GEM-RAG: Graphical Eigen Memories For Retrieval Augmented Generation

Brendan Hogan Rappazzo, Yingheng Wang, Aaron Ferber, Carla Gomes
Department of Computer Science
Cornell University
Ithaca, New York 14850
Email: {bhr54, yw2349, amf272, cpg5}@cornell.edu

2024 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-7488-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICMLA61862.2024.00196

***Abstract*—The ability to form, retrieve, and reason about memories in response to stimuli is central to general intelligence, enabling learning, adaptation, and insight. Large Language Models (LLMs), when given proper memories or context, can reason and respond effectively. However, they still struggle to optimally encode, store, and retrieve memories, a limitation that constrains their full potential as specialized AI agents. Retrieval Augmented Generation (RAG) seeks to address this by enriching LLMs with in-context examples. Inspired by human memory, we introduce Graphical Eigen Memories for Retrieval Augmented Generation (GEM-RAG), which tags information with LLM-generated "utility" questions, links information together in a graph by similarity, and uses eigendecomposition to form higher-level summary information. This approach not only enhances RAG tasks but also offers a way to explore text data sets. Using UnifiedQA, GPT-3.5 Turbo, SBERT, and OpenAI text encoders, we show that GEM-RAG outperforms state-of-the-art RAG methods on two standard QA tasks and discuss its implications for robust RAG systems.**

I. Introduction

The creation of intelligent machines has fascinated humanity for centuries, from medieval automata to modern computing and artificial intelligence. Today, the prospect of Artificial General Intelligence (AGI) feels closer than ever, especially with the rise of large-scale machine learning systems like Large Language Models (LLMs) and Large Multimodal Models (LMMs) [1]. LLMs have proven to be powerful knowledge repositories, excelling across various tasks [2], [3], [4], and their large context windows enable strong in-context learning for chat applications, classification, and reasoning tasks [5]. Perfecting memory encoding, storage, and retrieval in LLMs could enable AI agents to recall decades of conversations, research, and more, revolutionizing QA systems and expanding LMMs to handle multimodal data, unlocking even broader use cases.

Motivated by the need to adapt static LLMs to niche domains, retrieval augmented generation (RAG) has become an active research area [6], [7], [8], [9]. RAG typically works by splitting text into chunks, embedding them, and retrieving the top k nearest chunks for a given prompt [10].

However, current RAG systems are limited. They often retrieve superficially similar but irrelevant texts, and they struggle with synthesizing information from multiple chunks or understanding conceptual connections.

To address these issues, we drew inspiration from human cognition, where information is better encoded when its utility is understood [11], [12], and frequently retrieved memories are synthesized into higher-level summaries [13].

We propose Graphical Eigen Memories for Retrieval Augmented Generation (GEM-RAG), inspired by human cognition. Our method splits a text corpus into chunks and generates "utility" questions using an LLM. These questions form a weighted graph, where edges represent the similarity between utility questions. Using spectral decomposition of the graph's Laplacian, we identify key orthogonal themes—"eigenthemes"—which serve as summary nodes. During retrieval, the most similar utility question is identified, and a best-first search on the GEM retrieves the relevant context chunks for the LLM (Figure 1).

We believe GEM-RAG can transform LLMs into adaptable AI agents for niche domains. To evaluate our method, we tested it on the QuALITY [14] and Qasper [15] QA datasets, comparing it against standard RAG and RAPTOR [16] using SBERT [17] and OpenAI's text-embedding-ada-002 models, as well as UnifiedQA [18] and GPT-3 [19]. Our method generally outperformed these approaches. We also conducted ablation experiments to assess the impact of the number of eigen summary nodes and utility questions. Lastly, GEM can function independently of LLMs, enabling data exploration and visualization, as demonstrated in our web demo.

II. Related Work

Large Language Models (LLMs) LLMs have proven to be extremely powerful general knowledge stores [20], [18], [21]. It has also been shown in some cases, that fine-tuning can produce data specific models [22]. The advent of better hardware and algorithms has allowed for them to have larger context lengths, which can handle more information to be learned or retrieved in-context. However, it has been shown that longer contexts have diminishing returns, can lead to a loss of information [23], [24], still necessitating the use of retrieving relevant contextual information.

It has been shown that large-context models excel at short story QA tasks [25], but they retrieve entire stories with thousands of tokens. In contrast, we focus on returning only hundreds of tokens, as done in RAG methods, which is further explained in the experimental section.

1946-0759/24/$31.00 ©2024 IEEE
DOI 10.1109/ICMLA61862.2024.00196

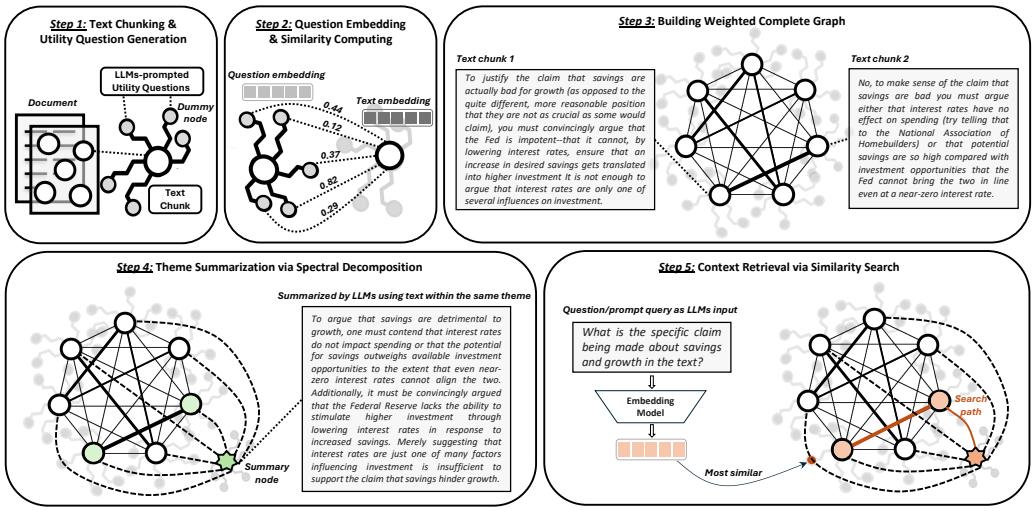

Fig. 1. An overview of the graph construction and retrieval process for GEM-RAG. Given a corpus of text, the text is first grouped into chunks of text. GEM-RAG then generates utility questions for each chunk of text using an LLM, where a utility question asks something that could be answered given the text chunk as context. Next GEM-RAG uses these utility questions and respective embeddings to build a weighted graph. Summary nodes are then generated using the graph's spectral decomposition, using the eigenvectors to represent different orthogonal modes or "eigenthemes" of the text. For retrieval, GEM-RAG embeds the question or prompt, and searches the graph for the optimal nodes or context to return.

Retrieval Augmentation Methods Retrieval-augmented generation has gained attention recently. [6] introduced augmenting LLM context with retrieved information, followed by work on tracing the source of responses [7], [8], [9], [26], [10]. [27] proposed jointly learning the retriever and LLM, while [28] introduced tree-decoding for multi-answer retrieval. Hierarchical data summaries have also been explored [29], [30], including recursive summarization [31]. While RAG improves baselines, it often lacks key context [32], leading to studies on summary generation and custom encoder modules [33], [34]. Recent work on tree-structured chunk summaries [16] doesn't account for textual similarity biases or go beyond tree structures. Our method uses utility questions for similarity, builds a fully connected graph, and applies eigendecomposition for higher-level summaries.

III. Methods

A. Connection to Human Cognition

GEM-RAG is motivated by human cognition, namely, the processes by which humans encode, store and retrieve information. Specifically, we draw inspiration from how it is believed the human brain prioritizes information based on its utility, and that information that is most often retrieved together gets summarized together [11].

Our first observation from psychology is the so called "testing effect", which observes that if humans are tested on subject material, they are more likely to accurately remember it [12]. This may be because, by testing, the information can be associated to a specific utility in our cognition, and thus is "tagged" with information that makes it easier to retrieve. To this end, we aim to better tag each chunk of information, by "tagging" it with LLM generated utility questions, which help express what information a specific text chunk has, and why it might be useful.

Our second observation in that in human cognition, the more often memories are retrieved together, the more likely the will continue to be retrieved together [13]. In the context of our utility questions method and their respective text embeddings, we can build a graph constituting all chunks of the text as nodes, and the strength of their connections as the similarity of their question embeddings. We can then, inspired by this observation of human cognition, try to build higher level summary nodes based off the modes of the graph, i.e. the nodes that are likely to be retrieved together given a prompt. Our intuition is that by performing an eigendecomposition, each eigenvector will capture different mode or themes in the text, that would often be retrieved together, and thus should be used to generate summary nodes.

B. Graphical Eigen Memories

Our method at a high level involves several steps to first construct the graph, by chunking the text, generating utility questions, building the initial graph, and finally using the eigenvectors from spectral decomposition of normalized graph Laplacian to build the summary nodes. Then, with the GEM produced, we show how it can be used to perform RAG. The specifics of our implementation are discussed in the experiments section. A schematic of our method can be seen in Figure 1.

1) Memory Graph Construction: **Chunking** Text chunking is standard practice in RAG where the corpus of text of interest C is split apart into chunks, $\{\texttt{CHUNK}_1 \dots \texttt{CHUNK}_n\}$, where each chunk, $\texttt{CHUNK}_i$ has some number T tokens. In practice the text can be split by number of characters or number of tokens. For a corpus of text we first chunk the text into n chunks, where

each chunk is T tokens long, and $n = \frac{N}{c}$, where N is the total number of tokens in the text.

Generating Utility Questions Given each text $\texttt{CHUNK}_i$, we then prompt an LLM to generate some m number of utility questions. This can be represented by a function Q, that takes a chunk of text, and an integer, m, and generates m utility questions. Formally, given $\texttt{CHUNK}_i$, we compute $Q(\texttt{CHUNK}_i, m)$ which give a set of utility questions $\{q_{i1}, q_{i2}, \ldots, q_{im}\}$.

Text Embeddings In order to quantify the similarity of each utility question to each other and/or to a prompt, we need to embed the text into a high dimensional feature space, given by a text encoder. More specifically, given a text embedding function E, for each node i, we compute the embedding v as follows: $v_i = E(\texttt{CHUNK}_i)$. Similarly, we also compute this embedding for each utility question by the same function. However, in order to best encode the information of the utility question we make the utility question's embedding the average of the questions embedding, and the base text's embedding, that is $v_{ij} = (E(q_{ij}) + v_i)/2$.

Building the Weighted Complete Graph Given the embedding of each chunk of text, as well as each chunk's corresponding utility questions and respective embeddings, we then generate the fully connected graph. For each node/chunk pair i, j, we consider the sum of the similarity metric between all of node i's utility question embeddings, to that of node j's base text embedding. More formally, let $G = (\mathcal{N}, E)$ be the memory graph we are constructing. Let the nodes be given by: $\mathcal{N} = \{\texttt{CHUNK}_1, \texttt{CHUNK}_2, \ldots, \texttt{CHUNK}_K\}$, where each node t has utility questions $Q_t = \{q_{t1}, q_{t2}, \ldots, q_{tm}\}$, and base text embedding v_t. Then, For each $(t, v) \in \mathcal{N} \times \mathcal{N}$, and for each $i \in \{1, 2, \ldots, m\}$, generate an edge between t and v with weight $\sum_{i=0}^{m} \texttt{SIM}(v_{ti}, v_v)$. Any function could be used to compute the similarity, but in all cases we use standard cosine similarity, i.e., $\texttt{SIM}(a, b) = \frac{a \cdot b}{\|a\| \|b\|}$.

Building the Summary Nodes In order to build an encoding system that encodes this higher level information we formulate this as a random walk or spectral decomposition problem. Intuitively, in this context, each eigenvalue and its corresponding eigenvector reveal a distinct 'theme' or conceptual dimension in the graph. By summarizing the top component nodes of each eigenvector components, using an LLM, we can understand the most significant relationships and conceptual clusters within the graph.

More specifically, let $S = (s_{ij})_{n \times n}$ be the similarity matrix of the graph, where s_{ij} is the sum weight between the m $\texttt{SIM}$ values of each utility questions of nodes i and to node j. By attaching each s_{ij} to a weighted edge $e_i j$, we can map the similarity matrix S onto the memory graph G. Thus, we can better understand the relationship between different text pieces of the document by analyzing the properties and behaviors of G.

Since different documents possess different connectivity and node centrality, the spectrums will also be at different scales. To better quantify how influential each node is without degree bias, we transform S to a variant of normalized graph Laplacian L, which is $L = D^{-1/2}(S - I)D^{-1/2}$, where

Algorithm 1 Retrieval from GEM-RAG

Input: p (prompt), $\mathcal{B}$ (budget)
Output: $\mathcal{C}$ (set of context nodes)

$v_p \leftarrow E(p)$ ▷ Embed the prompt p using the embedding function E
$\mathcal{Q} \leftarrow$ Set of all utility question embeddings
$\mathcal{C} \leftarrow \{\emptyset\}$ ▷ Initialize retrieval set $\mathcal{C}$ to the empty set.
while $|\mathcal{C}| < \mathcal{B}$ **do** ▷ Expand the set until the budget $\mathcal{B}$ is reached
    $n = \underset{n_i \in \mathcal{Q} \setminus \mathcal{C}}{\operatorname{argmax}} \texttt{SIM}(v_p, v_{ni})$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{n\}$ ▷ Add the most relevant node n to the set $\mathcal{C}$
end while
return $\mathcal{C}$ ▷ Return the set $\mathcal{C}$ as the context for the prompt p

$D = \text{diag}(d_i)$ is the degree matrix and I is the identity matrix. Then we conduct spectral decomposition by solving the following $L\vec{x} = \lambda\vec{x}$. With the resulting eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ ordered in non-increasing order of their magnitude. The corresponding eigenvectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ represent the principal themes. Then, for each eigenvector $\vec{x}_k$, select the top e components, $x_{k1}, x_{k2}, \ldots, x_{ke}$, representing the most relevant nodes for the k-th theme. Then we prompt an LLM, given the text passages associated with $x_{k1}, x_{k2}, \ldots, x_{ke}$, to summarize the text, summarizing the high-level information. With the produced summary text, we introduce it as a new node in the graph. Lastly, as in previous steps, we produce utility questions, embeddings for the utility questions, and connect it to every other node in the graph in the manner previously described.

Analysis of Graph Spectrum

From spectral graph theory [35], [36], the eigenvalues of L exhibit these properties: (1) $\sum \lambda_i = 0$, (2) each eigenvalue λ_i falls within the range $[-1, 1]$, and (3) the largest eigenvalue λ_1 is 1. The similarity matrix $S - I$, with zero diagonals, has $(n^2 - n)$ non-zero elements, leading to an $O(n^2)$ complexity for the normalized Laplacian transformation. Subsequent spectral decomposition via the Lanczos algorithm [37] incurs an $O(n^3)$ computational complexity.

Given that all text chunk nodes are clustered into different themes via spectral decomposition, we can observe some interesting properties on S from such clustering behavior.

Remark 1: Suppose the document includes k essential themes. The modularity of S can be formulated as follows:

$$S = \begin{bmatrix} \hat{S}_{11} & \cdots & \hat{S}_{1k} \\ \vdots & \ddots & \vdots \\ \hat{S}_{k1} & \cdots & \hat{S}_{kk} \end{bmatrix}$$

where $\hat{S}_i j$ has n_i rows. Thus, each diagonal block $\hat{S}_{ii}$ satisfies $0 < n_i - ||\hat{S}_{ii}||_F < \epsilon$, which implies $||\hat{S}_{ij}||_F \to 0$ for each off-diagonal block. Then we can characterize S's spectrum

Embedding	LLM	RAG	Acc	HARD Acc
SBERT	UnifiedQA	GEM-RAG	**52.14**%	**44.70**%
SBERT	UnifiedQA	RAPTOR	51.71%	43.30%
SBERT	UnifiedQA	Embed	51.04%	44.06%
SBERT	GPT3.5	GEM-RAG	**61.84**%	**51.60**%
SBERT	GPT3.5	RAPTOR	60.13%	50.32%
SBERT	GPT3.5	Embed	58.61%	47.25%
OpenAI	UnifiedQA	GEM-RAG	52.81%	44.83%
OpenAI	UnifiedQA	RAPTOR	**53.48**%	**44.96**%
OpenAI	UnifiedQA	Embed	52.14%	42.53%
OpenAI	GPT3.5	GEM-RAG	**63.37**%	**51.85**%
OpenAI	GPT3.5	RAPTOR	60.32%	50.96%
OpenAI	GPT3.5	Embed	60.32%	49.55%
OpenAI	GPT3.5	GEM-RAG (k-Means)	61.42%	50.83%

TABLE I
RESULTS ON THE QUALITY DEV DATASET FOR ALL EMBEDDING, LLM AND RAG PAIRS.

via the behavior of its eigenvalues, i.e., $0 < 1 - \lambda_i < \epsilon, \forall i \in \{1, 2, \ldots, k\}$ and $0 < |\lambda_i| < \epsilon, \forall i \in \{k+1, k+2, \ldots, n\}$.

These properties confer on S effective clustering capabilities, where text chunks within the same theme exhibit higher similarity and lower similarity across different themes.

Remark 2: Consider a sequence where each term is the ratio β_i of $\lambda_i, \forall i \geq 2$ to the largest eigenvalue λ_1. If there exists some index $d \geq 2$ such that β_d is close to 1, and $\beta_d - \beta_{d+1} > c$ where c indicates the cutoff that identifies the first significant gap between a pair of adjacent ratios, then d is the estimated number of essential themes in the document.

Remark 3: Let $\Lambda = \sum_i \lambda_i^2$. A higher Λ indicates more distinct clustering themes within the document; otherwise, the differentiation between themes becomes ambiguous.

2) Retrieval: Given a built GEM, our method is then ready to answer prompts/question about the given dataset. The process works as follows, given a prompt/question p, and some budget $\mathcal{B}$ of nodes to return, we first produce an embedding of the prompt to give $v_p = E(p)$. We then find, out of the entire graph, the utility question that has the highest `SIM`. Specifically, let $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ be the set of all utility questions, find $q^* = \underset{q \in \mathcal{Q}}{\operatorname{argmax}}\, \texttt{SIM}(v_p, E(q))$. Then, from the associated node with q^* we perform a best first search, to find the next nodes, up to $\mathcal{B}$ to include in the context set. Once we reach the budget we return the context. The full details can be seen in Algorithm 1.

The LLM is then given this context, followed by the question and prompted to answer.

C. Method Trade-Offs

While the robustness of our method leads to our improved results, it does come with some trade-offs that are important to discuss, particularly in terms of computational complexity, and potential costs.

First, generating utility questions via an LLM can become a significant cost, either computationally or monetarily ,depending on the number of nodes in the dataset, and the number of utility questions. Generating a graph with n nodes and q utility questions requires nq LLM calls. The graph building is all pre-computed, so in most cases this extra cost is okay, but it is worth noting.

Secondly, in order to generate higher level summary nodes we do eigendecomposition which has a complexity of $O(n^3)$ where n is the number of nodes. With a large number of nodes this complexity may require consideration.

IV. EXPERIMENTS

A. Setting

RAG methods focus on retrieving optimal small context windows for LLMs, which remains important despite growing context lengths in large models. Even with larger contexts, real-world datasets often exceed these limits, and processing massive amounts of tokens is costly. Moreover, larger contexts can lead to memory issues and hallucinations, making precise retrieval crucial for verification. While some models outperform retrieval methods using thousands of tokens, we focus on datasets and method that only use and return 400 tokens of context to compare fairly with recent work, as larger-scale retrieval remains more practical for real-world applications.

B. Datasets

a) QuALITY: The Question Answering with Long Input Texts, Yes! (QuALITY) dataset [14] contains 230 medium length passages (about 5000 tokens), for which each passage has associated multiple choice questions and ground truth answers for. Additionally, each set of questions has a subset of *HARD* questions which are particularly challenging. Specifically we use the 'dev' data split. For the ablations experiments we use the first 50 passages from this set, whereas for the main experiments in Table I we use the remaining 180.

b) Qasper: The Qasper data set [15] contains over 1500 academic NLP papers. Each paper has associated multiple choice questions and ground truth answers. We perform our experiments over the first 100 papers in the data set.

C. Baselines

We compare our method to the standard RAG method, of embedding each chunk of text, and embedding the given prompt, and finding the most similar chunks to the prompt, up to a specific budget. Additionally, we compare to a recent work that showed promising results, Recursive Abstractive Processing For Tree-Organized Retrieval (RAPTOR) method [16]. RAPTOR primarily aims to tackle the problem of producing hierarchies of nodes, that summarizes the text passage appropriately.

D. Evaluation

The QuALITY dataset has multiple choice questions with ground truth answers, for this dataset we calculate the accuracy and report the overall accuracy, as well as the accuracy on the HARD subset. The Qasper dataset has answer types that are either Abstractive, Extractive, Yes/NO or Answerable/Unanswerable. We use the F1 score to evaluate efficacy of the methods for this dataset.

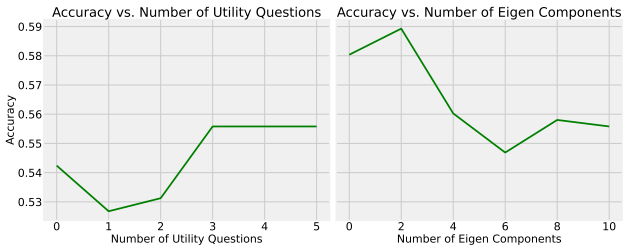

Fig. 2. Ablation study on the effect of number of utility questions and eigencomponents for a 50 passage subset of the QuALITY data set. The accuracy from more utility questions quickly becomes saturated, whereas the best number of eigencomponents is two. Intuitively the number of utility questions capture the amount of information per chunk, whereas the number of components covers the importance of summaries in the domain of questions.

E. Experimental Parameters

Our objective with these experiments is to evaluate the efficacy of our model across different text embedding models and LLMs. We use the SBERT [17], and the OpenAI text-embedding-ada-002 text embedding models. We also use the UnifiedQA [18] and GPT3.5 Turbo LLMs. For the QuALITY data set we consider all possible combinations of text encoders, LLMs and RAG methods. For the Qasper dataset we consider all embedding and RAG models but only using GPT-3.5 Turbo as the LLM.

For all experiments we use a chunk size of 100 tokens, and we allow 400 tokens of context, meaning four nodes of context. Even though GPT3.5 can support a much larger context, we aim to study the setting where only few nodes/chunks can be used, to better isolate the effectiveness of the RAG method in question, rather than the attention mechanism of the LLM. For the summarization method for GEM-RAG and RAPTOR, as well at the utility question method for GEM-RAG we use GPT3.5 Turbo as the LLM. All similarity measurements were done using cosine similarity. For the GEM-RAG method we use two eigencomponents, and five utility questions. In our ablation experiments we show the effect of varying both of these parameters. For the ablation experiments we evaluate the accuracy on the first 50 articles of the QuALITY data set, whereas for the main experiments we evaluate on the latter 180 articles.

F. Results

QuALITY Results Our model achieved the best overall accuracy and HARD subset performance in all settings, except when using the OpenAI embedding model and UnifiedQA LLM, with the largest gains seen when using GPT-3.5 Turbo, as can be seen in Table I. Switching from spectral clustering to k-Means led to a performance drop, highlighting the effectiveness of spectral graph analysis.

Qasper Results The results from our experimentation can be seen in Table II. In this setting we only use GPT3.5Turbo as the LLM and consider using both SBERT and the OpenAPI text embedding models. We see that our method performs best for the OpenAI's text-embedding-ada-002 embedding model, and gives the best over all score. However, it performs behind the standard RAG method when considering SBERT.

Embedding	LLM	RAG	F1
SBERT	GPT3.5	GEM-RAG	18.53%
SBERT	GPT3.5	RAPTOR	18.51%
SBERT	GPT3.5	Embed	**19.24**%
OpenAI	GPT3.5	GEM-RAG	**20.13**%
OpenAI	GPT3.5	RAPTOR	18.13%
OpenAI	GPT3.5	Embed	19.07%

TABLE II
RESULTS ON THE QASPER DATA SET FOR ALL EMBEDDING, LLM AND RAG PAIRS.

GEM For Data Visualization We would like to stress that while GEM has been formulated to be primarily used as a method for RAG, the produced GEM is a standalone object that can be used with any LLM to do QA work, as well as a tool to visualize and organize data. We provide an example visualization for a single story in the QuALITY data set at the following url: https://detailed-swan.static.domains/GEM.html.

Ablation Study We tested the tunable hyperparameters of our model—eigencomponents and utility questions—through two ablation experiments as seen in Figure 2. In the first, we held the number of components constant at ten and varied the number of utility questions. Accuracy increased up to three questions before saturating, while zero utility questions meant direct comparison of text embeddings. In the second, we held utility questions at five and varied eigencomponents, finding the best performance with two components.

Intuitively, eigencomponents capture the importance of high-level themes, and utility questions capture the detail in a chunk. These parameters should be tuned for specific tasks; we used a holdout set of 50 QuALITY passages to guide hyperparameter selection for the remaining 180 passages.

V. CONCLUSION

We developed GEM-RAG, a method inspired by human cognition that tags memory chunks by their utility and relations, forming a weighted graph and extracting "eigenthemes" through eigendecomposition. Our method outperforms standard baselines across multiple embeddings and LLMs, and the produced GEM can function independently for searching and visualizing textual data. We believe GEM-RAG can enhance LLMs, enabling them to handle vast histories or niche datasets, and extend to multimodal models for richer retrieval capabilities.

VI. ACKNOWLEDGEMENTS

This project is partially supported by the National Science Foundation (NSF); the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a program of Schmidt Sciences, LLC; the National Institute of Food and Agriculture (US-DA/NIFA); the Air Force Office of Scientific Research) (AFOSR), and Toyota Research Institute (TRI).

REFERENCES

[1] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with gpt-4," 2023.

[2] F. Petroni, T. RocktÃ€schel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" 2019. [Online]. Available: https://arxiv.org/abs/1909.01066v2

[3] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *CoRR*, vol. abs/1911.12543, 2019. [Online]. Available: http://arxiv.org/abs/1911.12543

[4] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with gpt-4," 2023.

[5] A. Raventós, M. Paul, F. Chen, and S. Ganguli, "Pretraining task diversity and the emergence of non-bayesian in-context learning for regression," 2023.

[6] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *CoRR*, vol. abs/2005.11401, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

[7] ——, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *CoRR*, vol. abs/2005.11401, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

[8] A. W. Yu, D. Dohan, M. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," *CoRR*, vol. abs/1804.09541, 2018. [Online]. Available: http://arxiv.org/abs/1804.09541

[9] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham, "In-context retrieval-augmented language models," 2023.

[10] E. Akyürek, T. Bolukbasi, F. Liu, B. Xiong, I. Tenney, J. Andreas, and K. Guu, "Towards tracing factual knowledge in language models back to the training data," 2022.

[11] B. Goldstein, *Cognitive Psychology: Connecting Mind, Research, and Everyday Experience, 5th Edition*. Boston, Massachusetts: Cengage Learning, 2018.

[12] I. Henry L. Roediger and J. D. Karpicke, "Test-enhanced learning: Taking memory tests improves long-term retention," *Psychological Science*, vol. 17, no. 3, pp. 249–255, 2006, pMID: 16507066. [Online]. Available: https://doi.org/10.1111/j.1467-9280.2006.01693.x

[13] M. Bolognesi and G. Steen, "Editors' introduction: Abstract concepts: Structure, processing, and modeling," *Topics in Cognitive Science*, vol. 10, no. 3, pp. 490–500, 2018.

[14] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, and S. R. Bowman, "Quality: Question answering with long input texts, yes!" 2022.

[15] P. Dasigi, K. Lo, I. Beltagy, A. Cohan, N. A. Smith, and M. Gardner, "A dataset of information-seeking questions and answers anchored in research papers," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Online: Association for Computational Linguistics, Jun. 2021, pp. 4599–4610. [Online]. Available: https://aclanthology.org/2021.naacl-main.365

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

[17] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[18] D. Khashabi, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi, "Unifiedqa: Crossing format boundaries with a single QA system," *CoRR*, vol. abs/2005.00700, 2020. [Online]. Available: https://arxiv.org/abs/2005.00700

[19] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[20] ——, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[21] OpenAI, "Gpt-4 technical report," 2023.

[22] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 5418–5426. [Online]. Available: https://aclanthology.org/2020.emnlp-main.437

[23] 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," 2023.

[24] S. Sun, K. Krishna, A. Mattarella-Micke, and M. Iyyer, "Do long-range language models actually use long-range context?" in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 807–822. [Online]. Available: https://aclanthology.org/2021.emnlp-main.62

[25] U. Shaham, M. Ivgi, A. Efrat, J. Berant, and O. Levy, "Zeroscrolls: A zero-shot benchmark for long text understanding," 2023. [Online]. Available: https://arxiv.org/abs/2305.14196

[26] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Atlas: Few-shot learning with retrieval augmented language models," 2022.

[27] ——, "Atlas: Few-shot learning with retrieval augmented language models," 2022.

[28] S. Min, K. Lee, M.-W. Chang, K. Toutanova, and H. Hajishirzi, "Joint passage ranking for diverse multi-answer retrieval," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6997–7008. [Online]. Available: https://aclanthology.org/2021.emnlp-main.560

[29] M. G. Arivazhagan, L. Liu, P. Qi, X. Chen, W. Y. Wang, and Z. Huang, "Hybrid hierarchical retrieval for open-domain question answering," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 10 680–10 689. [Online]. Available: https://aclanthology.org/2023.findings-acl.679

[30] Y. Liu, K. Hashimoto, Y. Zhou, S. Yavuz, C. Xiong, and P. Yu, "Dense hierarchical retrieval for open-domain question answering," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 188–200. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.19

[31] J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike, and P. Christiano, "Recursively summarizing books with human feedback," 2021.

[32] B. Newman, L. Soldaini, R. Fok, A. Cohan, and K. Lo, "A question answering framework for decontextualizing user-facing snippets from scientific documents," 2023.

[33] T. Gao, H. Yen, J. Yu, and D. Chen, "Enabling large language models to generate text with citations," 2023.

[34] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550

[35] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[36] W. Li, W.-K. Ng, Y. Liu, and K.-L. Ong, "Enhancing the effectiveness of clustering with spectra analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 887–902, 2007.

[37] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *J. Res. Natl. Bur. Stand. B*, vol. 45, pp. 255–282, 1950.