagnosis:** "According to the disease diagnosis result in reference output, which output is better match."

- **Drug recommendation:** "According to the drug recommendation result in reference output, which output is better match."

Note that for the second dataset CMCQA, since the reference label is derived from the actual dialogue answer, it may not contain facts. When the GPT-4 rater performs pairwise ranking evaluation, it is very easy to judge it as a tie. Therefore, we add an additional instruction: "*If they are the same, output "2". Try to output "1" or "0"*", so as to force the rater to make a preference judgment.

H Limitations and Potential Risks

The integration of knowledge graphs (KGs) with large language models (LLMs), particularly in medical contexts, presents several potential challenges. One significant concern is the risk of replicating any existing biases or errors in the knowledge graphs. These graphs, often built from pre-existing data sources, might contain outdated or partial information, which could inadvertently influence the LLM's outputs. Another issue lies in the integration complexity between KGs and LLMs, which could lead to unexpected errors or logical inconsistencies, especially when addressing intricate or vague queries. This aspect is critically important in the medical field, where precision is paramount. Moreover, there's a possibility that the LLMs might become excessively dependent on the KGs, which could hinder their performance in scenarios where KGs are not accessible or are lacking in information. Additionally, the use of "mind maps" to trace the LLMs' reasoning paths, while innovative, raises questions about the models' interpretability. If these visual representations are complex or obscure, it may be difficult for users to understand how conclusions were reached, potentially diminishing trust in these advanced systems. In summary, while the merger of KGs with LLMs is a promising development, it is crucial to address these potential issues to ensure the responsible and efficacious application of this technology.

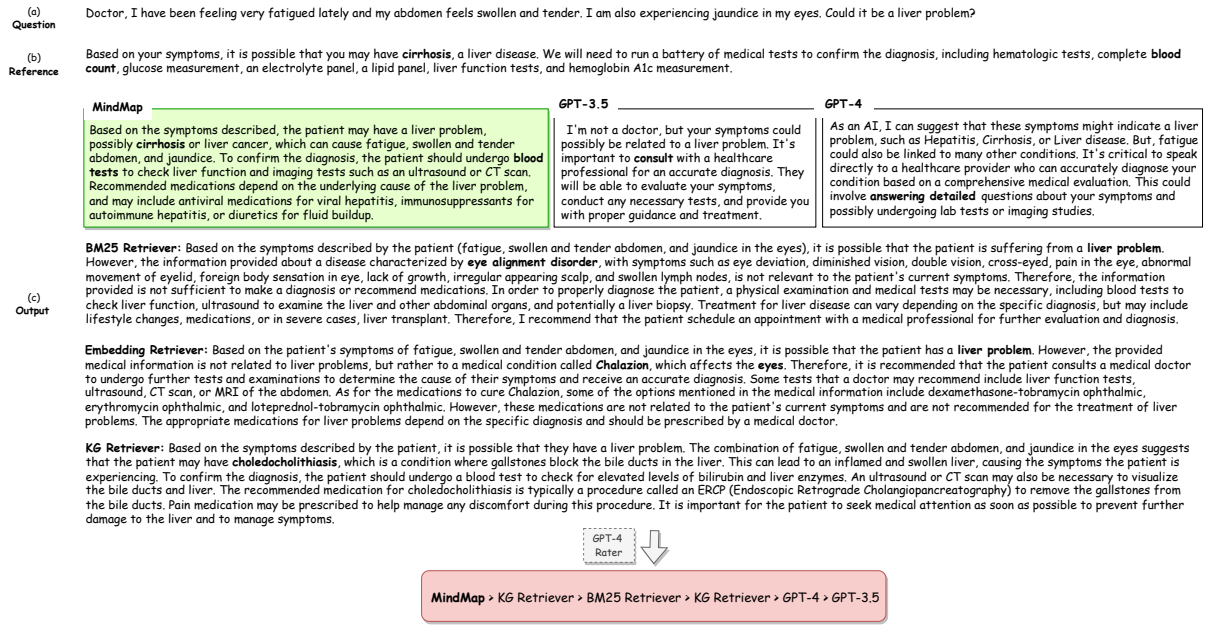

Figure 6: **A case compares `MindMap` and baselines with mismatched retrieved knowledge**, evaluated by the GPT factual correctness preference rater.

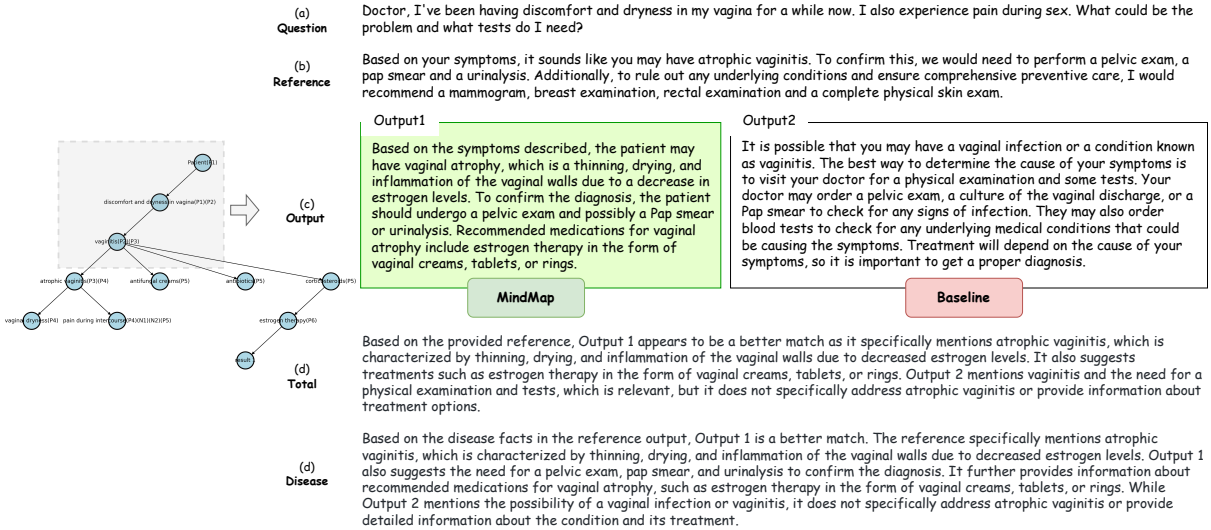

Figure 7: **Factually correctness evaluation in GenMedGPT-5k using GPT-4 preference ranking**: `MindMap` shows a strong ability in fact-matching subtasks of question-answering by generating a mind map.

```
template = """
There are some samples:
\n\n
### Instruction:\n'Learn to extract entities from the following
   medical questions.'\n\n### Input:\n
<CLS>Doctor, I have been having discomfort and dryness in my vagina
   for a while now. I also experience pain during sex. What could be
   the problem and what tests do I need?<SEP>The extracted entities
   are\n\n ### Output:
<CLS>Doctor, I have been having discomfort and dryness in my vagina
   for a while now. I also experience pain during sex. What could be
   the problem and what tests do I need?<SEP>The extracted entities
   are Vaginal pain, Vaginal dryness, Pain during intercourse<EOS>
\n\n
Instruction:\n'Learn to extract entities from the following medical
   answers.'\n\n### Input:\n
<CLS>Okay, based on your symptoms, we need to perform some diagnostic
    procedures to confirm the diagnosis. We may need to do a CAT scan
    of your head and an Influenzavirus antibody assay to rule out any
    other conditions. Additionally, we may need to evaluate you
   further and consider other respiratory therapy or physical therapy
    exercises to help you feel better.<SEP>The extracted entities are
   \n\n ### Output:
<CLS>Okay, based on your symptoms, we need to perform some diagnostic
    procedures to confirm the diagnosis. We may need to do a CAT scan
    of your head and an Influenzavirus antibody assay to rule out any
    other conditions. Additionally, we may need to evaluate you
   further and consider other respiratory therapy or physical therapy
    exercises to help you feel better.<SEP>The extracted entities are
    CAT scan of head (Head ct), Influenzavirus antibody assay,
   Physical therapy exercises; manipulation; and other procedures,
   Other respiratory therapy<EOS>
\n\n
Try to output:
### Instruction:\n'Learn to extract entities from the following
   medical questions.'\n\n### Input:\n
<CLS>{input}<SEP>The extracted entities are\n\n ### Output:
"""
```

Table 9: The prompt template of Entity Recognition. The input is the question.

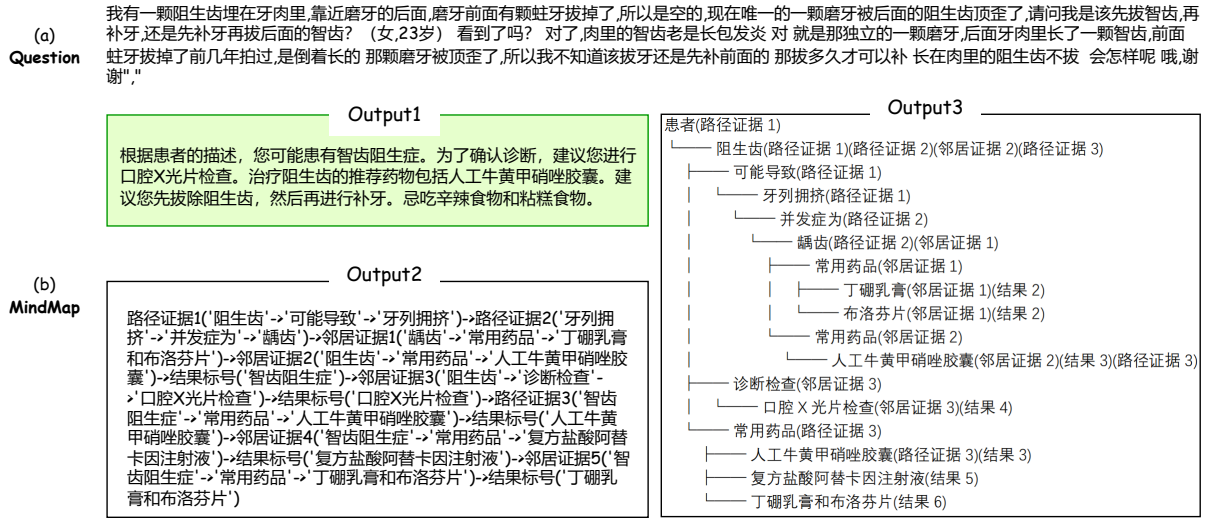

Figure 8: **An example to show the visualization of `MindMap`**. By generating mind maps, `MindMap` guides LLM to obtain the correct factual outputs for different subtasks.

```
template = """
    There are some knowledge graph path. They follow entity->
       relationship->entity format.
    \n\n
    {Path}
    \n\n
    Use the knowledge graph information. Try to convert them to
       natural language, respectively. Use single quotation marks for
        entity name and relation name. And name them as Path-based
       Evidence 1, Path-based Evidence 2,...\n\n


    Output:
    """


template = """
    There are some knowledge graph. They follow entity->relationship
       ->entity list format.
    \n\n
    {neighbor}
    \n\n
    Use the knowledge graph information. Try to convert them to
       natural language, respectively. Use single quotation marks for
        entity name and relation name. And name them as Neighbor-
       based Evidence 1, Neighbor-based Evidence 2,...\n\n


    Output:
    """
```

Table 10: The prompt templates of transfering path-based evidence subgraphs and neighbor-based evidence subgraphs to natural language.

```
template = """
    You are an excellent AI doctor, and you can diagnose diseases and
        recommend medications based on the symptoms in the
       conversation.\n\n
    Patient input:\n
    {question}
    \n\n
    You have some medical knowledge information in the following:
    {instruction}
    \n\n
    What disease does the patient have? What tests should patient
       take to confirm the diagnosis? What recommened medications can
        cure the disease?
    """
```

Table 11: The prompt templates of BM25 Retrieval and Embedding Retrieval. The input is the question and the most related document context.

```
def prompt_comparation(reference,output1,output2):
    template = """
    Reference: {reference}
    \n\n
    output1: {output1}
    \n\n
    output2: {output2}
    \n\n
    According to the facts of disease diagnosis and drug and tests
       recommendation in reference output, which output is better
       match. If the output1 is better match, output '1'. If the
       output2 is better match, output '0'. If they are same match,
       output '2'.
    """
    prompt = template.format(reference=reference, output1=output1,
       output2=output2)
    response = openai.ChatCompletion.create(
      model="gpt-4",
      messages=[
          {"role": "system", "content": """You are an excellent AI
             doctor."""},
          {"role": "user", "content": prompt}
      ]
    )
    response_of_comparation = response.choices[0].message.content
    return response_of_comparation
```

Table 12: The prompt template for GPT-4 rater to evaluate the factual correctness between our method and baselines, the reference is the answer or explanation label.