Journal of Engineering Science *Vol. XXXI, no. 2 (2024), pp. 48 - 59*
Fascicle *Electronics and Computer Science* *ISSN 2587-3474*
Topic *Computers and Information Technology* *eISSN 2587-3482*

https://doi.org/10.52326/jes.utm.2024.31(2).05
UDC 004.8:81'322

RETRIEVAL-AUGMENTED GENERATION USING DOMAIN-SPECIFIC TEXT: A PILOT STUDY

Victor Iapăscurtă [1, 2] *, ORCID: 0000-0002-4540-7045,
Sergey Kronin [3], ORCID: 0009-0001-6164-6708,
Ion Fiodorov [1], ORCID: 0000-0003-0938-3442

[1] *Technical University of Moldova, 168 Stefan cel Mare Blvd., Chisinau, Republic of Moldova*
[2] *N. Testemitanu State University of Medicine and Pharmacy, 165 Stefan cel Mare Blvd., Chisinau, Republic of Moldova*
[3] *Open Temperological Institute, Moscow, Russian Federation*
* Corresponding author: Victor Iapăscurtă, *victor.iapascurta@doctorat.utm.md*

Received: 05. 25. 2024
Accepted: 06. 30. 2024

Abstract. The natural language processing (NLP) field has witnessed remarkable advancements with the advent of large language models (LLMs) like GPT, Gemini, Claude, etc. These models are trained on vast amounts of text data, allowing them to generate human-like responses for various tasks. However, despite their impressive capabilities, LLMs have limitations in their ability to incorporate and reason over external knowledge that is not in their training data. This limitation of LLMs is particularly evident in the case of specific domain knowledge. This situation has given rise to the concept of retrieval augmented generation (RAG), an approach that combines the generative power of LLMs with the ability to retrieve and integrate relevant information from external knowledge sources. This research attempts to use RAG as a module in an application designed to answer questions concerning a specific domain, namely social philosophy/philosophy of management, using a published book from the respective domain as an external source. The paper analyzes the mentioned application output, draws conclusions, and traces future directions to improve the accuracy of the output.

Keywords: *natural language processing; retrieval-augmented generation; large language models; domain-specific knowledge; domain-specific text.*

Rezumat: Domeniul procesării limbajului natural (NLP) a fost martorul unor progrese remarcabile odată cu apariția modelelor de limbaj mari (LLM) precum GPT, Gemini, PaLM, Claude și altele. Aceste modele sunt antrenate pe cantități mari de date text, permițându-le să genereze răspunsuri asemănătoare omului pentru diferite sarcini. Cu toate acestea, în ciuda capacităților lor impresionante, LLM-urile au limitări în capacitatea lor de a încorpora și raționa cunoștințele externe care nu sunt în datele lor de formare. Această limitare a LLM-urilor este deosebit de evidentă în cazul cunoștințelor specifice domeniului. Această situație a dat naștere conceptului de retrieval augmented generation (RAG), o abordare care combină puterea generativă a LLM-urilor cu capacitatea de a prelua și integra informații relevante

din surse externe de cunoștințe. Această cercetare încearcă să utilizeze RAG ca modul într-o aplicație menită să răspundă la întrebări referitoare la un anumit domeniu, și anume filosofia socială/filosofia managementului, folosind ca sursă externă o carte publicată din domeniul respectiv. Lucrarea analizează rezultatul aplicației menționate, trage concluzii și urmărește direcțiile viitoare pentru a îmbunătăți acuratețea rezultatelor.

Cuvinte cheie: *procesarea limbajului natural; generare augmentată de recuperare; modele mari de limbaj; cunoștințe specifice domeniului; text specific domeniului.*

1. Introduction

The field of natural language processing (NLP) has experienced remarkable advancements in recent years, largely due to the emergence of large language models (LLMs) such as GPT, Gemini, PaLM, Claude, and others. These LLMs have been trained on extensive amounts of text data, enabling them to generate responses that closely resemble human-like behavior across various tasks. However, despite their impressive capabilities, LLMs have certain limitations when it comes to incorporating and reasoning over external knowledge that is not included in their training data [1].

Due to this limitation, a new concept known as retrieval-augmented generation (RAG) has emerged [2, 3]. This innovative approach combines LLMs' generative power with the ability to retrieve and integrate relevant information from external knowledge sources, thereby addressing the aforementioned limitation and enhancing NLP systems' overall performance and versatility. According to a recent survey [4], the development of RAG technology has shown significant progress. It has evolved alongside the emergence of the Transformer architecture, aiming to enhance language models by incorporating additional knowledge through Pre-Training Models (PTM). Initially, researchers focused on improving pre-training techniques [3, 5, 6]. Subsequently, the introduction of ChatGPT demonstrated the impressive capabilities of LLMs in in-context learning (ICL). This led to a shift in RAG research towards enhancing the information available to LLMs, enabling them to handle more complex and knowledge-intensive tasks during inference. As a result, significant advancements in RAG studies have been achieved.

Three stages of the RAG research paradigm are described [4]:

1. Naive RAG is the earliest methodology primarily comprising three steps: (a) Indexing refers to the process of creating an organized and searchable list or database of information. The process involves dividing documents into segments, converting them into numerical representations, and then storing them in a database specifically designed for vectors; (b) Information retrieval: retrieval of the *k* most relevant chunks based on semantic similarity to the question; (c) Generation.
2. Advanced RAG implements targeted enhancements to address the constraints of the Naive RAG model. It utilizes strategies both before and after the retrieval process to improve retrieval quality. To address the problems related to indexing, Advanced RAG enhances its indexing methods by employing a sliding window approach, precise segmentation, and metadata integration. Furthermore, it integrates multiple optimization techniques to streamline retrieval [7].
3. Modular RAG is a modular system that uses an architecture that surpasses the previous two RAG paradigms by providing improved adaptability and versatility. The system employs various methods to enhance its elements, including adding a search module for conducting similarity searches and improving the retriever

through fine-tuning. Specific challenges have been addressed through the implementation of restructured RAG modules [8] and rearranged RAG pipelines [9]. Adopting a modular RAG approach is increasingly common, facilitating both sequential processing and integrated end-to-end training across its components. Modular RAG, although unique in its characteristics, is based on the fundamental principles of Advanced and Naive RAG.

This research uses elements from all three stages described above. It includes creating a question-answering application using a set of free/open-source tools. The database/vector store is based on philosophical text. The answers generated by the application are for a set of test questions present in the original text, and they are analyzed for accuracy/consistency.

The paper's main purpose is to describe the development and evaluation of a question-answering application using retrieval-augmented generation techniques.

The hypotheses being tested are:

- Whether a RAG system can be created using freely available open-source tools to answer questions based on a philosophical text database.
- How accurate and consistent are the answers generated by the RAG system for test questions present in the original text?

The main conclusion is that RAG technologies can be adapted to solve domain-specific tasks and extrapolated to similar applications with adjustments. The paper provides an example of implementing this.

2. Materials and Methods

2.1 An overview of the RAG to be created

LLMs can reason about wide-ranging topics, but their knowledge is limited to the public data up to the specific time they were trained. To build AI applications that can reason about private data or data introduced after a model's cutoff date, there is a need to augment the knowledge of the model with the specific information it needs. The process of bringing and inserting the appropriate information into the model prompt is known as Retrieval Augmented Generation (RAG).

RAG dynamically retrieves relevant context from external sources, integrates it with user queries, and feeds the retrieval-augmented prompt to an LLM for generating responses.

To build the system, one must first set up the vector database with the external data by chunking the text, embedding the chunks, and loading them into the vector database. Once this is complete, the following steps in real-time to generate the answer for the user should be orchestrated:

- *Retrieve*. Embedding the user query into the vector space to retrieve relevant context from an external knowledge source.
- *Augment*. Integrating the user query and the retrieved context into a prompt template.
- *Generate*. Feeding the retrieval-augmented prompt to the LLM for the final response generation.

The general architecture of a RAG system is presented in Figure 1.

Respectively, the main components of RAG are:

(a) Vector database: a vector DB stores vector embeddings of the external source documents.

(b) LLM: language models such as OpenAI or others are the foundation for generating responses.

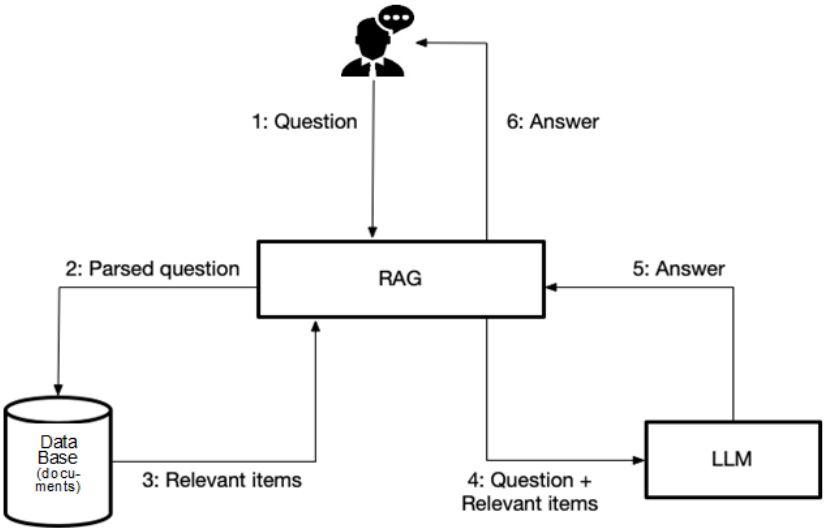

Figure 1. The general architecture of the RAG.

(c) Embedding model: often derived from the LLM, the embedding model plays a crucial role in creating meaningful text representation.
(d) Orchestration tool: An orchestration tool like Langchain or Llamaindex manages the workflow and interactions between components.

Creating a RAG system can start with building a vector store. This step can be implemented using different tools, but the result will be the vector database. Unlike traditional relational databases that primarily handle structured data, vector databases are optimized for managing unstructured and semi-structured data, such as images, text, and tables, represented as numerical vectors in a high-dimensional space. These vectors capture the inherent structure and relationships within the data, enabling sophisticated similarity search, recommendation, and data analysis tasks. Figure 2 denotes the main steps of this process.

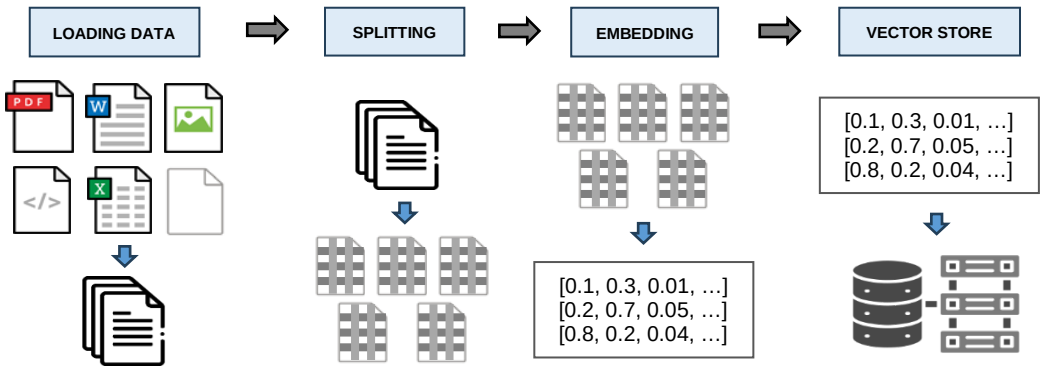

Figure 2. The RAG: creating the vector store.

First, the documents in various formats should be loaded and then split/chunked. Chunking involves breaking down texts into smaller, manageable pieces. Each chunk becomes a unit of information that is vectorized and stored in a database, fundamentally shaping the efficiency and effectiveness of natural language processing tasks.

Embeddings usually refer to dense, continuous vectors representing text in a high-dimensional space. These vectors serve as coordinates in a semantic space, capturing the relationships and meanings between words. In the context of LLMs, embeddings play an important role in retrieving the right context for RAG.

2.2. Word embedding techniques

Embedding techniques, including natural language processing, are widely used in various fields to represent data in a lower-dimensional vector space. These techniques aim to capture the underlying semantic structure and relationships between data points. Three popular embedding techniques are *"l_2," "ip,"* and "*cosine*," each with its own mathematical formalism behind it.

The l_2 embedding technique, or Euclidean distance, calculates the distance between two vectors in the Euclidean space. Given two vectors, ***a*** and ***b***, of dimension n, the l_2 distance is computed using the following formula:

$$l_2(a,b) = \sqrt{\sum(a_i - b_i)^2}, \quad (1)$$

where: a_i and b_i represent the i^{th} elements of vectors ***a*** and ***b***, respectively. The l_2 embedding technique aims to minimize this distance, which indicates the similarity between two vectors. Smaller l_2 distances imply a higher degree of similarity.

The *ip* embedding technique, or inner product, computes the dot product between two vectors. Given two vectors, ***a*** and ***b***, of dimension n, the *ip* embedding is calculated as follows:

$$ip(a,b) = \sum(a_i * b_i). \quad (2)$$

In this formula, a_i and b_i represent the i^{th} elements of vectors ***a*** and ***b***, respectively. The *ip* embedding technique measures the similarity between two vectors based on their angle. Higher dot products indicate vectors that are more aligned or similar.

The *cosine* embedding technique calculates the cosine of the angle between two vectors. Given two vectors, ***a*** and ***b***, of dimension n, the *cosine* embedding is computed as:

$$\cos(a,b) = \frac{ip(a,b)}{\|a\| * \|b\|}, \quad (3)$$

where: $ip(a,b)$ represents the inner product between vectors ***a*** and ***b***, and $\|a\|$ and $\|b\|$ represent the Euclidean norms of vectors ***a*** and ***b***, respectively. The *cosine* embedding technique ranges between -1 and 1, where 1 indicates perfect similarity, 0 indicates orthogonality, and -1 indicates perfect dissimilarity.

In summary, the l_2 embedding technique measures the Euclidean distance between two vectors, the *ip* embedding technique computes the dot product, and the *cosine* embedding technique calculates the cosine of the angle between two vectors. These embedding techniques provide different perspectives on representing the similarity or dissimilarity between vectors and are widely used in various applications.

The techniques used in this research are usually based on l_2 or *cosine* similarity.

2.3. Assembling the RAG components

Once all the components are ready, they are assembled, and the data flow in an RAG system looks like in Figure 3.

2.4. Tools used throughout this research

Most of the tools/approaches used in this research are open source/free and include:

a) **LangChain** is a framework for developing applications powered by large language models (LLMs) [10]. LangChain enables building applications that connect external sources of data and computation to LLMs. LangChain has a number of components designed to help build Q&A applications and RAG applications more generally.

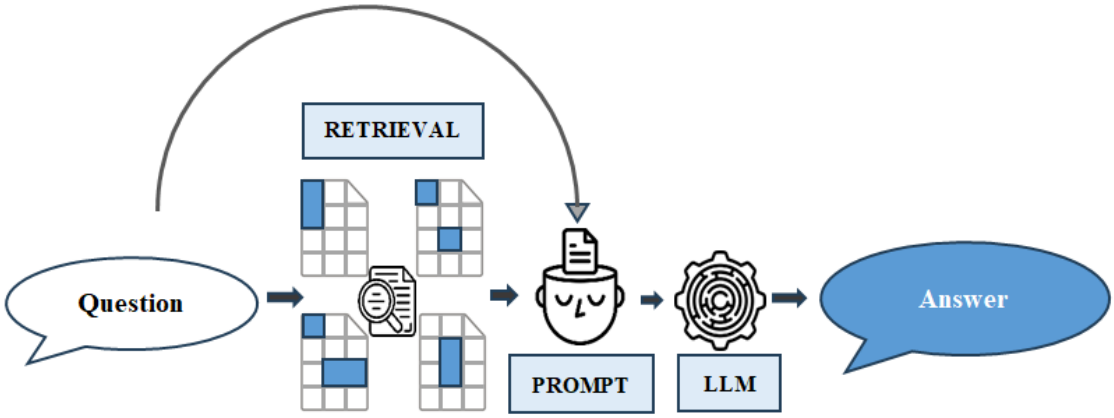

Figure 3. The RAG: main steps of the data flow.

b) **Chroma DB** is a database for building AI applications with embeddings [11]. This tool uses valid embedding options: l_2, *ip*, or *cosine*. The default is l_2, which is the squared L_2 norm. By default, Chroma uses the Sentence Transformers all-MiniLM-L6-v2 model to create embeddings. This embedding model can create sentence and document embeddings that can be used for various tasks. Chroma provides lightweight wrappers around popular embedding providers, making using them in applications easy.

c) **LlamaIndex** is a data framework for LLM-based applications [12]. It uses cosine similarity when comparing embeddings by default. It also uses the OpenAI GPT-3.5-turbo model for text generation and the text-embedding-ada-002 model for retrieval and embeddings.

d) **GPT-3.5 Turbo models** [13] can understand and generate natural language or code. They have been optimized for chat using the Chat Completions API but also work well for non-chat tasks. This research uses the latest GPT-3.5 Turbo model (0125) with higher accuracy at responding in requested formats available when the work was performed.

e) **Uniform Manifold Approximation and Projection (UMAP)** is a dimension reduction technique that can be used for visualization similarly to t-SNE but also for general non-linear dimension reduction [14]. The algorithm is founded on three assumptions about the data: (1) The data is uniformly distributed on the Riemannian manifold; (2) The Riemannian metric is locally constant (or can be approximated as such); (3) The manifold is locally connected. From these assumptions it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low-dimensional data projection with the closest possible equivalent fuzzy topological structure [14, 15].

f) **Triad tool:** In this research, the "Triad approach" proposed by TruEra [16] is used to evaluate the RAG results. It includes three aspects/evaluations: context relevance, groundedness, and answer relevance. According to [16], satisfactory evaluations on each aspect are expected to provide confidence that the LLM application is free from hallucination. Figure 4 illustrates the architecture of this approach.

The text used in this research is philosophical, coming from the field of social philosophy, namely the philosophy of management. It is from a published book [17] in the respective domain and represents the external source.

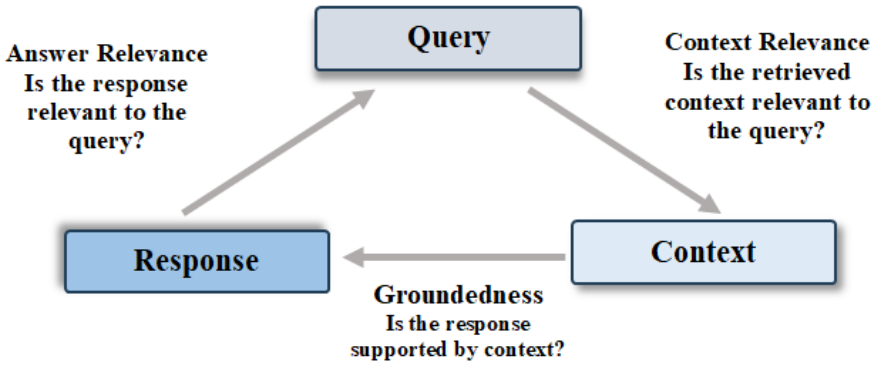

Figure 4. The architecture of the Triad approach [10].

The book includes fifty questions for self-evaluation. The original text is in Russian. For the purpose of the research, the text was translated into English. The answers provided by the RAG systems were also translated back into Russian for analysis by a native Russian-speaking human expert.

3. Results

3.1. General description of the RAG systems built in this study

There had been created several RAG systems based on the same external text source using combinations of tools described above. Three of them are analyzed in this article:

a) a system based on the Sentence Window Retrieval using LlamaIndex tools. This approach gives an LLM better context by retrieving not just the most relevant sentence but the window of sentences that occur before and after it in the document. The embedding model used here is the HuggingFace BAAI/bge-small-en-v1.5;

b) a CromaDB-based system: as with other approaches the text from the book translated into English was extracted with PdfReader, split in chunks using LangChain text splitter, particularly Sentence Transformer Text Splitter with the dimension of a chunk equal to 256 characters, no overlap, embedding the chunks using the Sentence Transformer Embedding Model (all-MiniLM-L6-v2) with the generation of multidimensional dense vectors and collecting the resulting vectors into a vector store (i.e., Chroma database);

c) a more sophisticated system based on Auto-merging Retrieval, which organizes the document into a tree-like structure where each parent node's text is divided among its child nodes. When meta-child nodes are identified as relevant to the user question, then the entire text of the parent node is provided as context for the LLM. This version is also heavily based on LlamaIndex tools using the same model as a) for embeddings and the BAAI/bge-reranker-base model for reranking.

The final steps for all three systems included retrieving the information from the DB based on text similarity (e.g., l_2 similarity for a) and c), and *cosine* for b)) with the user query and passing the retrieved information to LLM using an API with a system prompt requiring the LLM to use only the retrieved information. The programming language used throughout this research is Python [18].

3.2. Retrieval details

Since the final answer of an RAG system to a user query greatly depends on the quality of the original text embeddings, it is important to evaluate the quality of these embeddings. Within this research UMAP is used for this. UMAP makes it possible to project the

multidimensional vector space to two or three dimensions, which are easy to visualize and, at the same time, preserve the structure of the data (in Figures 5 and 6, these are represented by grey dots cloud). Analyzing Figure 5, it can be observed that the retrieved data points (denoted by green circles) are close to the query, implying semantic similarity.

a) What is Temperology from the point of view of systems approach?

b) Are there any contradictions between Temperology and Systems theory?

Figure 5. The closeness (denoting similarity) of the query (red cross) and retrieved data (green circles) for two queries/questions (based on our own research data).

In the case of a query (e.g., How does Mars influence the well-being of humans?) totally irrelevant to the original text, the retrieved chunks are located further away in the vector space, Figure 6.

How does Mars influence the well-being of humans?

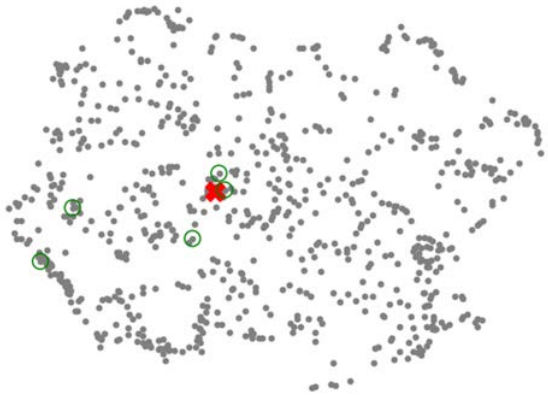

Figure 6. The query (red cross) and the spread of retrieved data in the case of an irrelevant query (based on our own research data).

This geometric visualization can help develop intuitions concerning the analyzed data and select the most appropriate embedding techniques to provide the best final results.

3.2. Evaluation results

The Triad approach is used to evaluate answers provided by the RAG systems built in this research. The aspects subjected to evaluation included (a) context relevance, (b) groundedness, and (c) answer relevance. Figures 7 to 9 illustrate the results for one of the three RAG systems developed in this research.

Context relevance (Figure 7) verifies the retrieval quality to ensure that each chunk of context is relevant to the input query. This is critical because this context will be used by the LLM to form an answer, so any irrelevant information in the context could be weaved into a hallucination.

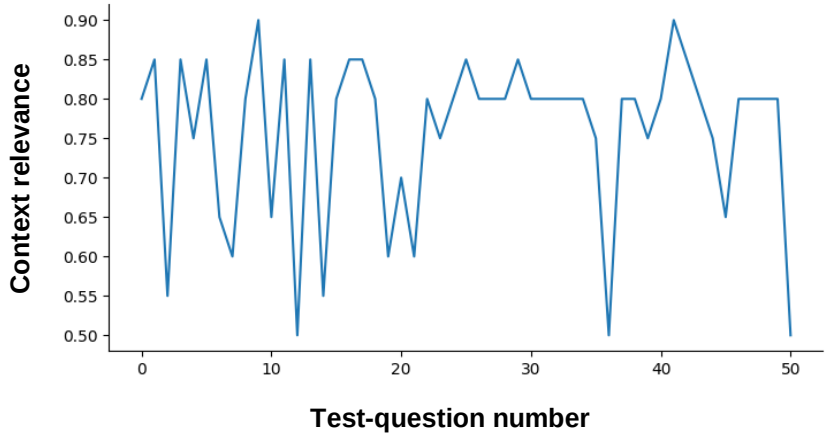

Figure 7. Context relevance with Chain of Thoughts (CoT) reasons approach.

Groundedness (Figure 8) represents how close the answer provided by an LLM is to the facts provided by retrieval. After the context is retrieved, an LLM forms it into an answer. LLMs often stray from the facts provided, exaggerating or expanding to a correct-sounding answer.

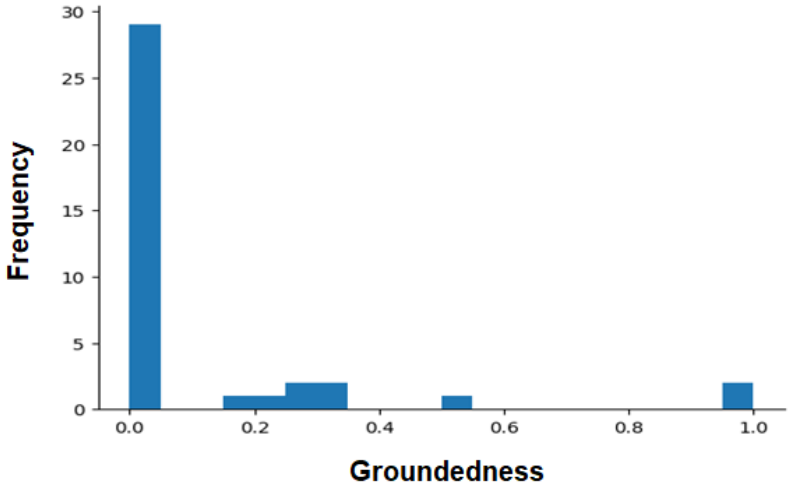

Figure 8. Groundedness with Chain of Thoughts (CoT) reasons approach.

Answer relevance (Figure 9) represents how useful is the answer provided by LLM.

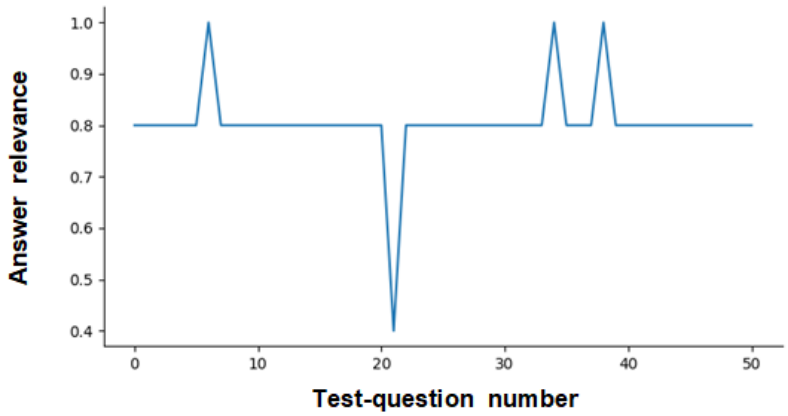

Figure 9. Answer relevance.

Evaluation results for all three RAG systems created in this study are summarized in Table 1.

Table 1

Results of using three RAG approaches to the same external data set

Application	Answer relevance	Context relevance	Groundedness	Latency (seconds)	Cost, ($ US)
RAG based on Sentence Window Retrieval	0.804	0.760	0.107	6.10	0.003
ChromaDB-based RAG	0.897	0.865	0.240	3.10	0.002
RAG based on Auto-merging approach	0.846	0.529	0.621	37.84	0.006

Note: RAG – Retrieval augmented generation

Along with the performance parameters described previously (i.e., context relevance, groundedness, and answer relevance), the latency and cost of computation are also presented in Table 1, which are important when building and using such applications.

4. Discussion

Overall, the results show that the ChromaDB-based RAG system performed best regarding answer relevance, context relevance, and groundedness. This suggests that the approach of embedding text chunks into vectors and storing them in a ChromaDB followed by cosine similarity retrieval worked better than the other approaches for this dataset.

The Sentence Window Retrieval approach showed reasonably good results for answer and context relevance but very low groundedness. This indicates that while the retrieved context was relevant, the LLM drifted far from the facts in the context while formulating the answer.

The Auto-merging approach had the lowest context relevance score. This suggests that organizing the text into a tree structure and retrieving parent nodes did not accurately capture the relevant context compared to the other approaches. However, it had the highest groundedness score.

The ChromaDB approach was the fastest and most cost-effective regarding latency and cost, making it the best-performing overall system.

Some possibilities to improve the results would include:

a) Experiment with different embedding models and similarity measures for the Sentence Window approach to improve groundedness.
b) Trying refining the tree structuring and retrieval algorithm in the Auto-merging approach to capture more relevant context.
c) Evaluating larger and more complex datasets to test the scalability of the approaches
d) Adding explanatory capabilities to the LLM responses to improve transparency and trustworthiness.
e) Combine ideas from the best performing ChromaDB approach and Auto-merging approach with its high groundedness for a hybrid system.

Although the answer's relevance is being evaluated, in the case of the domain-specific test, the “ground truth” evaluated by a human expert might be of ultimate importance. Table

2 showcases the answer to one of the evaluation questions with an unambiguous answer (based on the original text).

Table 2

The answers to one of the evaluation questions provide by RAGs and CPT (without RAG)

Question (query): *"How many elements are there in the temperological structure of personality?"*	**Answers provided by different approaches**
RAG 1 (based on Sentence Window Retrieval / Auto-merging approach)	*"Three elements."*
RAG 2 (based on ChromaDB)	*"The temperological structure of personality consists of three elements - the mind, reason, and self of a person."*
ChatGPT (without RAG)	*"The temperological structure of personality typically consists of three main elements, which are emotionality, activity, and sociability."*

Note: RAG - Retrieval augmented generation

Although the answers provided by RAG 1 and RAG 2 are both correct, they differ in details, and only the human expert can probably decide on the most appropriate/the best answer.

5. Conclusions

In conclusion, the ChromaDB approach showed the best balance of performance and efficiency, but all systems could be enhanced further through additional experimentation and modeling efforts.

Although the main goal of this research is to evaluate the possibilities of RAG using available approaches and tools (e.g., context relevance, groundedness, relevance, etc.), the answers to the fifty questions from the book mentioned above were presented to a human expert in the field the text comes from, who appreciated the level of the answers as "acceptable." Since the understandability and trustworthiness of the answers may bear specific domain nuances, the future steps are to include an evaluation by human experts and use new approaches and tools in the RAG family, which are currently being actively developed.

A multi-pronged empirical and human-centered evaluation approach that considers diverse use cases seems needed to comprehensively understand the strengths, weaknesses, and opportunities for improving such research.

Acknowledgments: The authors would like to thank Andrew Ng from DeepLearning.AI and his partners from LangChain, LlamaIndex, ChromaDB, and TruLens for the series of short courses that greatly inspired and helped this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kandpal, N.; Deng, H.; Roberts, A.; Wallace, E.; Raffel, C. Large language models struggle to learn long-tail knowledge. In: *International Conference on Machine Learning*, *PMLR*, 2023, pp. 15696– 15707.
2. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; Chang, M-W. REALM: Retrieval-augmented language model pre-training. ArXiv, abs/2002.08909, 2020.

3. Lewis, P.; E. Perez, E.; Piktus, A,; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W-t.; Rocktäschel, T.; Riedel, S.; Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems* 2020, 33, pp. 9459–9474.
4. Gaoa, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, M.; Wang, H. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997v5.
5. Arora, D.; Kini, A.; Chowdhury, S.R.; Natarajan, N.; Sinha, G.; Sharma, A. Gar-meets-rag paradigm for zero-shot information retrieval. arXiv:2310.20158v1.
6. Borgeaud, A.; Mensch, J.; Hoffmann, T.; Cai, T.; Rutherford, E.; Millican, K.; Bm, G.; Driessche, V.D.; Lespiau, J-B.; Damoc, B.; Clark, A.; De Las Casas, D.; Guy, A.; Menick, J.; Ring, R.; Hennigan, T.; Huang, S.; Maggiore, L.; Jones, C.; Cassirer, A.; Brock, A.; Paganini, M.; Irving, G.; Vinyals, O.; Osindero, S.; Simonyan, K.; Rae, J.; Elsen, E.; Sifre, L. Improving language models by retrieving from trillions of tokens. In International conference on machine learning. In: *International Conference on Machine Learning*, *PMLR*, 2022, pp. 2206–2240.
7. Ilin, I. Advanced rag techniques: an illustrated overview. 2023. Available online: https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6 (accessed on 24.04.2024).
8. Yu, W.; Iter, D.; Wang, S.; Xu, Y.; Ju, M.; Sanyal, S.; Zhu, C.; Zeng, M.; Jiang, M. Generate rather than retrieve: Large language models are strong context generators. arXiv:2209.10063v3.
9. Shao, Z.; Gong, Y.; Shen, Y.; Huang, M.; Duan, N.; Chen, W. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. arXiv:2305.15294v2.
10. Applications that can reason. Powered by LangChain. Available online: https://www.langchain.com/ (accessed on 24.04.2024).
11. Chroma. The AI-native open-source embedding database. Available online: https://www.trychroma.com/ (accessed on 24.04.2024).
12. LlamaIndex. Available online: https://www.llamaindex.ai/ (accessed on 24.04.2024).
13. GPT-3.5 Turbo Models. Available online: https://platform.openai.com/docs/models/gpt-3-5-turbo (accessed on 24.04.2024).
14. Uniform Manifold Approximation and Projection (UMAP). Available online: https://umap-learn.readthedocs.io/en/latest/ (accessed on 24.04.2024).
15. McInnes, L.; Healy, J.; Saul, N.; Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software* 2018, 3(29), p. 861. https://doi.org/10.21105/joss.00861.
16. TruEra: Building an LLM app? Evaluate, debug, and monitor. At scale. Available online: https://truera.com/ (accessed on 24.04.2024).
17. Kronin, S. Temperology is … Open Tempeprological Institute, Moscow, 2019, 273 p. [in Russian]. ISBN 978-5-9902317-2-6.
18. Van Rossum, G.; Drake, F.L. Python 3 Reference Manual. Scotts Valley, CreateSpace, CA, 2009, 242p. ISBN:978-1-4414-1269-0

Citation: Iapăscurtă, V.; Kronin, S.; Fiodorov, I. Retrieval-augmented generation using domain-specific text: a pilot study. *Journal of Engineering Science* 2024, XXXI (2), pp. 48-59. https://doi.org/10.52326/jes.utm.2024.31(2).05.

Submission of manuscripts: jes@meridian.utm.md