Graph RAG: Efficient and Powerful LLM Enhancement Systems

Shu Wang *School of Data Science*
The Chinese University of Hong Kong, Shenzhen
shuwang3@link.cuhk.edu.cn

***Abstract*—This paper proposes three novel Graph Retrieval-Augmented Generation (GraphRAG) methods to address limitations in traditional Retrieval-Augmented Generation (RAG) systems. Specifically, we propose three methods: the Agent-based method, the Document-based method, and the Global-information-based method. The Agent-based method integrates structured knowledge from knowledge graphs (KGs) and unstructured text, leveraging large language models (LLMs) as autonomous agents. The Document-based method, inspired by hippocampal indexing theory, enhances knowledge integration by simulating human memory processes. The Global-information-based method utilizes global community perspectives to improve performance on summary tasks. We evaluate these methods on Knowledge Graph Question Answering (KGQA), Document QA, and summarization tasks, demonstrating their effectiveness in overcoming the challenges of traditional RAG approaches.**

***Index Terms*—LLM, RAG, Graph RAG.**

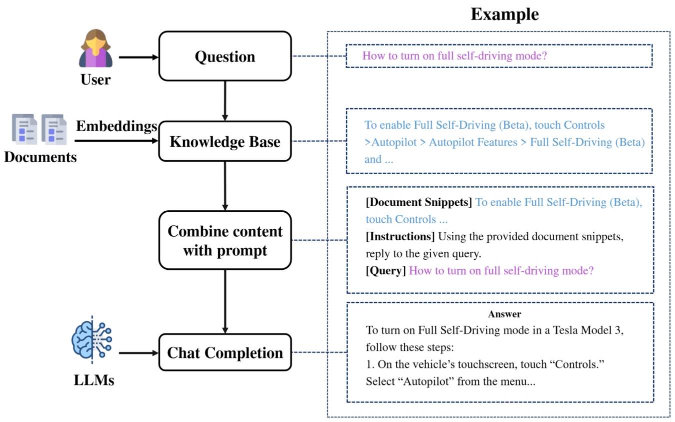

Example
User
Question
How to turn on full self-driving mode?
Documents
Embeddings
Knowledge Base
To enable Full Self-Driving (Beta), touch Controls >Autopilot > Autopilot Features > Full Self-Driving (Beta) and ...
Combine content with prompt
[Document Snippets] To enable Full Self-Driving (Beta), touch Controls ...
[Instructions] Using the provided document snippets, reply to the given query.
[Query] How to turn on full self-driving mode?
LLMs
Chat Completion
Answer
To turn on Full Self-Driving mode in a Tesla Model 3, follow these steps:
1. On the vehicle's touchscreen, touch "Controls." Select "Autopilot" from the menu...

Fig. 1. The workflow of Retrieval-Augmented Generation (RAG).

I. Introduction

Large Language Models (LLMs) have emerged as revolutionary tools that show impressive performance in many tasks [1]. With the growing size of LLMs, they can serve standalone as very effective knowledge stores, with facts encoded within their parameters [2]–[9] and models can be further improved with fine-tuning on downstream tasks [10]. Nevertheless, even a large model does not contain sufficient domain-specific knowledge for particular tasks, and the world continues to change, invalidating facts in the LLM. Moreover, the knowledge embedded within LLMs is encoded in their parameters, meaning that incorporating new knowledge requires further fine-tuning, which is both time-consuming and resource-intensive. Besides, LLMs have a tendency to generate hallucinations [11], [12], producing content that appears plausible but lacks factual accuracy or support [13].

A popular approach to equip LLM with domain knowledge and mitigate hallucination issues is Retrieval-Augmented Generation (RAG). RAG framework answers a question in four steps: the user proposes a query, the system retrieves the relevant content from private knowledge bases, combines it with the user query as context, and finally asks the LLM to generate an answer. This is illustrated in Fig. 1 with a simple example. VectorRAG (the traditional RAG technique that is based on vector databases) focuses on improving Natural Language Processing (NLP) tasks by retrieving relevant textual information to support the generation tasks. VectorRAG excels in situations where context from related textual documents is crucial for generating meaningful and coherent responses.

Although RAG has achieved impressive results and has been widely applied across various domains, it faces limitations in real-world scenarios: (1) Neglecting Relationships: In practice, textual content is not isolated but interconnected. Traditional RAG fails to capture significant structured relational knowledge that cannot be represented through semantic similarity alone. For instance, in a citation network where papers are linked by citation relationships, traditional RAG methods focus on finding the relevant papers based on the query but overlook important citation relationships between papers. (2) Redundant Information: RAG often recounts content in the form of textual snippets when concatenated as prompts. This makes the context become excessively lengthy, leading to the "lost in the middle" dilemma [14]. (3) Lacking Global Information: RAG can only retrieve a subset of documents, fails to grasp global information comprehensively, and hence struggles with tasks such as Query-Focused Summarization (QFS).

Graph Retrieval-Augmented Generation (GraphRAG) [15], [16] emerges as an innovative solution to address these challenges. Unlike traditional RAG, GraphRAG retrieves graph elements containing relational knowledge pertinent to a given query from a pre-constructed graph database, as depicted in Fig. 2. These elements may include nodes, triples, paths, or subgraphs, which are utilized to generate responses. GraphRAG considers the interconnections between texts, enabling a more accurate and comprehensive retrieval of relational information. Additionally, graph data, such as knowledge graphs, offer abstraction and summarization of textual data, thereby significantly shortening the length of the

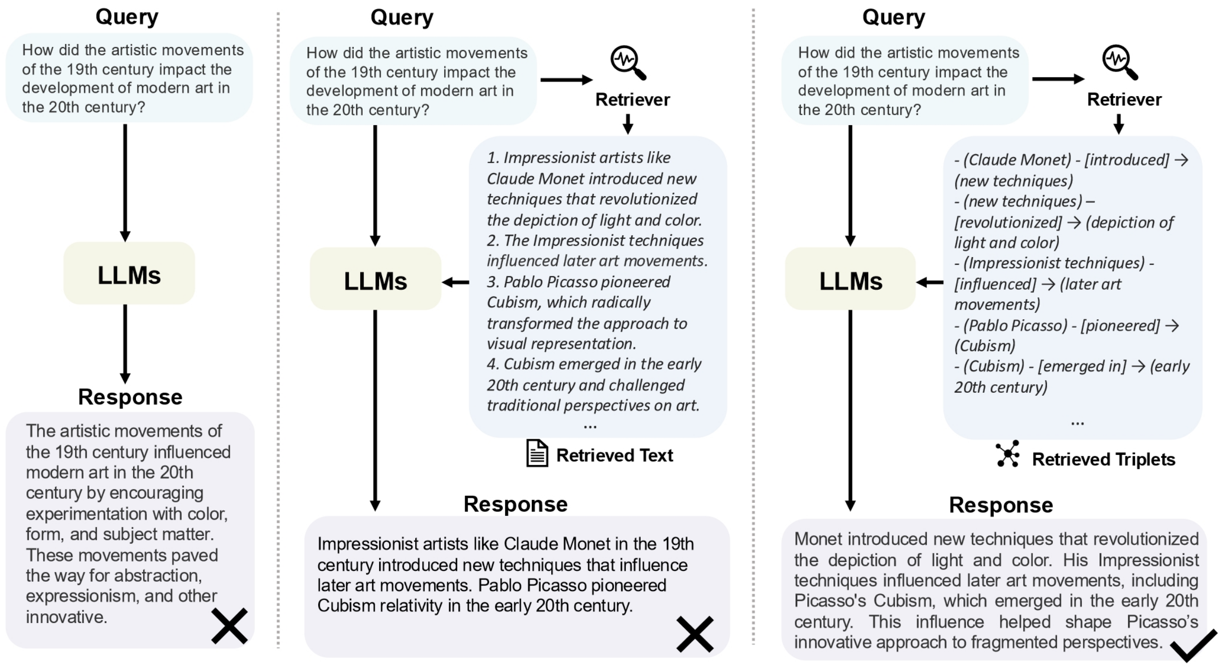

Query
How did the artistic movements of the 19th century impact the development of modern art in the 20th century?
LLMs
Response
The artistic movements of the 19th century influenced modern art in the 20th century by encouraging experimentation with color, form, and subject matter. These movements paved the way for abstraction, expressionism, and other innovative.
Query
How did the artistic movements of the 19th century impact the development of modern art in the 20th century?
Retriever
1. Impressionist artists like Claude Monet introduced new techniques that revolutionized the depiction of light and color. 2. The Impressionist techniques influenced later art movements. 3. Pablo Picasso pioneered Cubism, which radically transformed the approach to visual representation. 4. Cubism emerged in the early 20th century and challenged traditional perspectives on art. ...
Retrieved Text
LLMs
Response
Impressionist artists like Claude Monet in the 19th century introduced new techniques that influence later art movements. Pablo Picasso pioneered Cubism relativity in the early 20th century.
Query
How did the artistic movements of the 19th century impact the development of modern art in the 20th century?
Retriever
- (Claude Monet) - [introduced] → (new techniques)
- (new techniques) – [revolutionized] → (depiction of light and color)
- (Impressionist techniques) - [influenced] → (later art movements)
- (Pablo Picasso) - [pioneered] → (Cubism)
- (Cubism) - [emerged in] → (early 20th century)
...
Retrieved Triplets
LLMs
Response
Monet introduced new techniques that revolutionized the depiction of light and color. His Impressionist techniques influenced later art movements, including Picasso's Cubism, which emerged in the early 20th century. This influence helped shape Picasso's innovative approach to fragmented perspectives.

Fig. 2. Comparision between Direct LLM, RAG, and GraphRAG.

input text and mitigating concerns of verbosity. By retrieving subgraphs or graph communities, we can access comprehensive information to effectively address the QFS challenge by capturing the broader context and interconnections within the graph structure.

Based on the above motivation, in this paper, we propose three Graph RAG methods: the Agent-based method, the Document-based method, and the Global-information-based method. LLMs possess strong reasoning and analytical capabilities, making them effective as autonomous agents and offering promising opportunities to enhance and replicate human workflows. By leveraging the agent's abilities alongside knowledge graphs and document content, we introduce ToG-2, an agent-based method that integrates both knowledge graphs and documents. Furthermore, recognizing real-world entities' diverse connections, we propose a Document-based method inspired by the hippocampal indexing theory of human long-term memory. Finally, we introduce the Global-information-based method, which utilizes the global perspective provided by communities, yielding better results in summary tasks.

The contributions of this paper are outlined as follows:

- **Agent-based method.** Leveraging the capabilities of LLM agents, we propose a tight coupling hybrid (KG$\times$Text) RAG paradigm, which effectively integrates unstructured knowledge from texts with structured insights from KGs.
- **Document-based method.** We proposed a novel retrieval framework inspired by the hippocampal indexing theory of human long-term memory to enable deeper and more efficient knowledge integration over new experiences.
- **Global-information-based method.** By incorporating the global perspective of communities, our proposed Graph RAG enables RAG systems to address global questions that pertain to an entire text corpus.
- **Experiments.** We comprehensively tested the performance of the three proposed methods on KGQA, DocumentQA, and Summary tasks, providing an in-depth analysis of the results.

•**Organization.** The remainder of this paper is organized as follows: Section II provides a brief review of existing works. Section III introduces the preliminary concepts related to Graph RAG. In Section IV, we propose three types of Graph RAG methods: the agent-based method, the document-based method, and the global-information-based method. Section V presents a comprehensive evaluation of the proposed methods. Finally, Section VI concludes this paper.

II. RELATED WORK

• **Retrieval-Augmented Generation (RAG).** The concept of Retrieval-Augmented Generation, initially proposed by Lewis et al. [17], has gained increased attention for its ability to mitigate the issue of hallucination within LLMs and enhance trustworthiness and explainability [18]. Despite its success in language-related tasks, the application of retrieval-based approaches to general graph tasks remains largely unexplored. Most existing work focuses primarily on the knowledge graph [19]–[22].

• **LLMs & KGs.** Combining the strengths of language models and knowledge graphs has been an active research direction for many years, both for augmenting LLMs with a KG in different ways [23]–[25] or augmenting KGs by either distilling knowledge from an LLM's parametric knowledge [26], [27] or using them to parse text directly [28]–[30]. Chain-of-Knowledge [31] is a hybrid RAG method that retrieves knowledge from Wikipedia, Wikidata, and Wikitable to ground LLMs' outputs. HybridRAG [32] retrieves information from both vector databases and KGs, achieving superior reasoning performance compared to either text-based RAG or KG-based RAG methods alone. In an exceptionally comprehensive survey, Pan et al. [33] present a roadmap for this research direction and highlight the importance of work that synergizes these two important technologies [22], [34]–[36].

However, existing hybrid RAG approaches merely aggregate information retrieved from KGs and texts but do not improve the retrieval results on one knowledge source through the other. In this work, ToG-2 aims to tightly couple KG-based RAG and text-based RAG methods, enabling both in-depth context retrieval and precise graph retrieval to enhance complex reasoning performances of LLMs. Another of our works, HippoRAG, exemplifies the synergy between these two technologies. It combines the power of LLMs for knowledge graph construction with the strengths of structured knowledge and graph search, enhancing the augmentation of an LLM's capabilities.

• **Long Context as Long-Term Memory.** Context lengths for both open and closed-source LLMs have increased dramatically in the past year [37]–[41]. This scaling trend seems to indicate that future LLMs could perform long-term memory storage within massive context windows. However, the viability of this future remains largely uncertain, given the many engineering hurdles involved and the apparent limitations of long-context LLMs, even within current context lengths [42]–[44].

III. Preliminary

A. LLM and language model techniques

Definition 1 (Life cycle of LLMs): Typically, the life cycle of LLMs contains 4 steps: pre-training, supervised fine-tuning (SFT), reinforcement learning from human feedback (RLHF), and Inference.

1) In the pre-training phase, LLMs amass a vast amount of knowledge from an enormous volume of training data, which is then stored within their model parameters.
2) SFT (supervised fine-tuning) generally involves massive-task instruction-following data, aiming at learning how to interact with users
3) RLHF (reinforcement learning from human feedback) not only closes the gap between machine-generated content and human preference but also helps LLMs align with desired criteria or goals.
4) In the inference stage, the LLM performs various tasks based on the user's needs.

Most RAG systems are involved in the inference stage because the inference stage could be more cost-effective and controllable. The life cycle of LLMs and RAG is shown in Fig. 3

In the context of textual graphs, language models (LMs) are essential for encoding the text attributes associated with nodes and edges, thereby learning representations that capture their semantic meaning.

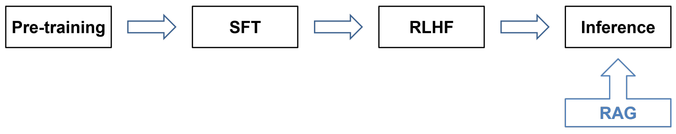


Fig. 3. The life cycle of LLMs and RAG

Definition 2 (Language Models for Text Encoding [45]): For a node n with text attributes $x_n \in D^{L_n}$, an LM encodes these attributes as:

$$z_n = LM(x_n) \in R^d \quad (1)$$

where z_n is the output of the LM, and d is the dimension of the output vector.

LLMs have introduced a new paradigm for task adaptation known as "pre-train, prompt, and predict," replacing the traditional "pre-train, fine-tune" paradigm. In this paradigm, the LLM is first pre-trained on a large corpus of text data to learn general language representations. Then, rather than fine-tuning the model on task-specific labeled data, the model is prompted with a textual prompt that specifies the task and context. Subsequently, the model generates the output directly based on the prompt and the input.

Definition 3 (Large Language Models and Prompt Tuning [45]): The LLM, parameterized by weights θ, takes a sequence of tokens X, and a prompt P as input, and generates a sequence of tokens $Y = y_1, y_2, ..., y_r$ as output. Formally, the probability distribution of the output sequence given the concatenated input sequence and prompt, i.e., $[P; X]$, is defined as

$$p_\theta(Y|[P;X]) = \prod_{i=1}^{r} p_\theta(y_i|y_{<i}, [P;X]) \quad (2)$$

Here, $y_{<i}$ represents the prefix of sequence y up to position $i-1$, and $p(y_i|y_{<i}, [P;X])$ represents the probability of generating token y_i given $y_{<i}$ and $[P;X]$.

The semantic relevance between data has been mapped into vector space by the language model. Therefore, we can use ANN (Approximate Nearest Neighbor) to find the closest neighbor data to a given query, which is essentially the concept of NNS (Nearest Neighbor Search) and ANN.

Definition 4 (Nearest Neighbor Search (NNS) [46]): Given a set P of n objects represented as points in a normed space l_p^d, preprocess P so as to efficiently answer queries by finding the point in P closest to a query point q.

The definition generalizes naturally to the case where we want to return $K > 1$ points. Specifically, in the K-Nearest Neighbors Search (K-NNS), we wish to return the K points in the database that are closest to the query point. The approximate version of the NNS problem is defined as follows:

Definition 5 (ϵ-Nearest Neighbor Search (ϵ-NNS)): Given a set P of objects in a normed space l_p^d, preprocess P so as to efficiently return a point $p \in P$ for any given query point q, such that $d(q,p) \leq (1+\epsilon)d(q,P)$, where $d(q,P)$ is the distance of q to its closest point in P.

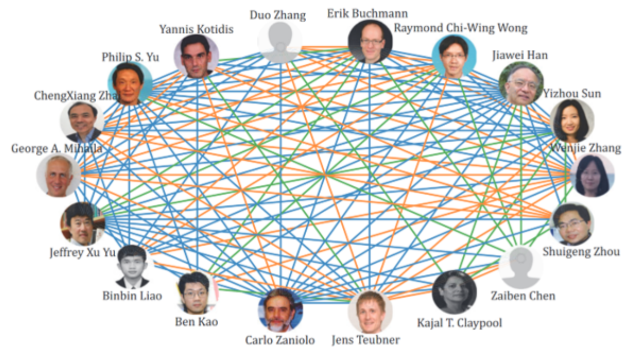


Fig. 4. An Example of a Scholarly Community

B. Graph

When it comes to Graph RAG, graph data and other relational data become key components of the RAG system. Next, we introduce TAG and community structures.

Definition 6 (Text-Attributed Graph (TAG) [45]): Formally, a Text-Attributed Graph (TAG) is a graph where nodes and edges possess textual attributes. Formally, it can be defined as $G = (V, E, \{x_n\}_{n \in V}, \{x_e\}_{e \in E})$, where V and E represent the sets of nodes and edges, respectively. Additionally, $x_n \in D^{L_n}$ and $x_e \in D^{L_e}$ denote sequential text associated with a node $n \in V$ or an edge $e \in E$, where D represents the vocabulary, and L_n and L_e signify the length of the text associated with the respective node or edge.

A TAG incorporates external data to enhance LLM with additional knowledge, forming the foundation for GraphRAG tasks. In the real world, knowledge graphs (e.g., WikiData [1], DBPeida [2], GeoNames [3]) and document graphs are examples of TAGs.

An important component of these graphs is the community.

Definition 7 (community [47]): Formally, a community is a group of vertices which are densely connected internally. We denote a community as $C = \{v_1, v_2, \ldots, v_n\}$, where the community consists of n vertices.

Because a community covers the many nodes of the graph, it can provide global information rather than a single node with its own attributes. For example, Fig.4 is a community consisting of many scholars in the fields of databases and data mining extracted from an academic network.

IV. Graph RAG methods

In this section, we introduce three types of Graph RAG methods: Agent-Based methods, Document-Based methods, and Global-Information-Based methods.

A. Agent-Based Methods

LLMs have demonstrated the ability to act as agents and search data in a human-like manner. Therefore, we propose an agent-based RAG method, ToG-2, to harness these capabilities for more effective data search and retrieval. Our ToG-2 framework is shown in Fig. 5.

The proposed method ToG-2 draws from the ToG approach [22] in multi-hop searches within KGs, starting from key entities identified in the query and exploring outward based on entities and relationships with a prompt-driven inferential process. ToG-2 combines the logical chain extensions based on triples with unstructured contextual knowledge of relevant entities by iteratively performing knowledge-guided context retrieval and context-enhance graph retrieval, thus more effectively integrating and utilizing heterogeneous external knowledge.

Specifically, ToG-2 begins by extracting entities from the given question as initial topic entities. It then performs an iterative process of graph retrieval, context retrieval, and LLM reasoning. At the start of each iteration, ToG-2 selectively explores entities neighboring the current topic entities on the KG, using the newly encountered entities to refine the retrieval scope, thereby enhancing both efficiency and accuracy. Next, ToG-2 ranks and selects entities based on the query and the contextual knowledge retrieved from relevant documents, reducing ambiguity and ensuring accurate exploration for the next step. After, the LLM utilizes heterogeneous knowledge, including triple paths and entity contexts, to either answer the question or proceed with further rounds of retrieval if the information gathered is insufficient. In this section, we will explain each step in detail.

Next, we will provide a detailed introduction to how ToG-2 iteratively utilizes both structured and unstructured knowledge for reasoning. Formally, in the $(i+1)_{th}$ iteration, given the original question q, the clue queries from the i_{th} iteration $Q_c^i = \{q_1^i, q_2^i, \ldots, q_N^i\}$, the topic entities $E_{topic}^i = \{e_1, e_2, \ldots, e_N\}$ and their preceding triple paths $P^i = \{P_1^i, P_2^i, \ldots, P_N^i\}$,$P_j^i = \{p_0^j, p_1^j, \ldots, p_i^j\}$, each iteration includes three steps: relation prune (RP), entity prune (EP), and examine and reasoning (ER). Note that $i = 0$ indicates the initialization phase, and the P^0 is empty.

•**Relation Prune (RP)**: Based on q and Q_c^i, we prompt the LLM to select the relations that are most likely to find entities containing helpful context information for solving q and that match the description of Q_c^i. Unlike selecting relations for a single topic entity at a time, we provide GPT-3.5 with all topic entities within a single prompt. This approach not only reduces the number of API calls, thereby accelerating inference time but also enables the LLM to simultaneously consider the interconnections between multiple reasoning paths, allowing it to make selections from a more global perspective. The selected relations for the topic entity e_j are denoted as $R_j = \{r_{j1}, r_{j2}, \ldots, r_{jW}\}$, where W denotes the hyper-parameter width.

•**Entity Prune (EP)**: Given a topic entity e_j and one of the selected relation r_{jk}, ToG-2 will identify all interconnected candidate entity nodes $\{e_{jkl}\}$ within the Wiki Knowledge Graph (KG) and get their associated Wikipedia page documents D_{jkl} through locally deployed service. The document context of each candidate entity is initially segmented into appropriately sized chunks $\{t_{jklm}\}$. Subsequently, a two-

[1]https://www.wikidata.org/wiki
[2]https://www.dbpedia.org/
[3]https://www.geonames.org/

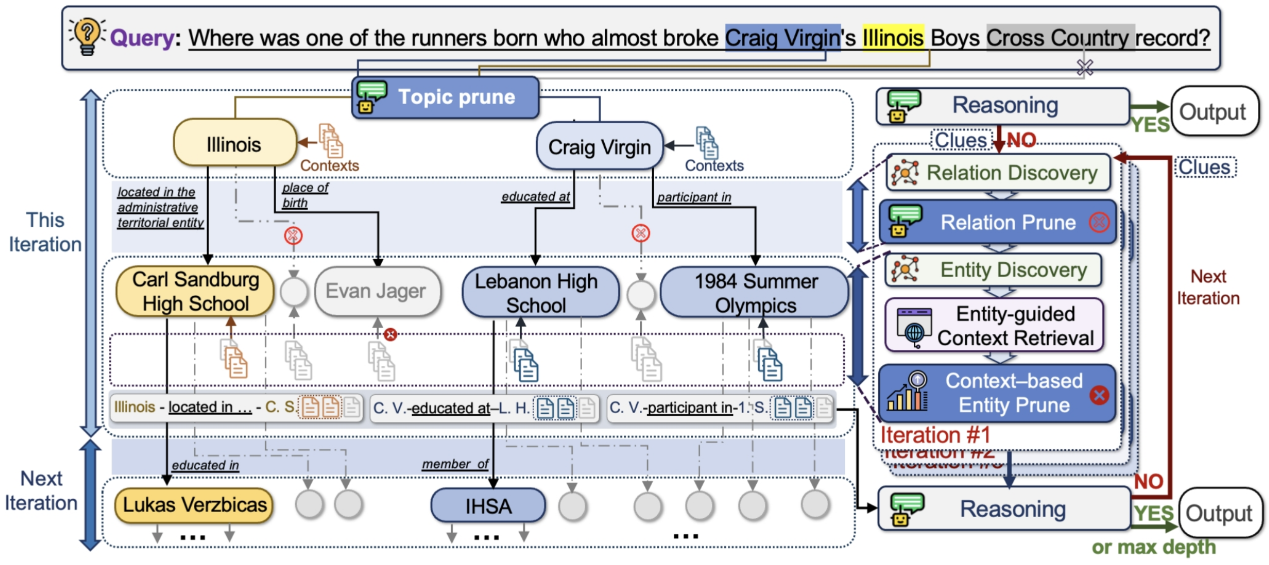


Fig. 5. An example illustrating the workflow of ToG-2.

stage search F_{retr} is employed, utilizing pre-trained language models for all candidate entities' chunks. Formally, $s_{jklm} = F_{retr}([q, q_j^i, p_{jkl}^i], t_{jklm})$, denotes the relevance score of the m_{th} paragraph of the l_{th} candidate, where p_{jkl}^i is the triples from which the current candidate entity is derived. Then, the ranking score of a candidate entity e_{jkl} is calculated as the exponentially decayed weighted sum of scores of its chunks that rank in top-K, and the weight for the i_{th} ranked chunk is calculated as $w = e^{-\alpha \cdot i}$, where K and α are hyper-parameters. Finally, top-W candidate entities are selected as the new topic entities E_{topic}^{i+1} for the next iteration; meanwhile, the corresponding preceding triple paths P^{i+1} will be updated.

•**Examine and reasoning (ER)**: Following RP and EP, we give LLM carefully aggregated references, including q, Q_c^i, P^{i+1} and the top $L(L \leq K)$ chunks. Then the LLM is prompted to examine the logical coherence and the completeness of factual evidence. If the LLM believes it can answer the question, the iteration ends. If not, based on the question and the collected contextual clues, a new clue query needs to be generated for the next round.

B. Documents-Based Methods

Our proposed approach, HippoRAG, is closely inspired by the Hippocampal Memory Indexing Theory, as shown in Fig.6. Tasks that require knowledge integration are particularly challenging for current RAG systems. In the above example, we want to find a Stanford professor who does Alzheimer's research from a pool of passages describing potentially thousands of Stanford professors and Alzheimer's researchers. Since current methods encode passages in isolation, they would struggle to identify Prof. Thomas unless a passage mentions both characteristics at once. In contrast, most people familiar with this professor would remember him quickly due to our brain's associative memory capabilities, thought to be driven by the index structure depicted in the C-shaped hippocampus above (in blue). Inspired by this mechanism, HippoRAG allows LLMs to build and leverage a similar graph of associations to tackle knowledge integration tasks.

• Motivation of Hippocampal Memory Indexing Theory. The hippocampal memory indexing theory [48] is a well-established theory that provides a functional description of the components and circuitry involved in human long-term memory. In this theory, Teyler and Discenna [48] propose that human long-term memory is composed of three components that work together to accomplish two main objectives: pattern separation, which ensures that the representations of distinct perceptual experiences are unique, and pattern completion, which enables the retrieval of complete memories from partial stimuli [49], [50].

The theory suggests that pattern separation is primarily accomplished in the memory encoding process, which starts with the neocortex receiving and processing perceptual stimuli into more easily manipulatable, likely higher-level, features, which are then routed through the parahippocampal regions (PHR) to be indexed by the hippocampus. When they reach the hippocampus, salient signals are included in the hippocampal index and associated with each other.

After the memory encoding process is completed, pattern completion drives the memory retrieval process whenever the hippocampus receives partial perceptual signals from the PHR pipeline. The hippocampus then leverages its context-dependent memory system, thought to be implemented through a densely connected network of neurons in the CA3 sub-region [49], to identify complete and relevant memories within the hippocampal index and route them back through the PHR for simulation in the neocortex. Thus, this complex process allows for new information to be integrated by changing only the hippocampal index instead of updating neocortical representations.

• **Offline Indexing.** Our offline indexing phase, analogous

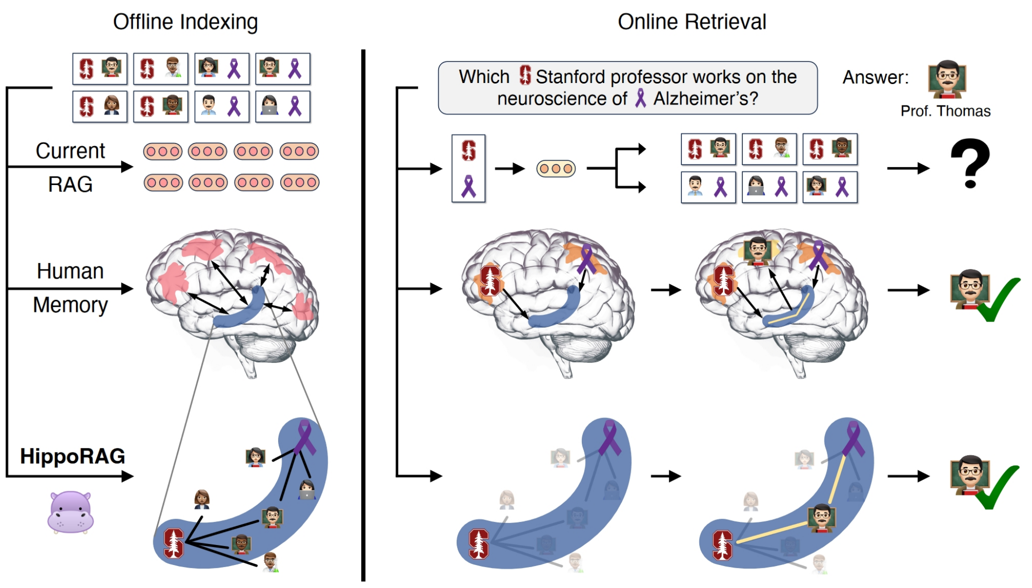


Fig. 6. Knowledge Integration & Hippocampal Memory Indexing Theory..

to memory encoding, starts by leveraging a strong instruction-tuned LLM, our artificial neocortex, to extract knowledge graph (KG) triples. The KG is schema-less, and this process is known as open information extraction (OpenIE) [51]–[54]. This process extracts salient signals from passages in a retrieval corpus as discrete noun phrases rather than dense vector representations, allowing for more fine-grained pattern separation. It is, therefore, natural to define our artificial hippocampal index as this open KG, which is built on the whole retrieval corpus passage-by-passage. Finally, to connect both components as is done by the parahippocampal regions, we use off-the-shelf dense encoders fine-tuned for retrieval (retrieval encoders). These retrieval encoders provide additional edges between similar but not identical noun phrases within this KG to aid in downstream pattern completion.

Next, we will introduce more details in offline indexing. Our indexing process involves processing a set of passages P using an instruction-tuned LLM L and a retrieval encoder M. As seen in Fig.7, we first use L to extract a set of noun phrase nodes N and relation edges E from each passage in P via OpenIE. This process is done via 1-shot prompting of the LLM. Specifically, we first extract a set of named entities from each passage. We then add the named entities to the OpenIE prompt to extract the final triples, which also contain concepts (noun phrases) beyond named entities. We find that this two-step prompt configuration leads to an appropriate balance between generality and bias towards named entities. Finally, we use M to add the extra set of synonymy relations E' discussed above when the cosine similarity between two entity representations in N is above a threshold τ. As stated above, this introduces more edges to our hippocampal index and allows for more effective pattern completion. This indexing process defines a $|N| \times |P|$ matrix P, which contains the number of times each noun phrase in the KG appears in each original passage.

- **Online Retrieval.** These same three components are then leveraged to perform online retrieval by mirroring the human brain's memory retrieval process. Just as the hippocampus receives input processed through the neocortex and PHR, our LLM-based neocortex extracts a set of salient named entities from a query which we call query-named entities. These named entities are then linked to nodes in our KG based on the similarity determined by retrieval encoders; we refer to these selected nodes as query nodes. Once the query nodes are chosen, they become the partial cues from which our synthetic hippocampus performs pattern completion. In the hippocampus, neural pathways between elements of the hippocampal index enable relevant neighborhoods to become activated and recalled upstream. To imitate this efficient graph search process, we leverage the Personalized PageRank (PPR) algorithm [55], a version of PageRank that distributes probability across a graph only through a set of user-defined source nodes. This constraint allows us to bias the PPR output only towards the set of query nodes, just as the hippocampus extracts associated signals from specific partial cues. Finally, as is done when the hippocampal signal is sent upstream, we aggregate the output PPR node probability over the previously indexed passages and use that to rank them for retrieval.

During the retrieval process, we prompt L using a 1-shot prompt to extract a set of named entities from a query q, our previously defined query named entities $C_q = \{c_1, \ldots, c_n\}$ (Stanford and Alzheimer's in our Fig.7 example). These named entities C_q from the query are then encoded by the same retrieval encoder M. Then, the previously defined query nodes are chosen as the set of nodes in N with the highest cosine similarity to the query named entities C_q. More formally, query nodes are defined as $R_q = \{r_1, \ldots, r_n\}$ such that $r_i = e_k$ where $k = \arg\max_j cosine\_similarity(M(c_i), M(e_j))$, represented as the Stanford logo and the Alzheimer's purple ribbon symbol in Fig.7.

After the query nodes R_q are found, we run the PPR algorithm over the hippocampal index, i.e., a KG with $|N|$

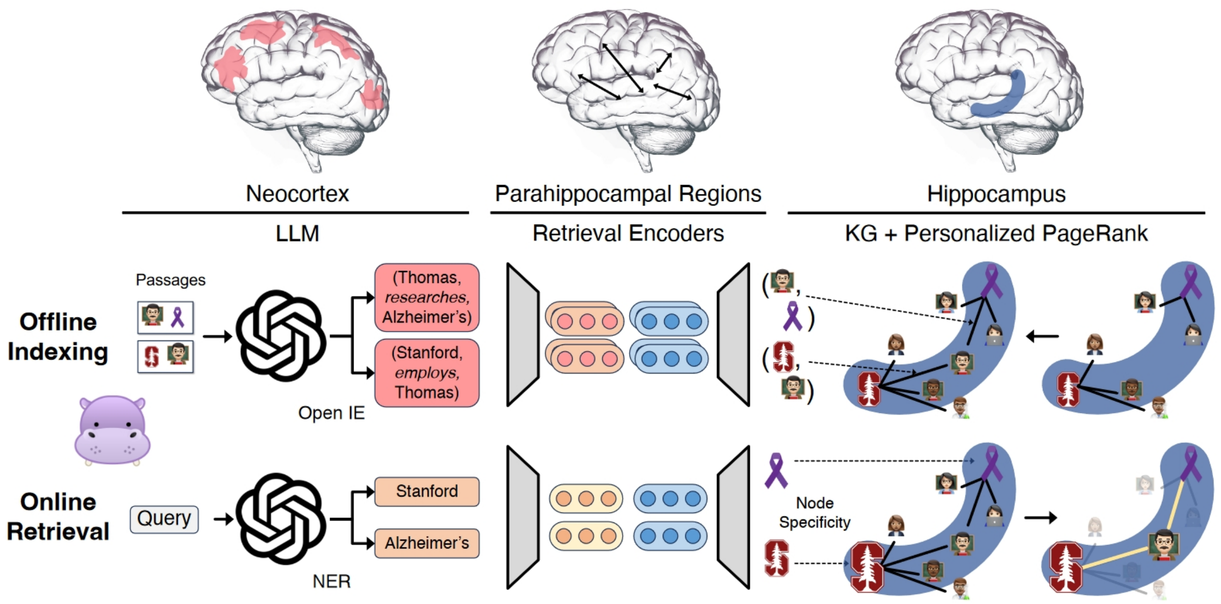


Fig. 7. Detailed HippoRAG Methodology.

nodes and $|E|+|E'|$ edges (triple-based and synonymy-based), using a personalized probability distribution $\vec{n}$ defined over N, in which each query node has equal probability and all other nodes have a probability of zero. This allows probability mass to be distributed to nodes that are primarily in the (joint) neighborhood of the query nodes, such as Professor Thomas, and contribute to eventual retrieval. After running the PPR algorithm, we obtain an updated probability distribution $\vec{n'}$ over N. Finally, in order to obtain passage scores, we multiply $\vec{n'}$ with the previously defined P matrix to obtain $\vec{p}$, a ranking score for each passage, which we use for retrieval.

•**Node Specificity.** We introduce node specificity as a neurobiologically plausible way to further improve retrieval. It is well known that global signals for word importance, like inverse document frequency (IDF), can improve information retrieval. However, in order for our brain to leverage IDF for retrieval, the number of total "passages" encoded would need to be aggregated with all node activations before memory retrieval is complete. While simple for normal computers, this process would require activating connections between an aggregator neuron and all nodes in the hippocampal index every time retrieval occurs, likely introducing prohibitive computational overhead. Given these constraints, we propose node specificity as an alternative IDF signal that requires only local signals and is thus more neurobiologically plausible. We define the node specificity of node i as $s_i = |P_i|^{-1}$, where P_i is the set of passages in P from which node i was extracted, information that is already available at each node. Node specificity is used in retrieval by multiplying each query node probability $\vec{n}$ with s_i before PPR; this allows us to modulate each of their neighborhood's probability as well as their own. We illustrate node specificity in Fig. 7 through relative symbol size: the Stanford logo grows larger than the Alzheimer's symbol since it appears in fewer documents.

C. Global-Information-Based Methods

Next, we introduce a global information-based method that leverages the global perspective of communities to achieve document summarization tasks.

GraphRAG consists of two stages: the indexing and querying phases. In the indexing stage, a knowledge graph is constructed from documents, and communities are formed through clustering. During the querying phase, LLMs are used to analyze the communities, ultimately generating a global summary. The pipeline is shown in Fig. 8. Next, we will describe the key design parameters, techniques, and implementation details for each step.

•**Source Documents** → **Text Chunks**. A fundamental design decision is the granularity with which input texts extracted from source documents should be split into text chunks for processing. In the following step, each of these chunks will be passed to a set of LLM prompts designed to extract the various elements of a graph index. Longer text chunks require fewer LLM calls for such extraction but suffer from the recall degradation of longer LLM context windows [14], [56]. This behavior can be observed in Figure 2 in the case of a single extraction round (i.e., with zero gleanings): on a sample dataset (HotPotQA, [57]), using a chunk size of 600 tokens extracted almost twice as many entity references as when using a chunk size of 2400. While more references are generally better, any extraction process needs to balance recall and precision for the target activity.

•**Text Chunks** → **Element Instances**. The baseline requirement for this step is to identify and extract instances of graph nodes and edges from each chunk of source text. We do this using a multipart LLM prompt that first identifies all entities in the text, including their name, type, and description, before identifying all relationships between clearly related entities, including the source and target entities and a description of

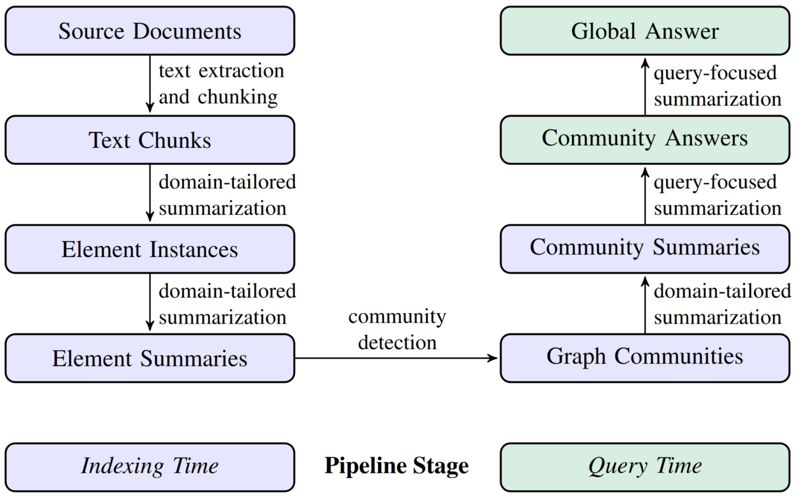


Fig. 8. Detailed HippoRAG Methodology.

their relationship. Both kinds of element instances are output in a single list of delimited tuples.

The primary opportunity to tailor this prompt to the domain of the document corpus lies in the choice of few-shot examples provided to the LLM for in-context learning [58]. For example, while our default prompt extracting the broad class of "named entities" like people, places, and organizations is generally applicable, domains with specialized knowledge (e.g., science, medicine, law) will benefit from few-shot examples specialized to those domains. We also support a secondary extraction prompt for any additional covariates we would like to associate with the extracted node instances. Our default covariate prompt aims to extract claims linked to detected entities, including the subject, object, type, description, source text span, and start and end dates.

To balance the needs of efficiency and quality, we use multiple rounds of "gleanings" up to a specified maximum to encourage the LLM to detect any additional entities it may have missed on prior extraction rounds. This is a multi-stage process in which we first ask the LLM to assess whether all entities were extracted, using a logit bias of 100 to force a yes/no decision. If the LLM responds that entities were missed, then a continuation indicating that "MANY entities were missed in the last extraction" encourages the LLM to glean these missing entities. This approach allows us to use larger chunk sizes without a drop in quality or the forced introduction of noise.

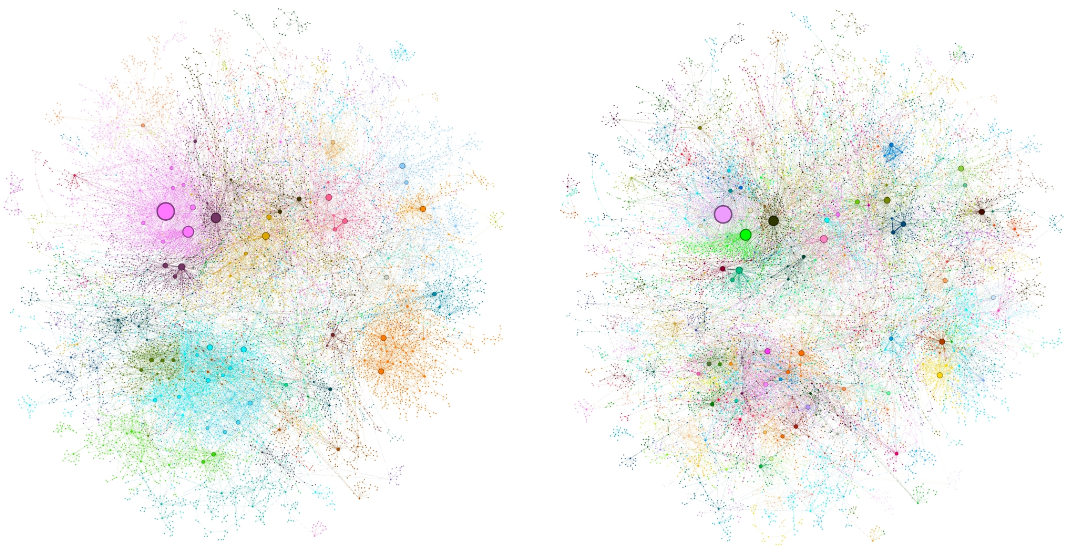
•**Element Instances** $\rightarrow$ **Element Summaries**. The use of an LLM to "extract" descriptions of entities, relationships, and claims represented in source texts is already a form of abstractive summarization, relying on the LLM to create independently meaningful summaries of concepts that may be implied but not stated by the text itself (e.g., the presence of implied relationships). Converting all such instance-level summaries into single blocks of descriptive text for each graph element (i.e., entity node, relationship edge, and claim covariate) requires a further round of LLM summarization over matching groups of instances.

A potential concern at this stage is that the LLM may not consistently extract references to the same entity in the same text format, resulting in duplicate entity elements and, thus, duplicate nodes in the entity graph. However, since all closely related "communities" of entities will be detected and summarized in the following step, and given that LLMs can understand the common entity behind multiple name variations, our overall approach is resilient to such variations, provided there is sufficient connectivity from all variations to a shared set of closely-related entities.

Overall, our use of rich descriptive text for homogeneous nodes in a potentially noisy graph structure is aligned with both the capabilities of LLMs and the needs of global, query-focused summarization. These qualities also differentiate our graph index from typical knowledge graphs, which rely on concise and consistent knowledge triples (subject, predicate, object) for downstream reasoning tasks.

•**Element Summaries** $\rightarrow$ **Graph Communities**. The index created in the previous step can be modeled as a homogeneous undirected weighted graph in which entity nodes are connected by relationship edges, with edge weights representing the normalized counts of detected relationship instances. Given such a graph, a variety of community detection algorithms may be used to partition the graph into communities of nodes with stronger connections to one another than to the other nodes in the graph (e.g., see the surveys by [59] and [60]). In our pipeline, we use Leiden [61] because of its ability to recover the hierarchical community structure of large-scale graphs efficiently. Each level of this hierarchy provides a community partition that covers the nodes of the graph in a mutually exclusive, collective-exhaustive way, enabling divide-and-conquer global summarization. Fig.9 shows an example of graph communities detected using the Leiden algorithm.

•**Graph Communities** $\rightarrow$ **Community Summaries**. The next step is to create report-like summaries of each community in the Leiden hierarchy, using a method designed to scale to very large datasets. These summaries are independently useful in their own right as a way to understand the global structure



(a) Root communities at level 0 (b) Sub-communities at level 1

Fig. 9. An example of graph communities detected using the Leiden algorithm.

and semantics of the dataset and may themselves be used to make sense of a corpus in the absence of a question. For example, a user may scan through community summaries at one level looking for general themes of interest, then follow links to the reports at the lower level that provide more details for each of the subtopics. Here, however, we focus on their utility as part of a graph-based index used for answering global queries.

Community summaries are generated in the following way:

- *Leaf-level communities.* The element summaries of a leaf-level community (nodes, edges, covariates) are prioritized and then iteratively added to the LLM context window until the token limit is reached. The prioritization is as follows: for each community edge in decreasing order of combined source and target node degree (i.e., overall prominence), add descriptions of the source node, target node, linked covariates, and the edge itself.
- *Higher-level communities.* If all element summaries fit within the token limit of the context window, proceed as for leaf-level communities and summarize all element summaries within the community. Otherwise, rank sub-communities in decreasing order of element summary tokens and iteratively substitute sub-community summaries (shorter) for their associated element summaries (longer) until fit within the context window is achieved.

•**Community Summaries** → **Community Answers** → **Global Answer**. Given a user query, the community summaries generated in the previous step can be used to generate a final answer in a multi-stage process. The hierarchical nature of the community structure also means that questions can be answered using community summaries from different levels, raising the question of whether a particular level in the hierarchical community structure offers the best balance of summary detail and scope for general sense-making questions.

For a given community level, the global answer to any user query is generated as follows:

- *Prepare community summaries.* Community summaries are randomly shuffled and divided into chunks of pre-specified token size. This ensures relevant information is distributed across chunks rather than concentrated (and potentially lost) in a single context window.
- *Map community answers.* Generate intermediate answers in parallel, one for each chunk. The LLM is also asked to generate a score between 0-100, indicating how helpful the generated answer is in answering the target question. Answers with a score of 0 are filtered out.
- *Reduce to global answer.* Intermediate community answers are sorted in descending order of helpfulness score and iteratively added into a new context window until the token limit is reached. This final context is used to generate the global answer returned to the user.

V. Experiments

We now present the experimental results. Section V-A discusses the RAG tasks, datasets, and evaluate matrices. We discuss the experimental results in Section V-B.

A. RAG tasks and evaluations.

GraphRAG is applied to various downstream tasks (especially in NLP tasks), including question answering (QA), information extraction, and others [62]. Among these, QA is the primary evaluation metric, as it provides a large dataset of question-answer pairs. Additionally, for global information, we also evaluate the summary generation capability of RAG.

•**KGQA.** For ToG-2, We evaluated our method on different knowledge-intensive reasoning benchmark datasets (i.e., KGQA, Knowledge graph question answering), including two multi-hop KBQA datasets WebQSP [63] and QALD10-en

Dataset	Example activity framing and generation of global sense-making questions
Podcast transcripts	*User:* A tech journalist looking for insights and trends in the tech industry. *Task:* Understanding how tech leaders view the role of policy and regulation. *Questions:* 1. Which episodes deal primarily with tech policy and government regulation? 2. How do guests perceive the impact of privacy laws on technology development? 3. Do any guests discuss the balance between innovation and ethical considerations? 4. What are the suggested changes to current policies mentioned by the guests? 5. Are collaborations between tech companies and governments discussed, and how?
News articles	*User:* Educator incorporating current affairs into curricula *Task:* Teaching about health and wellness *Questions:* 1. What current topics in health can be integrated into health education curricula? 2. How do news articles address the concepts of preventive medicine and wellness? 3. Are there examples of health articles that contradict each other, and if so, why? 4. What insights can be gleaned about public health priorities based on news coverage? 5. How can educators use the dataset to highlight the importance of health literacy?

TABLE I
EXAMPLES OF POTENTIAL USERS, TASKS, AND QUESTIONS GENERATED BY THE LLM BASED ON SHORT DESCRIPTIONS OF THE TARGET DATASETS. QUESTIONS TARGET GLOBAL UNDERSTANDING RATHER THAN SPECIFIC DETAILS.

[64], a multi-hop complex document QA dataset AdvHotpotQA [65] which is a challenging subset of HotpotQA [57], a slot filling dataset Zero-Shot RE [2], and two fact verification dataset FEVER [66] and Creak [67]. The evaluation metric for FEVER and Creak is Accuracy (Acc.), while the metric for other datasets is Exact Match (EM). Recall and F1 scores are not used since knowledge sources are not limited to document databases. Following previous work [31], full Wikipedia and WikiData are used as unstructured and structured knowledge sources for all of these datasets. Compared to the distractor setting, this full Wiki setting makes the retrieval process more challenging and can better evaluate the effectiveness of various methods on knowledge reasoning tasks.

We compare ToG-2 with both widely-used baselines and state-of-the-art methods to provide a more comprehensive overview: 1) LLM-only methods without external knowledge, including Direct Reasoning, Chain-of-Thought [68] (CoT), Self-Consistency [69] (CoT-SG); 2) Vanilla RAG, indicating the text-based RAG method that directly retrieves from entity documents and answers the question; 3) KG-based RAG method: Think-on-Graph [22](ToG), a KG-based RAG method that searches useful KG triples for reasoning; 4) Chain-of-Knowledge [31] (CoK), a hybrid RAG method retrieving knowledge from Wikipedia, Wikidata, and Wikitable. For a fair comparison, all baselines are used with GPT-3.5-turbo and evaluated under an unsupervised setting.

•**DocQA.** We evaluate our method's retrieval capabilities primarily on two challenging multi-hop QA benchmarks, MuSiQue (answerable) [70] and 2WikiMultiHopQA [71]. For completeness, we also include the HotpotQA [57] dataset, even though it has been found to be a much weaker test for multi-hop reasoning due to many spurious signals [70]. To limit the experimental cost, we extract 1,000 questions from each validation set as done in previous work [72], [73]. In order to create a more realistic retrieval setting, we follow IRCoT [73] and collect all candidate passages (including supporting and distractor passages) from our selected questions and form a retrieval corpus for each dataset. We report retrieval and QA performance on the datasets above using recall@2 and recall@5 (R@2 and R@5 below) for retrieval and exact match (EM) and F1 scores for QA performance.

We compare against several strong and widely used retrieval methods: BM25 [41], Contriever [74], GTR [75] and ColBERTv2 [76]. Additionally, we compare against two recent LLM-augmented baselines: Propositionizer [77], which rewrites passages into propositions, and RAPTOR [1], which constructs summary nodes to ease retrieval from long documents. In addition to the single-step retrieval methods above, we also include the multi-step retrieval method IRCoT [73] as a baseline.

• **Summary tasks.** Many benchmark datasets for open-domain question answering exist, including HotPotQA [57], MultiHop-RAG [78], and MT-Bench [79]. However, the associated question sets target explicit fact retrieval rather than summarization for the purpose of data sense-making, i.e., the process through which people inspect, engage with and contextualize data within the broader scope of real-world activities [80]. Similarly, methods for extracting latent summarization queries from source texts also exist [81], but such extracted questions can target details that betray prior knowledge of the texts.

To evaluate the effectiveness of RAG systems for more global sense-making tasks, we need questions that convey only a high-level understanding of dataset contents and not the details of specific texts. We used an activity-centered approach to automate the generation of such questions: given a short description of a dataset, we asked the LLM to identify N potential users and N tasks per user, then for each (user, task) combination, we asked the LLM to generate N questions that require understanding of the entire corpus. For our evaluation, a value of $N = 5$ resulted in 125 test questions per dataset. Table 1 shows example questions for each of the two evaluation datasets.

We compare six different conditions in our analysis, including Graph RAG using four levels of graph communities (**C0, C1, C2, C3**), a text summarization method applying our map-

Baseline Type	Method	Datasets					
		WebQSP (EM)	AdvHotpotQA (EM)	QALD-10-en (EM)	FEVER (Acc.)	Creak (Acc.)	Zero-Shot RE (EM)
LLM-only	Direct	65.9	23.1	42.0	51.8	89.7	27.7
	CoT	59.9	30.8	42.9	57.8	90.1	28.8
	CoT-SC	61.1	34.4	45.3	59.9	90.8	45.4
Text-based RAG	Vanilla RAG	67.9	23.7	42.4	53.8	89.7	29.5
KG-based RAG	ToG	76.2	26.3	<u>50.2</u>	52.7	**93.8**	<u>88.0</u>
Hybrid RAG	CoK	<u>77.6</u>	<u>35.4</u>	47.1	**63.5**	90.4	75.5
Proposed	ToG-2	**81.1**	**42.9**	**54.1**	<u>63.1</u>	<u>93.5</u>	**91.0**

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT METHODS ACROSS VARIOUS DATASETS IN KGQA. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINED RESPECTIVELY.

Datasets	Llama-3-8B		Qwen2-7B		GPT-3.5-turbo		GPT-4o	
	Direct	ToG-2	Direct	ToG-2	Direct	ToG-2	Direct	ToG-2
AdvHotpotQA	20.8	34.7 (66.8% ↑)	17.9	30.8 (72.1% ↑)	23.1	42.9 (85.7% ↑)	47.7	53.3 (11.3% ↑)
FEVER	35.5	52.9 (49.0% ↑)	38.6	53.1 (38.1% ↑)	51.8	63.1 (21.8% ↑)	66.2	70.1 (5.9% ↑)

TABLE III
PERFORMANCE COMPARISON OF DIRECT REASONING AND TOG-2 WITH DIFFERENT BACKBONE MODELS.

reduce approach directly to source texts (**TS**), and a naive "semantic search" RAG approach (**SS**):

- **CO.** Uses root-level community summaries (fewest in number) to answer user queries.
- **C1.** Uses high-level community summaries to answer queries. These are sub-communities of C0; if present. Otherwise, C0 communities projected down.
- **C2.** Uses intermediate-level community summaries to answer queries. These are subcommunities of C1, if present. Otherwise, C1 communities are projected down.
- **C3.** Uses low-level community summaries (greatest in number) to answer queries. These are sub-communities of C2, if present, otherwise C2 communities projected down.
- **TS.** The same method as in subsection 2.6, except source texts (rather than community summaries) are shuffled and chunked for the map-reduce summarization stages.
- **SS.** An implementation of naive RAG in which text chunks are retrieved and added to the available context window until the specified token limit is reached.

The size of the context window and the prompts used for answer generation are the same across all six conditions (except for minor modifications to reference styles to match the types of context information used). Conditions only differ in how the contents of the context window are created.

LLMs have been shown to be good evaluators of natural language generation, achieving state-of-the-art or competitive results compared to human judgments [79], [82]. While this approach can generate reference-based metrics when gold standard answers are known, it is also capable of measuring the qualities of generated texts (e.g., fluency) in a reference-free style [82] as well as in head-to-head comparison of competing outputs (LLMas-a-judge, [79]). LLMs have also shown promise at evaluating the performance of conventional RAG systems, automatically evaluating qualities like context relevance, faithfulness, and answer relevance (RAGAS, [83]).

Given the multi-stage nature of our Graph RAG mechanism, the multiple conditions we wanted to compare, and the lack of gold standard answers to our activity-based sense-making questions, we decided to adopt a head-to-head comparison approach using an LLM evaluator. We selected three target metrics capturing qualities that are desirable for sense-making activities, as well as a control metric (directness) used as an indicator of validity. Since directness is effectively in opposition to comprehensiveness and diversity, we would not expect any method to win across all four metrics.

Our head-to-head measures computed using an LLM evaluator are as follows:

- *Comprehensiveness.* How much detail does the answer provide to cover all aspects and details of the question?
- *Diversity.* How varied and rich is the answer in providing different perspectives and insights on the question?
- *Empowerment.* How well does the answer help the reader understand and make informed judgments about the topic?
- *Directness.* How specifically and clearly does the answer address the question?

B. Main results.

•**KGQA Experimental result.** The main results of several open-source datasets are shown in TABLE II. We note that ToG-2 outperforms other baselines on WebQSP, AdvHotpotQA, QALD-10-en, and Zero-Shot RE. Notably, WebQSP, AdvHotpotQA, and QALD-10-en are multi-hop reasoning datasets. On Fever, ToG-2 has a competitive performance to CoK since the fact statements in Fever mainly are about

Method	MuSiQue		2Wiki		HotpotQA		Average	
	R@2	R@5	R@2	R@5	R@2	R@5	R@2	R@5
BM25	32.3	41.2	51.8	61.9	55.4	72.2	46.5	58.4
Contriever	34.8	46.6	46.6	57.5	57.2	75.5	46.2	59.9
GTR	37.4	49.1	60.2	67.9	59.4	73.3	52.3	63.4
ColBERTv2	37.9	49.2	59.2	68.2	**64.7**	**79.3**	53.9	65.6
RAPTOR	5.7	45.3	46.3	53.8	58.1	71.2	46.7	56.8
Proposition	37.6	49.3	56.4	63.1	58.7	71.1	50.9	61.2
HippoRAG (Contriever)	**41.0**	**52.1**	**71.5**	**89.5**	59.0	76.2	<u>57.2</u>	<u>72.6</u>
HippoRAG (ColBERTv2)	<u>40.9</u>	<u>51.9</u>	<u>70.7</u>	<u>89.1</u>	<u>60.5</u>	<u>77.7</u>	**57.4**	**72.9**

TABLE IV
SINGLE-STEP RETRIEVAL PERFORMANCE. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINED, RESPECTIVELY.

Method	MuSiQue		2Wiki		HotpotQA		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
None	12.5	24.1	31.0	39.6	30.4	42.8	24.6	35.5
ColBERTv2	15.5	26.4	33.4	43.3	43.4	57.7	30.8	42.5
HippoRAG (ColBERTv2)	<u>19.2</u>	29.8	<u>46.6</u>	<u>59.5</u>	41.8	55.0	<u>35.9</u>	<u>48.1</u>
IRCoT (ColBERTv2)	19.1	<u>30.5</u>	35.4	45.1	<u>45.5</u>	<u>58.4</u>	33.3	44.7
IRCoT + HippoRAG (ColBERTv2)	**21.9**	**33.3**	**47.7**	**62.7**	**45.7**	**59.2**	**38.4**	**51.7**

TABLE V
DOCUMENT QA PERFORMANCE. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINED, RESPECTIVELY.

single-hop relations and thus do not need in-depth information retrieval. In another fact verification dataset, Creak, all fact statements can be verified based on Wikidata. Thus, ToG-2 and ToG have similar performances on Creak. Meanwhile, compared to the original ToG, ToG-2 achieved a substantial improvement on AdvHotpotQA (16.6%) and also demonstrated notable enhancements on other datasets (4.93% on WebQSP, 3.85% on QALD-10-en, 3% on Zero-Shot RE, and 10.4% on FEVER). This demonstrates the advantages of our KG$\times$Text RAG framework in addressing complex problems.

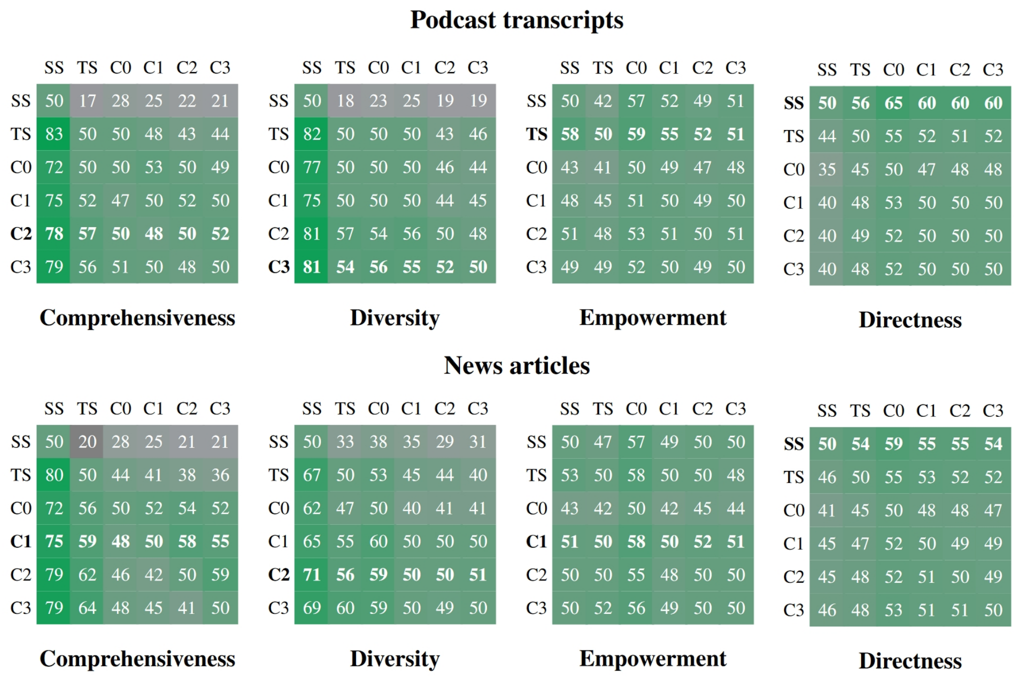
To explore the benefits of LLMs with varying capabilities for ToG-2, we analyzed its performance enhancement under different LLM backbone selections through experiments on AdvHotpotQA and FEVER. The experimental results are shown in Table III. We observe that ToG-2 can elevate the reasoning capability of weaker LLMs, e.g., Llama-3-8B, Qwen2-7B to the level of direct reasoning by more powerful LLMs, e.g., GPT-3.5-turbo, which supports our intuition that ToG-2 helps LLMs with knowledge and comprehension bottlenecks. On the other hand, powerful LLMs, e.g., GPT-3.5-turbo and GPT-4o, can still benefit from ToG-2 to improve their own performances on complex knowledge reasoning tasks, indicating that ToG-2 may achieve even higher performance with stronger LLMs. On AdvHotpotQA and FEVER, which use Wikipedia as knowledge sources, the most powerful LLMs among all backbone LLMs, GPT-4o experiences the least improvement since GPT-4o has a better memory for knowledge related to Wikipedia than other backbone LLMs, making its direct reasoning performance on these datasets already desirable.

•**DocQA experimental result.** As seen in TABLE IV, HippoRAG outperforms all other methods, including recent LLM-augmented baselines such as Propositionizer and RAPTOR, on our main datasets, MuSiQue and 2WikiMultiHopQA, while achieving competitive performance on HotpotQA. We notice an impressive improvement of 11 and 20% for R@2 and R@5 on 2WikiMultiHopQA and around 3% on MuSiQue. This difference can be partially explained by 2WikiMultiHopQA's entity-centric design, which is particularly well-suited for HippoRAG. Our lower performance on HotpotQA is mainly due to its lower knowledge integration requirements. More specifically, the distribution of the distractor scores in HotpotQA is much closer to the lower bound of the support passage scores than the other two datasets.

In terms of Question Answering Results, we report QA results for HippoRAG, the strongest retrieval baselines, ColBERTv2 and IRCoT, as well as IRCoT using HippoRAG as a retriever in TABLE V. As expected, improved retrieval performance in both single and multi-step settings leads to great overall improvements of up to 3%, 17%, and 1% F1 scores on MuSiQue, 2WikiMultiHopQA, and HotpotQA respectively using the same QA reader.

•**Summary task results.** *Global approaches vs. naive RAG.* As shown in Fig.10, global approaches consistently outperformed the naive RAG (**SS**) approach in both comprehensiveness and diversity metrics across datasets. Specifically, global approaches achieved comprehensiveness win rates between 72-83% for Podcast transcripts and 72-80% for News articles, while diversity win rates ranged from 75-82% and 62-71%, respectively. Our use of directness as a validity test also achieved the expected results, i.e., that naive RAG produces the most direct responses across all comparisons.

Community summaries vs. source texts. When comparing community summaries to source texts using Graph RAG, community summaries generally provided a small but consistent improvement in answer comprehensiveness and diversity, except for root-level summaries. Intermediate-level summaries



Podcast transcripts — Comprehensiveness

	SS	TS	C0	C1	C2	C3
SS	50	17	28	25	22	21
TS	83	50	50	48	43	44
C0	72	50	50	53	50	49
C1	75	52	47	50	52	50
C2	**78**	**57**	**50**	**48**	**50**	**52**
C3	79	56	51	50	48	50

Podcast transcripts — Diversity

	SS	TS	C0	C1	C2	C3
SS	50	18	23	25	19	19
TS	82	50	50	50	43	46
C0	77	50	50	50	46	44
C1	75	50	50	50	44	45
C2	81	57	54	56	50	48
C3	**81**	**54**	**56**	**55**	**52**	**50**

Podcast transcripts — Empowerment

	SS	TS	C0	C1	C2	C3
SS	50	42	57	52	49	51
TS	**58**	**50**	**59**	**55**	**52**	**51**
C0	43	41	50	49	47	48
C1	48	45	51	50	49	50
C2	51	48	53	51	50	51
C3	49	49	52	50	49	50

Podcast transcripts — Directness

	SS	TS	C0	C1	C2	C3
SS	**50**	**56**	**65**	**60**	**60**	**60**
TS	44	50	55	52	51	52
C0	35	45	50	47	48	48
C1	40	48	53	50	50	50
C2	40	49	52	50	50	50
C3	40	48	52	50	50	50

News articles — Comprehensiveness

	SS	TS	C0	C1	C2	C3
SS	50	20	28	25	21	21
TS	80	50	44	41	38	36
C0	72	56	50	52	54	52
C1	**75**	**59**	**48**	**50**	**58**	**55**
C2	79	62	46	42	50	59
C3	79	64	48	45	41	50

News articles — Diversity

	SS	TS	C0	C1	C2	C3
SS	50	33	38	35	29	31
TS	67	50	53	45	44	40
C0	62	47	50	40	41	41
C1	65	55	60	50	50	50
C2	**71**	**56**	**59**	**50**	**50**	**51**
C3	69	60	59	50	49	50

News articles — Empowerment

	SS	TS	C0	C1	C2	C3
SS	50	47	57	49	50	50
TS	53	50	58	50	50	48
C0	43	42	50	42	45	44
C1	**51**	**50**	**58**	**50**	**52**	**51**
C2	50	50	55	48	50	50
C3	50	52	56	49	50	50

News articles — Directness

	SS	TS	C0	C1	C2	C3
SS	**50**	**54**	**59**	**55**	**55**	**54**
TS	46	50	55	53	52	52
C0	41	45	50	48	48	47
C1	45	47	52	50	49	49
C2	45	48	52	51	50	49
C3	46	48	53	51	51	50

Fig. 10. Head-to-head win rate percentages of (row condition) over (column condition) across two datasets, four metrics, and 125 questions per comparison (each repeated five times and averaged).

	Podcast Transcripts					News Articles				
	CO	**C1**	**C2**	**C3**	**TS**	**CO**	**C1**	**C2**	**C3**	**TS**
Units	34	367	969	1310	1669	55	555	1797	2142	3197
Tokens	26657	225756	565720	746100	1014611	39770	352641	980898	1140266	1707694
% Max	2.6	22.2	55.8	73.5	100	2.3	20.7	57.4	66.8	100

TABLE VI
DOCUMENT QA PERFORMANCE. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINED, RESPECTIVELY.

in the Podcast dataset and low-level community summaries in the News dataset achieved comprehensiveness win rates of 57% and 64%, respectively. Diversity win rates were 57% for Podcast intermediate-level summaries and 60% for News low-level community summaries. Table 3 also illustrates the scalability advantages of Graph RAG compared to source text summarization: for low-level community summaries (**C3**), Graph RAG required 26-33% fewer context tokens, while for root-level community summaries (**C0**), it required over 97% fewer tokens. For a modest drop in performance compared with other global methods, root-level Graph RAG offers a highly efficient method for the iterative question answering that characterizes sense-making activity while retaining advantages in comprehensiveness (72% win rate) and diversity (62% win rate) over naive RAG.

Empowerment. Empowerment comparisons showed mixed results for both global approaches versus naive RAG (SS) and Graph RAG approaches versus source text summarization (**TS**). Ad-hoc LLM used to analyze LLM reasoning for this measure indicated that the ability to provide specific examples, quotes, and citations was judged to be key to helping users reach an informed understanding. Tuning element extraction prompts may help to retain more of these details in the Graph RAG index.

The indexing process resulted in a graph consisting of 8564 nodes and 20691 edges for the Podcast dataset and a larger graph of 15754 nodes and 19520 edges for the News dataset. TABLE.VI shows the number of community summaries at different levels of each graph community hierarchy.

VI. Conclusion

Graph RAG represents an advanced approach to enhancing knowledge retrieval in language models, addressing the limitations of existing systems based solely on KGs or text. Traditional RAG systems often struggle to retrieve deep knowledge for complex reasoning tasks, as they rely heavily on either KGs or text, without fully exploiting the synergy between both.

To overcome this challenge, we introduce the KG×Text RAG paradigm, which tightly integrates KG-based and text-based RAG systems. This hybrid approach allows for knowledge-guided context retrieval through KGs while leveraging textual information to improve graph-based retrieval. The ToG-2 framework, built upon this hybrid paradigm, enables a more reliable and comprehensive graph retrieval process. It performs iterative, cooperative retrieval between the KG and text, resulting in deeper, more accurate knowledge extraction for LLM reasoning. This ensures that complex knowledge retrieval tasks can be addressed with greater depth and faithfulness.

Additionally, HippoRAG offers a promising alternative, improving the efficiency of traditional RAG systems by overcoming their limitations. It integrates knowledge more effectively, especially in path-following multi-hop QA tasks, and provides significant efficiency gains. Its dynamic updating mechanism positions it as a robust solution for long-term memory in LLMs, balancing the advantages of parametric memory and standard RAG methods.

Furthermore, GraphRAG provides a global approach by combining knowledge graph generation, RAG, and query-focused summarization (QFS). This allows it to support human sense-making over entire corpora, offering significant improvements in data indexing. Summaries of root-level communities in the entity-based graph index outperform naive RAG methods and achieve competitive performance with other global methods, all while dramatically reducing token costs.

In summary, Graph RAG advances the field by enabling in-depth knowledge retrieval, enhancing the interpretability of responses, and improving efficiency, making it a powerful framework for next-generation LLM reasoning.

References

[1] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," *arXiv preprint arXiv:2401.18059*, 2024.

[2] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard *et al.*, "Kilt: a benchmark for knowledge intensive language tasks," *arXiv preprint arXiv:2009.02252*, 2020.

[3] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020.

[4] A. Talmor, Y. Elazar, Y. Goldberg, and J. Berant, "olmpics-on what language model pre-training captures," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 743–758, 2020.

[5] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[6] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[7] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[8] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.

[9] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 696–15 707.

[10] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" *arXiv preprint arXiv:2002.08910*, 2020.

[11] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung *et al.*, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.

[12] N. M. Guerreiro, D. M. Alves, J. Waldendorf, B. Haddow, A. Birch, P. Colombo, and A. F. Martins, "Hallucinations in large multilingual translation models," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1500–1517, 2023.

[13] 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," *ACM Transactions on Information Systems*, 2023.

[14] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.

[15] C. Mavromatis and G. Karypis, "Gnn-rag: Graph neural retrieval for large language model reasoning," *arXiv preprint arXiv:2405.20139*, 2024.

[16] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, "Grag: Graph retrieval-augmented generation," *arXiv preprint arXiv:2405.16506*, 2024.

[17] 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.

[18] 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.

[19] 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.

[20] M. Kang, J. M. Kwak, J. Baek, and S. J. Hwang, "Knowledge graph-augmented language models for knowledge-grounded dialogue generation," *arXiv preprint arXiv:2305.18846*, 2023.

[21] P. Sen, S. Mavadia, and A. Saffari, "Knowledge graph-augmented language models for complex question answering," in *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, 2023, pp. 1–8.

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

[23] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," *arXiv preprint arXiv:2310.01061*, 2023.

[24] J. Wang, Q. Sun, X. Li, and M. Gao, "Boosting language models reasoning with chain-of-knowledge prompting," *arXiv preprint arXiv:2306.06427*, 2023.

[25] 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.

[26] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. L. Bras, X. Lu, S. Welleck, and Y. Choi, "Symbolic knowledge distillation: from general language models to commonsense models," *arXiv preprint arXiv:2110.07178*, 2021.

[27] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, "Comet: Commonsense transformers for automatic knowledge graph construction," *arXiv preprint arXiv:1906.05317*, 2019.

[28] B. Chen and A. L. Bertozzi, "Autokg: Efficient automated knowledge graph generation for language models," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 3117–3126.

[29] J. Han, N. Collier, W. Buntine, and E. Shareghi, "Pive: Prompting with iterative verification improving graph-based generative capability of llms," *arXiv preprint arXiv:2305.12392*, 2023.

[30] K. Zhang, B. J. Gutiérrez, and Y. Su, "Aligning instruction tasks unlocks large language models as zero-shot relation extractors," *arXiv preprint arXiv:2305.11159*, 2023.

[31] X. Li, R. Zhao, Y. K. Chia, B. Ding, S. Joty, S. Poria, and L. Bing, "Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources," *arXiv preprint arXiv:2305.13269*, 2023.

[32] B. Sarmah, D. Mehta, B. Hall, R. Rao, S. Patel, and S. Pasquali, "Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction," in *Proceedings of the 5th ACM International Conference on AI in Finance*, 2024, pp. 608–616.

[33] 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.

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

[35] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. S. Liang, and J. Leskovec, "Deep bidirectional language-knowledge

graph pretraining," *Advances in Neural Information Processing Systems*, vol. 35, pp. 37 309–37 323, 2022.

[36] H. Zhu, H. Peng, Z. Lyu, L. Hou, J. Li, and J. Xiao, "Pre-training language model incorporating domain-specific heterogeneous knowledge into a unified representation," *Expert Systems with Applications*, vol. 215, p. 119369, 2023.

[37] Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia, "Longlora: Efficient fine-tuning of long-context large language models," *arXiv preprint arXiv:2309.12307*, 2023.

[38] Y. Ding, L. L. Zhang, C. Zhang, Y. Xu, N. Shang, J. Xu, F. Yang, and M. Yang, "Longrope: Extending llm context window beyond 2 million tokens," *arXiv preprint arXiv:2402.13753*, 2024.

[39] Y. Fu, R. Panda, X. Niu, X. Yue, H. Hajishirzi, Y. Kim, and H. Peng, "Data engineering for scaling language models to 128k context," *arXiv preprint arXiv:2402.10171*, 2024.

[40] B. Peng, J. Quesnelle, H. Fan, and E. Shippole, "Yarn: Efficient context window extension of large language models," *arXiv preprint arXiv:2309.00071*, 2023.

[41] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*. Springer, 1994, pp. 232–241.

[42] M. Levy, A. Jacoby, and Y. Goldberg, "Same task, more tokens: the impact of input length on the reasoning performance of large language models," *arXiv preprint arXiv:2402.14848*, 2024.

[43] T. Li, G. Zhang, Q. D. Do, X. Yue, and W. Chen, "Long-context llms struggle with long in-context learning," *arXiv preprint arXiv:2404.02060*, 2024.

[44] X. Zhang, Y. Chen, S. Hu, Z. Xu, J. Chen, M. Hao, X. Han, Z. Thai, S. Wang, Z. Liu *et al.*, "Bench: Extending long context evaluation beyond 100k tokens," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 15 262–15 277.

[45] X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, "G-retriever: Retrieval-augmented generation for textual graph understanding and question answering," *arXiv preprint arXiv:2402.07630*, 2024.

[46] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, no. 6, 1999, pp. 518–529.

[47] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *The VLDB Journal*, vol. 29, pp. 353–392, 2020.

[48] T. J. Teyler and P. DiScenna, "The hippocampal memory indexing theory." *Behavioral neuroscience*, vol. 100, no. 2, p. 147, 1986.

[49] T. J. Teyler and J. W. Rudy, "The hippocampal indexing theory and episodic memory: updating the index," *Hippocampus*, vol. 17, no. 12, pp. 1158–1169, 2007.

[50] H. Eichenbaum, "A cortical–hippocampal system for declarative memory," *Nature reviews neuroscience*, vol. 1, no. 1, pp. 41–50, 2000.

[51] G. Angeli, M. J. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 344–354.

[52] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[53] K. Pei, I. Jindal, K. C.-C. Chang, C. Zhai, and Y. Li, "When to use what: An in-depth comparative empirical analysis of openie systems for downstream applications," *arXiv preprint arXiv:2211.08228*, 2022.

[54] S. Zhou, B. Yu, A. Sun, C. Long, J. Li, H. Yu, J. Sun, and Y. Li, "A survey on neural open information extraction: Current status and future directions," *arXiv preprint arXiv:2205.11725*, 2022.

[55] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 517–526.

[56] Y. Kuratov, A. Bulatov, P. Anokhin, D. Sorokin, A. Sorokin, and M. Burtsev, "In search of needles in a 10m haystack: Recurrent memory finds what llms miss," *arXiv preprint arXiv:2402.10790*, 2024.

[57] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *arXiv preprint arXiv:1809.09600*, 2018.

[58] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[59] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[60] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, S. Y. Philip, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1149–1170, 2021.

[61] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[62] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, "Graph retrieval-augmented generation: A survey," *arXiv preprint arXiv:2408.08921*, 2024.

[63] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 201–206.

[64] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. Ngonga Ngomo *et al.*, "Qald-10–the 10th challenge on question answering over linked data," *Semantic Web*, no. Preprint, pp. 1–15, 2023.

[65] X. Ye and G. Durrett, "The unreliability of explanations in few-shot prompting for textual reasoning," *Advances in neural information processing systems*, vol. 35, pp. 30 378–30 392, 2022.

[66] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: a large-scale dataset for fact extraction and verification," *arXiv preprint arXiv:1803.05355*, 2018.

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

[68] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

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

[70] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Musique: Multihop questions via single-hop question composition," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 539–554, 2022.

[71] X. Ho, A.-K. D. Nguyen, S. Sugawara, and A. Aizawa, "Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps," *arXiv preprint arXiv:2011.01060*, 2020.

[72] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, "Measuring and narrowing the compositionality gap in language models," *arXiv preprint arXiv:2210.03350*, 2022.

[73] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," *arXiv preprint arXiv:2212.10509*, 2022.

[74] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised dense information retrieval with contrastive learning," *arXiv preprint arXiv:2112.09118*, 2021.

[75] J. Ni, C. Qu, J. Lu, Z. Dai, G. H. Ábrego, J. Ma, V. Y. Zhao, Y. Luan, K. B. Hall, M.-W. Chang *et al.*, "Large dual encoders are generalizable retrievers," *arXiv preprint arXiv:2112.07899*, 2021.

[76] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, and M. Zaharia, "Colbertv2: Effective and efficient retrieval via lightweight late interaction," *arXiv preprint arXiv:2112.01488*, 2021.

[77] T. Chen, H. Wang, S. Chen, W. Yu, K. Ma, X. Zhao, H. Zhang, and D. Yu, "Dense x retrieval: What retrieval granularity should we use?" *arXiv preprint arXiv:2312.06648*, 2023.

[78] Y. Tang and Y. Yang, "Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries," *arXiv preprint arXiv:2401.15391*, 2024.

[79] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 595–46 623, 2023.

[80] L. Koesten, K. Gregory, P. Groth, and E. Simperl, "Talking datasets–understanding data sensemaking behaviours," *International journal of human-computer studies*, vol. 146, p. 102562, 2021.

[81] Y. Xu and M. Lapata, "Text summarization with latent queries," *arXiv preprint arXiv:2106.00104*, 2021.

[82] J. Wang, Y. Liang, F. Meng, Z. Sun, H. Shi, Z. Li, J. Xu, J. Qu, and J. Zhou, "Is chatgpt a good nlg evaluator? a preliminary study," *arXiv preprint arXiv:2303.04048*, 2023.

[83] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," *arXiv preprint arXiv:2309.15217*, 2023.