Using LLMs for Analyzing Pancreatic Cancer Literature - A RAG Approach

Mateja Volčanšek[a], Milica Perišić Nanut[b,1], Biljana Mileva Boshkovska[a,b,1]

[a] *Faculty of Information Studies in Novo Mesto*
[b] *Jožef Stefan Institute*

Abstract

There is a serious overflow in biomedical publications, making it increasingly difficult for researchers to keep up with the literature. An effective, scalable system is needed that can utilize large language models and retrieval augmented generation, and enable researchers to directly ask complex questions and receive accurate, detailed answers based on current scientific evidence. Initially, we built a basic RAG (Retrieval Augmented Generation) system, focused on textual data. We then improved the system by refining key parameters including chunk size, chunk overlap, and the number of top retrieved contexts. We added an upgrade to display output context and references. Expert evaluation shows RAG outperforms ChatGPT and DeepSeek by delivering more accurate references and fewer hallucinations.

Keywords:
Retrieval Augmented Generation (RAG), Large Language Models (LLMs), Natural Language Processing (NLP), Text Generation

1. Introduction

For more than a decade, there has been a serious overflow of various types in biomedical science publications [27]. Types of overflow range from word count, page count, to the amount of articles produced, and consequently, megajournals were introduced - one even published 23020 papers in 2016. Moreover, the quantity can threaten the quality and objectivity of peer reviews because more publications require more reviewers. The increased amount also contributes to a noisier scientific landscape, making it harder to tell what is influential and relevant knowledge [26]. The cause lies in the current system that values producing more data and publishing more papers over creating meaningful, reproducible knowledge. If we want to keep up with the publications, due to sheer volume, we are forced to read papers faster, which impairs our capacity to properly evaluate publications [6].

PubMed is a free search engine tool, containing more than 38 million citations and abstracts of biomedical literature. It does not include full text. However, if there are sources available, there are also links to full texts on publishers' sites. It is developed and maintained by the U.S. National Library of Medicine (NLM) and includes several literature resources such as MEDLINE, PubMedCentral (PMC) and Bookshelf.

Large language models are systems designed for generation and interaction using natural language. They are built on the transformer architecture that uses mechanisms like tokenization, attention, and position encoding to process and generate text. They are characterized by their massive scale of parameters and datasets. This massive scale enables them to perform extremely well, but it comes at a cost of slow training and inference, hardware requirements, hallucinations (generating plausible but incorrect information) and bias inherited from training data. Recently, a lot of concerns are also raised over energy consumption and environmental impact [21].

Despite significant advancements in the field, the models remain vulnerable to hallucinations or simply produce results with lower accuracy than desired. To ensure higher accuracy, reduce hallucinations, and ensure models remain focused on domain-specific data, which is often missing from their original training sets, we can employ Retrieval-Augmented Generation (RAG). RAG is the retrieval of relevant information from an additional database before generating an answer with LLM [7]. Although recent increases in the context window size of LLMs offer comparable performance, RAG remains highly effective in tailoring models' responses to specific, unseen datasets.

Key advantages of RAG include automation, data security, local deployment, extensive customization options - including flexibility in selecting embedding models and specific LLMs for generation - and enhanced scalability. Even with the expanded context windows of LLMs, scalability issues quickly emerge, especially in handling large volumes of data.

In this work, we present the theoretical foundations and practical implementation of the RAG approach to simplify scientific literature analysis on the topic of pancreatic cancer. The goal was to streamline the review of extensive scientific literature. Our dataset consisted of available articles from PubCentral and abstracts from PubMed. Because PubMed includes more than 36 million articles, a lot of them are behind paywall journals. Abstracts are freely available and provide a summary of objectives and findings. If we really want to take into account all the available knowledge at the time or in a time period, we should use the abstract to at least get a reference to the relevant article. Wherever the full article was available in PubCentral, it was used instead of an abstract. We compare RAG approach to generic LLM answers and citations.

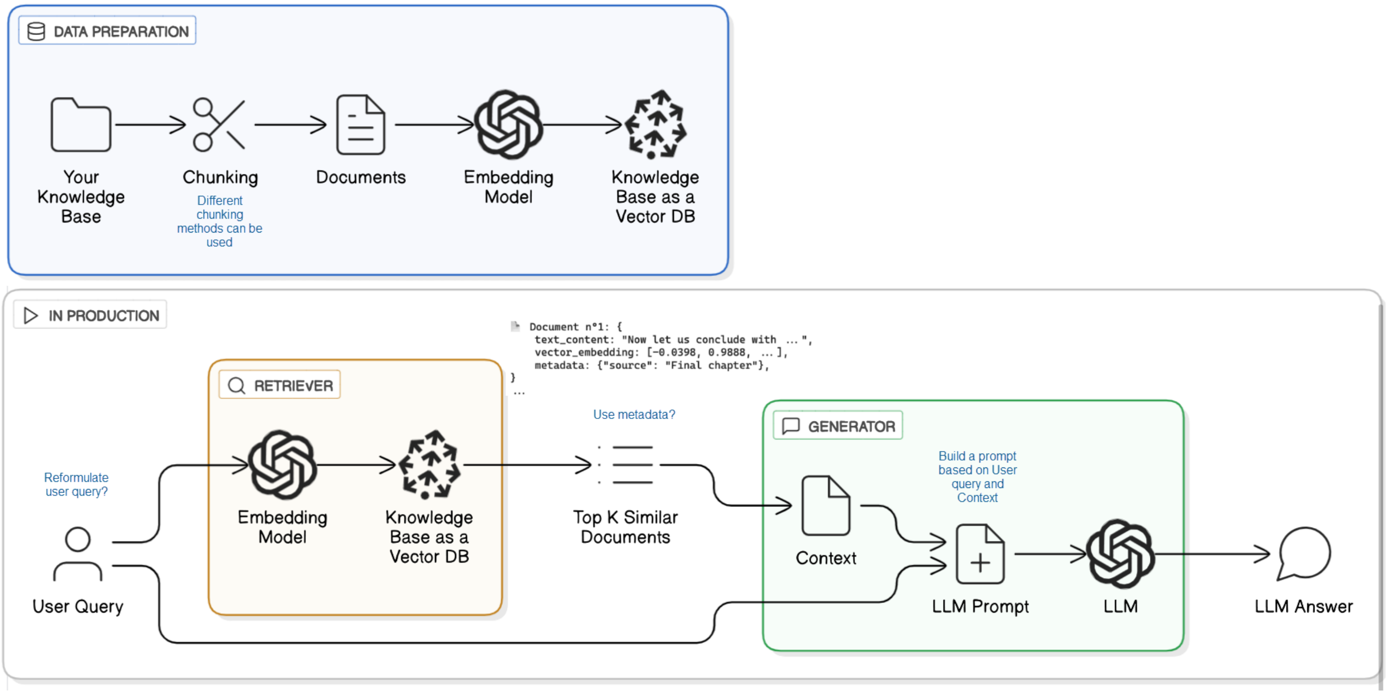

Figure 1: Retrieval Augmented Generation (RAG) system [10]

1.1. Retrieval-Augmented Generation (RAG)

Large Language Models (LLMs) are capable; however, they have certain limitations. The primary limitation is outdated knowledge. Training LLMs is computationally intensive and can take up to several months. As a result, each model's knowledge is frozen and limited to a specific knowledge cut-off date. For example, model 4o, provided by ChatGPT, has knowledge of data available up to June 2024. The second limitation is hallucinations. Hallucinations are the ability to generate answers that appear correct, but are factually wrong [30]. This issue occurs because the model relies on patterns learned from its training data, which may be incomplete, biased, outdated, or inaccurate. The hallucinations notably increase when handling prompts outside of their training domain or when they require current information.

To address LLM limitations, new solutions were developed, one of them being Retrieval Augmented Generation, which provides existing LLMs with additional knowledge by retrieving relevant information with semantic similarity calculation. The term was first used in a paper that proposed an architecture that links LLM (generative model) to a new, external database of information i.e. a vectorized index that can be queried in real time. Hybrid architecture directly confronts the core weakness of stand-alone LLM and improves factual accuracy, provides citable context and allows for continuous knowledge updates, without spending a lot of energy or time [14]. The RAG landscape is a very young field, and it is also rapidly expanding; new diverse implementations are presented on a daily basis. And while it is good that the field is so dynamic, there is a substantial challenge in methodically experimenting and evaluating systems. Because of this, we will focus on the most basic form of RAG - Naive RAG with vector storage.

Naive RAG pipeline is built from 3 primary phases. The initial phase is data preparation and indexing. In this phase, we transform raw source documents into a queryable database. We load the data from various sources and diverse formats. We extract the content and convert it to a uniform plain text format. The text is then prepared for the chunking stage. Chunking, often also called segmentation, is a step where large documents are segmented into smaller pieces. The goal is to get chunks that are small enough for efficient and accurate retrieval, but large enough to contain meaningful information [7]. There are several chunking strategies. They range from *fixed-size* - the simplest method where we decide on a fixed number of characters or tokens with typically 10-20% overlap of the chunk size to preserve at least some context between following chunks. The downside of this method is that it can cut sentences and hence break the semantic meaning. *Sentence splitting* uses sentence breakpoints like periods and new lines, ensuring that each chunk is at least one or more complete sentences. This method is better at preserving semantics but can lead to inconsistent chunk lengths. Maintaining consistent chunk lengths has been shown to improve retrieval accuracy in RAG pipelines and it also ensures they are compatible with transformer-based encoders that have strict token limits [1]. If we want to focus on whether the context preservation is important we can decide between early or late chunking. *Early chunking* is a method where documents are first split into smaller chunks and then embedded with the embedding model. Embedding is first done on the token level and then those are aggregated to produce one embedding for the whole chunk. *Late chunking* delays the chunking process. Instead of chunking the document first, we first embed the entire document at the token level. Then we use chunking on token embeddings and finally aggregate them into final chunk embeddings. In this way, the full original context is preserved [8], [19].

Chunks then need to be transformed into vectors by using an embedding model and then stored into a high-dimensional vector database. When the user inputs a query (prompt), this text is also embedded using the same embedding model as for the vector database.

Contextual text embedding is a numerical representation of a text of varied length in space where the ones with similar meanings are closer to each other. That means when meaning or context changes, the embeddings also change. In recent years, the development of universal text embedding models is on the rise, where the objective is to mimic human text processing, where a model can process and work well for many different text lengths, tasks, domains, and languages.[2] One of the most known benchmarks for embedding models is the Massive Text Embedding Benchmark (MTEB), and it's available on the Hugging Face platform [20]. The more popular models are Bidirectional Encoder Representations from Transformers (BERT)[4] based or LLM based.

After embedding the content and prompt, the retrieval stage happens. By comparing the query vector and chunk vectors, a similarity metric is used (e.g. Cosine similarity) and the top K chunks with the vectors closest to the query are retrieved. The top K chunks are called context. The query and context are forwarded to a pre-trained Large Language model such as ChatGPT, Gemini, Llama, which are then used to produce an augmented response based on the retrieved context and prompt [7], [5], [9].

Basic RAG consists out of two parts called a Retrieval and a (Augmented) Generation as presented in Figure 1. The number 1 in Figure 1 refers to the retriever part of the diagram and Generation is the 2. Reader part of the diagram.

1.2. LLamaIndex

LlamaIndex is a framework for building context augmentation AI applications with LLMs including agents and workflows. RAG is one of the most popular examples of context-augmentation. It combines context with LLMs at inference time. It is also one of the most used libraries to deal with RAG. They pride themselves on a famous "5 lines of code" starter example using OpenAI. We used LlamaIndex as a base framework, but with some advanced concepts of adding our own LLMs and embedding models.

To build the RAG, we used LlamaIndex built-in functions. Function *SimpleDirectoryReader* was used for ingesting the documents. This is the universal data loader. Every file format, including CSV, PDF, images, and even PPT, is treated like text. It can scan a directory or even sub-directories and create Document objects which are used for embedding. It can also filter out certain file types or skip specific files. The method *load_data()* returns a list of *Document* objects where each object includes raw text and a metadata dictionary.

When *Documents* were prepared, we used *VectorStoreIndex* to create an index over them. This is the default index type and is used in a lot of RAG implementations. Method *VectorStoreIndex.from_documents(documents)* splits each document into smaller chunks and computes a vector embedding for every single one. These embeddings are numerical representations of the text's meaning - text with similar meaning has similar vectors. By default, LlamaIndex uses OpenAI's *text-embedding-ada-002* for embeddings. The index stores all these vectors so searching is faster.

QueryEngine handles the retrieval and generation and sits on top of the index. We used *index.as_query_engine* to construct a default retriever and generator. Top-k chunks by similarity to the query from the index are returned and fed to LLM to generate an answer. As an input, *QueryEngine* receives a natural language query and returns a response based on embedding similarity, along with reference context that was retrieved and used for answer generation [29].

2. Methodology

2.1. Dataset

Initially our dataset consisted of article abstracts on pancreatic cancer from the PubMed website. As the project progressed through iterations, we expanded the dataset to include abstracts as well as whole articles when those were available from Pubmed Central. Selection parameters for articles were also made more precise to ensure specificity and relevant content. The selection period was limited to the years 2022, 2023, 2024, and 2025. The keywords used were the following "pancreatic cancer", "pancreatic adenocarcinoma PDAC", "natural killer cells", "cancer associated fibroblasts", "cysteine peptidases". The number of articles ranged from 300 to 6000. Initially, data was collected in the simplest format (.txt) due to prior familiarity with the technology, knowing this is the format that will probably yield the best initial results. The data was collected with a web crawler. Metadata was added to filenames. We cleaned collected data to produce raw text files appropriate for RAG system ingestion. We separated the metadata (title, author, journal) from the body and applied cleaning techniques that ensured we used lowercases (to reduce vocabulary size), normalised punctuations, removed bullet marks and other odd characters, collapsed excessive white spaces, removed any new line and link markdowns and repeated punctuations. This ensured only relevant characters and words were embedded and consequently retrieved and used for answer generation. Some traditional NLP normalization techniques (stop word removal, stemming, lemmatization) are not used because they interfere with the semantic representation of the text. For easier ingestion and metadata inclusion we structured python dictionary object. Metadata included file names, doi numbers, titles, authors, journals and publication date.

2.2. Model Selection

The RAG system requires two models - a large language model (LLM) for response generation and an embedding model for converting documents and queries into vectors. Embedding models are used to generate numeric vectors. The input is text

```
=== CONTEXT-GROUNDED ANSWER ===
The main NK cell activation markers within the PDAC microenvironment that are affected by cancer-associated fibroblasts (CAFs) include granzyme B, as reduced
expression of this marker is found to be related to the reduced killing activity of NK cells. This suppression of NK cell activity by CAFs contributes to tumor
progression in patients with PDAC.

=== QUERY-ONLY ANSWER ===
In the pancreatic ductal adenocarcinoma (PDAC) microenvironment, Natural Killer (NK) cells play a crucial role in tumor surveillance and immunity. The activation
of NK cells is characterized by changes in the expression of various surface markers. Here are some key NK cell activation markers that have been implicated in the
PDAC microenvironment:

1. CD69: Upon activation, NK cells upregulate CD69, a calcium-dependent integrin that plays a role in adhesion to target cells and cytokine-mediated activation.

2. CD107a (LAMP-1): This membrane protein is a lysosomal-associated membrane glycoprotein that translocates to the cell surface upon NK cell degranulation, which
is a key effector function of activated NK cells.

3. Perforin and Granzymes: These are cytotoxic proteins stored in NK cell granules and released upon activation to induce apoptosis in target cells.

4. Fas Ligand (FasL): FasL is a member of the tumor necrosis factor (TNF) family that can bind to its receptor Fas on tumor cells, leading to apoptosis.

5. Interferon-γ (IFN-γ): This cytokine is produced by activated NK cells and plays a significant role in the recruitment and activation of other immune cells, such
as macrophages and dendritic cells, against PDAC.

6. Tumor necrosis factor-α (TNF-α): TNF-α is another cytokine produced by activated NK cells that contributes to tumor cell death and inflammation in the PDAC
microenvironment.

=== TOP CONTEXTS (with metadata) ===
[1] Score: 0.608  Hits: 5  File: /home/lili/projects/rag_project/LIT_REVIEW/text_data/Natural_killer_cells_paper357_PMC10536815_cleaned.txt
[2] Score: 0.565  Hits: 4  File: /home/lili/projects/rag_project/LIT_REVIEW/text_data/pancreatic_cancer_paper283_PMC9354534_cleaned.txt
[3] Score: 0.549  Hits: 4  File: /home/lili/projects/rag_project/LIT_REVIEW/text_data/pancreatic_cancer_paper13_PMC11151541_cleaned.txt
[4] Score: 0.547  Hits: 4  File: /home/lili/projects/rag_project/LIT_REVIEW/text_data/PDAC_paper595_PMC11151541_cleaned.txt
[5] Score: 0.528  Hits: 3  File: /home/lili/projects/rag_project/LIT_REVIEW/text_data/pancreatic_cancer_paper855_PMC9119758_cleaned.txt
```

Figure 2: RAG response

and the output is a numeric vector with a semantic description of the text. Embedding models are specifically trained for this task and should not be confused with large language models, which are used for generation of an answer. The most basic use case of LlamaIndex uses the OpenAI embedding model *text-embedding-ada-002*, but customization is also possible. Embedding model selection is very important and it can substantially influence the results. Ranking of the most notable embedding models is available on the Hugging Face platform [10], [20].

During the literature review, we noticed that Mistral is a very popular and well-performing model, especially for multilingual uses. [16] Its performance is similar to OpenAI models, but we found it to be more consistent at the time, compared to GPT-3 - the answers we received back asking the same question were mostly the same. The API for Mistral models can also be used for free through Ollama locally. We used *nomic-embed-text* as an embedding model, because it was one of the available ones on Ollama.

2.3. Evaluation approach

A RAG system can be assessed as a whole or one can evaluate every component individually. This means that we can separate the evaluation of the retriever component and the generation component [11]. An RAG system that consistently outputs high-quality answers typically has both a robust data retrieval component and an excellent generation component. If the retrieval component accurately fetches relevant fragments of context and if the generator component can effectively utilize that context to produce answers that are precise and factually correct, then we can confidently say that our RAG system is performing well [12].

Due to the complex topic of the literature, we used an expert in the field to assess the quality of generation, that is, the quality of given answers, contexts, and citations for 3 questions. Scoring tells us, once the documents are retrieved, how well LLM uses them. The score is given 0-5 based on 5 criteria. Faithfulness tells us if the answer stays consistent with retrieved sources or if it hallucinates, relevance tells us semantic alignment with the question and if the response actually answers the user's question, conciseness tells us if the answer is clear and not overly verbose, readability tells us if the response is factual and explanatory, citation quality is a score given based on how many references given actually exist or are hallucinated.

3. Implementation and Results

This project involved developing a RAG question-answering system within the wsl and conda environment in jupyter lab, utilizing computational resources (an NVIDIA GeForce RTX 5070 Ti GPU with 16GB memory, 64GB RAM). Importing all the correct packages and versions was a very delicate task. For the use of large language model and embedding model we utilized Ollama - free open-source tool that enabled us to run LLMs locally. We built the basic RAG system and upgraded it with the settings for chunk size, overlap, and the number of retrieved contexts. For all three of these settings, it is generally good to experiment with a variety of values while keeping in mind text shape and expected answers. Chunks are typically configured based on the specifications of the documents that are processed and the level of detail with which we want to compare the documents to the query. In our case we set the chunk size to 400 and the overlap to 60. We also increased the number of retrieved contexts to 5. Furthermore we improved the output of the answer by including the context from which the answer

Question	RAG	ChatGPT	DeepSeek	Explanation
Faithfulness	5	3	5	RAG and DeepSeek give clear answers, whereas ChatGPT goes off topic; it is focused more on NK cell plasticity than tumor differentiation stage.
Relevance	4	3	5	DeepSeek frames the answer explicitly around PDAC grade and NK cells, which is relevant to the question. RAG gives general answers without citing specific studies or survival/prognostic data. ChatGPT has scientific grounding but is less tightly focused on the differentiation stage.
Conciseness	4	3	5	DeepSeek provides the clearest, most structured correlation with differentiation stages.
Readability	4	3	5	RAG is written in a narrative style, like a review; ChatGPT is dense and less smooth, whereas DeepSeek is the most readable, structured, and concise.
Citation quality	5	5	2	DeepSeek has cited the relevant paper; however, it uses a different DOI number.

Table 1: Evaluation of RAG, ChatGPT, and DeepSeek across different metrics for the research question: What are the main NK cell activation markers within the PDAC microenvironment?

Question	RAG	ChatGPT	DeepSeek	Explanation
Faithfulness	3	4	5	Very strong for DeepSeek, which explains how each marker is altered in the PDAC TME. ChatGPT is less exhaustive, and RAG goes slightly off-topic, mixing activation markers and cytokines.
Relevance	3	5	5	RAG starts with CAFs suppressing NK cells, which drifts from the main question. ChatGPT and DeepSeek give fully relevant answers focused on PDAC activation markers.
Conciseness	4	5	4	DeepSeek gives a long, detail-heavy answer. ChatGPT provides the most concise answer, whereas RAG gives a moderate-length answer, but the first paragraph about CAFs is unnecessary for the question.
Readability	5	5	5	All models provide concise and structured answers, making them easy to read for both expert and general audiences.
Citation quality	5	3	3	DeepSeek has cited the relevant paper; however, it uses a different DOI number.

Table 2: Evaluation of RAG, ChatGPT, and DeepSeek across different metrics for the research question: Are cancer associated fibroblasts targeted by NK cells in pancreatic cancer microenvironment?

Question	RAG	ChatGPT	DeepSeek	Explanation
Faithfulness	5	5	5	All models provided strict results by context. Hallucinations were not detected.
Relevance	3	3	5	All models provided generalised answers. Unlike RAG and ChatGPT, DeepSeek provided certain details which are relevant to the question. For example, DeepSeek explained both physical barriers (desmoplastic stroma) and molecular immunosuppressive mechanisms.
Conciseness	4	4	3	DeepSeek has detailed immunological terms, including an overwhelming amount of information for science student readers.
Readability	4	4	3	Generally, the clarity of the provided information is overwhelming.
Citation quality	5	2	3	DeepSeek has cited the relevant paper however, with a different DOI number.

Table 3: Evaluation of RAG, ChatGPT, and DeepSeek across different metrics for the research question: Are cancer associated fibroblasts targeted by NK cells in pancreatic cancer microenvironment?

was derived, together with the evaluation of relevancy for that context. We tested the query in its basic form - just the question and in an augmented form, where we provided more detailed instructions in addition to the question. We also added a list of synonyms that helped us generate more of the same queries then picked the top 5 relevant contexts that were used in answer generation.

The primary challenges we encountered during the development process were mainly related to versions of the libraries and packages we utilized. The temporal complexity of the system was also significantly impacted due to the high number of articles we incorporated into the processing. Specifically, when we scaled up the number of articles to 6000, the processing time increased considerably. To mitigate this, it would be necessary to include only the most relevant and crucial sources; however, the complexity of the domain knowledge here requires an expert.

3.1. Model evaluation

The answers generated by the RAG system were compared with responses to the same questions provided by ChatGPT and DeepSeek. We selected these two widely known and commonly used LLMs as baseline models for comparison across five evaluation criteria: Faithfulness, Relevance, Conciseness, Readability, and Citation Quality. The measurement scales and their meaning are as follows:

1. **Faithfulness:** 5 if fully supported by context, 0 if unsupported or hallucinated.
2. **Relevance:** Semantic alignment with the question (0–5).
3. **Conciseness:** 5 if direct and compact, 1 if verbose or off-topic.
4. **Readability:** Clarity and structure (0–5).
5. **Citation quality:** The score reflects the number of existing publications suggested by the model: 5 means all 5 references exist, 0 means none exist.

An expert in pancreatic cancer was asked to evaluate all three models based on three relevant research questions. These questions, also provided by the expert, were as follows:

1. Are cancer associated fibroblasts targeted by NK cells in pancreatic cancer microenvironment?
2. What is the correlation of NK cells infiltration and differentiation stage of pancreatic cancer?
3. What are the main NK cell activation markers within the PDAC microenvironment?

The evaluation results for the first question are provided in Table 3, for second one are provided in Table 1 and for the third one in Table 2.

The results clearly demonstrate the advantage of using RAG over general LLMs for domain-specific topics. The key strength of RAG lies in its ability to accurately provide references to the source documents supporting its answers, thereby avoiding hallucinations. This feature is particularly valuable for students who are new to the field and are in the process of conducting a literature review.

While RAG produced less elaborate answers compared to ChatGPT and especially DeepSeek, its responses were consistently accurate and demonstrated a markedly lower tendency toward hallucinations.

A key limitation of RAG is the complexity of its implementation. In contrast to the readily accessible versions of ChatGPT and DeepSeek, deploying RAG necessitates considerable resources and expertise, including domain specialists for curating relevant literature and computational scientists for system development and integration.

Another limitation is the wide array of implementation and customization options available for RAG. In this study, we employed Mistral for training; however, the rapidly expanding market for large language models complicates the decision of which model is most appropriate for long-term use.

4. Improvements and Future work

For this project, we built a very basic RAG system, but there is a lot of room for enhancements with various other techniques. These include leveraging metadata, reformulating and enriching queries, exploring different methods for calculating similarity scores, optimizing the approach for document ranking, re-ranking strategies, and fine-tuning the large model itself.

4.1. Advanced Evaluation Methods

Since the evaluation procedure of such a system is quite demanding, there were various tools made to make it easier. One of such open-source tools for evaluating RAG systems is RAGAS [24]. RAGAS offers a variety of metrics that assess different components or the system as a whole. Generally, the evaluation involves using *ground truth*, which refers to a correct answer verified by an expert who can guarantee its accuracy. Creating pairs of questions and answers for ground truth is, therefore, a very time-consuming task. The evaluation metrics then compare the ground truth context with the retrieved context (in the case of the retrieval component) or the ground truth answer with the generated answer. The more ground truth examples we have, the better we can assess the RAG system's quality. Additionally, it is important to include users while testing the system, so they can subjectively evaluate how well the system answered their questions, how accurate it was, and how helpful they found it. Tools struggle to correctly assess user experience.

The system can also be evaluated holistically based on its specific application. For instance, in a question-answering system, F1 score or EM (Exact Match) can be used, while for customer support systems, metrics like satisfaction scores are useful or those measuring the number of successfully resolved cases might be more appropriate. For comparing different RAG systems, A/B testing can be used. This method compares the performance of two system versions based on user interaction and their preference (for example, when ChatGPT occasionally presents two versions of an answer, and we can select the one we find more useful and/or accurate). Finally, it's essential to consider the RAG system's speed and resource consumption to ensure that our system is efficient enough for practical deployment.

4.2. Multimodality

A multimodal RAG system differs from a purely textual one because, in addition to text sources, it incorporates image, video, and audio sources. A key decision here is whether we want multimodal answers, or if we just want to retrieve information from multimodal sources. If we aim for multimodal answers, then we need to use multimodal large language models (like GPT-4o, and later ones, Llama-3.2-Vision) for generation [25]. A picture is worth a thousand words, some may say. Hence, multimodal answers can help users understand the system's responses faster and more easily [17].

There are several approaches to implement a multimodal RAG system. We can embed all the sources into the same vector space, convert all sources into a single modality, or store each source type separately.

If we opt for the first approach, where we embed all image and text sources into the same vector space, we can use a multimodal embedding model like CLIP [23]. Consequently, we can employ a similar RAG system to the textual approach - we simply swap out the embedding model, and if we desire multimodal answers, we also replace the large language model with a multimodal one.

In the second approach, we choose a primary modality. This decision is usually based on the application's primary functionality. For example, in our case, where the application is primarily designed for textual questions and generation of textual answers, we could create textual descriptions and metadata for image sources and embed them just like text. The images would be stored as an additional resource for later use; perhaps if we use a multimodal large language model for generation, we could then display the images. The downside of this approach is an increase in preprocessing time and potential data loss during the conversion of images to text [28].

With the third approach, we separate all the modalities and store them individually. When we query the system, we get the top k results from each source separately. Then, a dedicated multimodal re-ranker is used to extract the overall most suitable context [3].

5. Discussion and Conclusion

RAG (Retrieval Augmented Generation) is a recent technology, but it is evolving at a great pace, with recent and more advanced concepts emerging every day. While the core idea and implementation might seem straightforward, it often happens that our expectations are too high and the impression of simplicity is quite misleading. Nevertheless, this topic is exceptionally interesting and relevant in today's world, and it can significantly improve communication with artificial intelligence.

For this project, we built a RAG system based on scientific articles to answer specific questions on the topic of pancreatic cancer. We limited the scope to textual modality and enhanced the system using various methods. It was especially important for us to not only get the answers but also the context from which the answer was formed, along with the references to the articles from which that context was retrieved. This allowed the user to later verify the answers themselves, or to find and focus on the articles most relevant according to the posed question. Of course, many other techniques could be tried to improve the system, such as using different large language models, embedding models (including specialized medical large language models), implementing custom vector or graph databases, reranking, fine-tuning, further relying on metadata, different similarity metrics, and using agents. Advanced prompt engineering techniques, where the system takes a question and generates hypothetical answers, which are then also embedded and used in context retrieval, could also be implemented. The possibilities are numerous, but once we reach a point where we get reasonably sensible results, implementing an evaluation system becomes crucial. This ensures that when we add new improvement techniques, we can confirm whether our results are genuinely better or worse than in previous versions.

The most important step is to have domain experts test and evaluate the system's performance. The topic is highly specific and specialized, so expert knowledge is necessary for posing questions and assessing the answers. Future work may consider machine-based verification of the RAG performance and the implementation of multimodality.

Acknowledgments

The authors acknowledge the financing of the FUTURE project CAT-AI-THERAPY.

References

[1] Sinchana Ramakanth Bhat et al. *Rethinking Chunk Size For Long-Document Retrieval: A Multi-Dataset Analysis*. 2025. arXiv: 2505.21700 [cs.IR]. URL: https://arxiv.org/abs/2505.21700.

[2] Hongliu Cao. "Recent advances in text embedding: A Comprehensive Review of Top-Performing Methods on the MTEB Benchmark". In: *arXiv preprint arXiv:2406.01607* (2024).

[3] Wenhu Chen et al. *MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text*. 2022. arXiv: 2210.02928 [cs.CL]. URL: https://arxiv.org/abs/2210.02928.

[4] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (2019).

[5] Paulo Finardi et al. "The Chronicles of RAG: The Retriever, the Chunk and the Generator". In: *arXiv preprint arXiv:2401.07883* (2024).

[6] Thiago F. A. França and José M. Monserrat. "To Read More Papers, or to Read Papers Better? A Crucial Point for the Reproducibility Crisis". In: *BioEssays* 41.1 (2019), e1800206. DOI: 10.1002/bies.201800206.

[7] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* (2023).

[8] Michael Günther et al. *Late Chunking: Contextual Chunk Embeddings Using Long-Context Embedding Models*. 2025. arXiv: 2409.04701 [cs.CL]. URL: https://arxiv.org/abs/2409.04701.

[9] Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. *A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions*. 2024. arXiv: 2410.12837 [cs.CL]. URL: https://arxiv.org/abs/2410.12837.

[10] Hugging Face. *Hugging Face: State-of-the-art Machine Learning for Everyone*. https://huggingface.co/. 2024.

[11] Jeffrey Ip. *LLM Evaluation Metrics: The Ultimate LLM Evaluation Guide*. Sept. 2024. URL: https://www.confident-ai.com/blog/llm-evaluation-metrics-the-ultimate-llm-evaluation-guide (visited on 09/30/2024).

[12] Jeffrey Ip. *RAG Evaluation: The Definitive Guide to Unit Testing RAG in CI/CD*. Apr. 2024. URL: https://www.confident-ai.com/blog/how-to-evaluate-rag-applications-in-ci-cd-pipelines-with-deepeval (visited on 09/30/2024).

[13] Taja Kuzman et al. "PandaChat-RAG: Towards the Benchmark for Slovenian RAG Applications". In: (2024).

[14] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2020. arXiv: 2005.11401 [cs.CL].

[15] Qi Liu, Matt J Kusner, and Phil Blunsom. "A survey on contextual embeddings". In: *arXiv preprint arXiv:2003.07278* (2020).

[16] Nikola Ljubešić et al. "JSI and WüNLP at the DIALECT-COPA Shared Task: In-Context Learning From Just a Few Dialectal Examples Gets You Quite Far". In: *Proceedings of the Eleventh Workshop on NLP for Similar Languages, Varieties, and Dialects (VarDial 2024)*. 2024, pp. 209–219.

[17] Zi-Ao Ma et al. *Multi-modal Retrieval Augmented Multi-modal Generation: A Benchmark, Evaluate Metrics and Strong Baselines*. 2024. arXiv: 2411.16365 [cs.CL]. URL: https://arxiv.org/abs/2411.16365.

[18] Nigel Markey et al. "From RAGs to riches: Using large language models to write documents for clinical trials". In: *arXiv preprint arXiv:2402.16406* (2024).

[19] Carlo Merola and Jaspinder Singh. *Reconstructing Context: Evaluating Advanced Chunking Strategies for Retrieval-Augmented Generation*. 2025. arXiv: 2504.19754 [cs.IR]. URL: https://arxiv.org/abs/2504.19754.

[20] Niklas Muennighoff et al. "MTEB: Massive Text Embedding Benchmark". In: *arXiv preprint arXiv:2210.07316* (2022). DOI: 10.48550/ARXIV.2210.07316. URL: https://arxiv.org/abs/2210.07316.

[21] Humza Naveed et al. "A Comprehensive Overview of Large Language Models". In: *ACM Trans. Intell. Syst. Technol.* (June 2025). Just Accepted. ISSN: 2157-6904. DOI: 10.1145/3744746. URL: https://doi.org/10.1145/3744746.

[22] Pinecone. *Chunking Strategies for LLM Applications*. https://www.pinecone.io/learn/chunking-strategies/. Accessed: 2025-07-18. 2023.

[23] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[24] *RAGAS Documentation (Stable Version)*. URL: https://docs.ragas.io/en/stable/.

[25] Dipanjan (DJ) Sarkar. *A Comprehensive Guide to Building Multimodal RAG Systems*. 2025. URL: https://www.analyticsvidhya.com/blog/2024/09/guide-to-building-multimodal-rag-systems/.

[26] Sabina Siebert, Robert Insall, and Laura M Machesky. "More means less: Managing overflow in science publishing". In: (2019).

[27] Sabina Siebert, Laura M. Machesky, and Robert H. Insall. "Point of View: Overflow in science and its implications for trust". In: *eLife* 4 (Sept. 2015), e10825. ISSN: 2050-084X. DOI: 10.7554/eLife.10825. URL: https://doi.org/10.7554/eLife.10825.

[28] Annie Surla, Aditi Bodhankar, and Tanay Varshney. "An Easy Introduction to Multimodal Retrieval-Augmented Generation". In: *NVIDIA Technical Blog* (2024). URL: https://developer.nvidia.com/blog/an-easy-introduction-to-multimodal-retrieval-augmented-generation/.

[29] LlamaIndex Docs Production Team. *LlamaIndex Documentation*. 2024. URL: https://docs.llamaindex.ai/.

[30] SM Tonmoy et al. "A comprehensive survey of hallucination mitigation techniques in large language models". In: *arXiv preprint arXiv:2401.01313* (2024).

[31] Peng Xia et al. *MMed-RAG: Versatile Multimodal RAG System for Medical Vision Language Models*. 2024. arXiv: 2410.13085 [cs.LG]. URL: https://arxiv.org/abs/2410.13085.

[1] These authors contributed equally to this work.