Leveraging LLM-powered Systems to Accelerate *Mycobacterium tuberculosis* Research Step One: From Documents to the Vectorstore

Christophe Guyeux$^{1[0000-1111-2222-3333]}$, David Laiymani$^{1[0000-0003-2580-6660]}$, and Christophe Sola$^{2[0000-0003-4672-2140]}$

[1] FEMTO-ST Institute, UMR 6174 CNRS, Université de Franche-Comté, France
{christophe.guyeux,david.laiymani}@univ-fcomte.fr
[2] IAME, UMR 1137 INSERM-Université Paris-Cité, Université Paris-Saclay, France
christophe.sola@universite-paris-saclay.fr

Abstract. The remarkable performance of recent large language models has made the conception of new advanced tools to assist researchers in specialized fields feasible, provided these models are supplied with relevant specialized information. This article is the first in a series which aim to present our personal feedback in developing such tools within the context of studies on the bacterium *Mycobacterium tuberculosis*. One of the critical aspects of these developments involves extracting useful information from an extensive collection of over 100,000 research articles on this bacterium, which is highly specialized and diverse, and possesses unique characteristics. This initial article examines how information retrieval can be implemented in this context, based on our experience, and discusses optimal methods for recovering PDFs, extracting information, encoding it, and determining the appropriate type of vectorstore for storage. The approach presented here is not exclusive to this particular bacterium but can be extended to numerous other research areas.

Keywords: Large Language Models; Retrieval-Augmented Generation; Scientific RAG; LLM-powered Systems; Experience feedback.

1 Introduction

The development of new neural architectures, such as transformers and their attention mechanisms [18], has facilitated the emergence of high-quality generative artificial intelligence. This advancement has, in turn, led to several significant developments, with the creation of chatbots, ChatGPT-4 [15], Le Chat Mistral [11] and Claude [2] being among the most notable. These systems, which engage in conversations using extensive knowledge acquired from various sources like Wikipedia or the Common Crawl, have demonstrated a real capacity for discussion and some kind of reasoning.

However, the general nature of the knowledge acquired by such chatbots limits their daily utility for researchers within specific disciplines. This is attributed to the vast extent of scientific literature—for instance, over 100,000 articles on the tuberculosis (TB) disease alone—and to copyright restrictions that complicate the creation of a sophisticated information retrieval tool for all research topics. The impossibility of generating embeddings that finely capture the semantics of any research text across varying topics is also noted, for a field of human activity producing more than 2 million scientific articles a year. An additional challenge is the temporal nature of research, which continuously evolves, whereas the Large Language Models (LLMs), underpinning these chatbots, are trained on data up to a specific date.

From this perspective, Retrieval-Augmented Generation (RAG, see Fig. 1), has been identified as a first promising co-pilot for researchers allowing to overcome the weaknesses of LLMs chatbots. Let us recall that a typical RAG application has three main components:

- Indexing: a pipeline designed to ingest data from a source and index it, typically occurring offline.
- Retrieval: which takes the user's query in real time and retrieves the relevant data (also named *context*) from the index, previously built.
- Generation: which add the context to the query and submitted the both to an LLMs.

To sum up, this approach involves augmenting the LLM's response with context, which comprises additional knowledge automatically provided to the LLM to compensate for its limited domain-specific knowledge. Typically, this context includes excerpts from research articles that address the posed question. With the general knowledge and reasoning skills that today's LLMs possess, they are, when supplemented with appropriate research texts, capable of delivering relevant responses to specialist researchers.

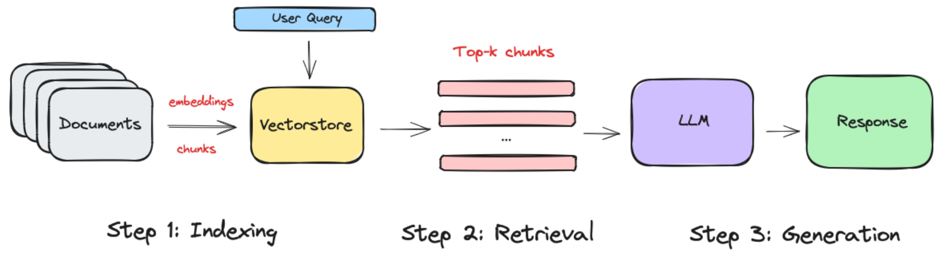

Fig. 1: The Retrieval-Augmented Generation approach

Frameworks such as LangChain[3] have greatly simplified the construction of RAG tools using default components (e.g., chromadb for vector storage, GPT-4 as the LLM), which perform adequately over small collections of personal documents. Furthermore, there are platforms allowing users to upload a PDF

[3] https://www.langchain.com

and create a question-and-answer chatbot based on it. However, such a basic approach fails to meet the specific needs of researchers in specialized fields due to the generic and generalist nature of the default integrated components. To develop an effective research co-pilot, for instance in the field of tuberculosis, a substantial volume of relevant, specialized data and sophisticated methods of indexing this data are required. This ensures the most useful information is extracted and fine-tuned usage of one or more LLMs is achieved.

However, the question-and-answer chatbot is just one of the advanced LLM-powered tools that can help researchers working on *Mycobacterium tuberculosis*, the agent of human and animal TB on a day-to-day basis, and this is a point worth emphasising. Other potentially very useful tools can be developed, including:

- Article search: a search engine that focuses not on the title or keywords, but on the meaning of the article. For example, to find the article in which a given author established a given fact (reference search). Or to find articles dealing, at a given point in their text, with the current or past TB pandemic situation in the Gulf of Benin, in order to take stock of the state of the art – the term may not even be found in the entire article. Or again, articles mentioning the effect of a particular mutation in a particular gene, or which have studied a particular strain.
- Summarizing, for example, of the whole scientific work of a particular author, or of current knowledge of a particular drug-resistant TB in a particular place (e.g., to Rifampin in northern Turkmenistan or Bedaquiline in India).
- Scientific monitoring: automatically analyse the twenty or so new articles published each day, indicate whether certain elements are important for a given research context, summarise the day's production, highlight articles or extracts from texts, etc.
- Find collaborators: help researchers find potential collaborators by analysing recent publications and citations, to identify experts in specific fields.
- Checking facts and beliefs: take a critical look at the chatbot's output, but also at the assertions contained in the researcher's request, to point out erroneous facts and suggest readings that contradict these facts.
- Creation of datasets: for example, a set of mutations that have led to clinically proven resistance (phenotypic) to a given antibiotic, in order to create a learning base for an AI model predicting this resistance (such datasets, which are invaluable, are still limited for recently launched drugs).
- Carry out health monitoring on weak clinical signals: events marginally spread in a few independent articles, which together show the emergence of a new event (e.g. a highly epidemic clone in a given lineage).
- Improve local personalized medicine for TB treatment given specific patient contextual history.
- Lastly, but importantly, the advent of integrating Next-Generation-Sequencing (NGS) genomic studies with dynamic, complex spatio-temporal parameters and socio-economic factors, thereby creating integrated perspectives in the ecology of infectious diseases, necessitates the development of new

approaches including RAG with LLMs. These approaches must be made increasingly familiar to researchers working in the field of infectious diseases medicine.

Note that the primary goal of such systems is not to replace the researcher with generative AI but to consider the design of intelligent tools with maximized utility, useful in the daily activities of researchers. Such tools would aid in investigations, expedite the information search process, facilitate the identification of consensus by comparing various articles, and support scientific monitoring, among other capabilities. Having such components in mind, we propose to write a series of articles which serve as feedbacks and progress reports on the development of LLM-powered tools relevant for researchers, by illustrating our point with the case of *Mycobacterium tuberculosis*. The various stages of development, potential outcomes, choices made, and the rationale behind these choices will be thoroughly analyzed.

In this first paper, we will focus on the constitution of the database in the information retrieval stage. The next section discusses the available sources in the context of an isolated researcher building up his own LLM-based tools, and the methods of accessing these sources within the specified domain. Section 3 explores the techniques for extracting information from these documents. In Sections 4 and 5 we discuss the different possible strategies for splitting and storing this information. The article concludes with a section summarizing what has been learned from these investigations and clarifying the questions that remain open.

2 Source provenance and selection

To enable information retrieval, text is required. In the context of a chatbot for researchers working on bacteria of the genus *Mycobacterium*, several sources of information can be considered. Research articles, of which there are 100,000 on PubMed for this bacterial genus alone [14], constitute the first source. However, this source poses several challenges.

Primarily, the vast majority of these articles are copyrighted, and for many, only the abstracts are available. The automatic retrieval of articles thus presents both technical and legal challenges, and currently, no solution has been found that would enable the setup of a public chatbot for all researchers working on this bacterium. On the other hand, as a researcher with access to various scientific journals through a university library, there is authorization to download and consult these journals, and the fact that this consultation is conducted using a tool developed personally for aiding one's research presents no legal issues. This simple fact justifies this article, which aims to assist other researchers in this field to do the same within a legal framework.

The challenge remains on how to build up a knowledge base of significant size. On the one hand, it is impossible to download over 100,000 articles by hand via the interface of a university's digital library. And on the other hand, sticking to relatively simple IT techniques and within one's own legal framework, no automatic means seems feasible or justifiable. Additionally, the freshness of the data is crucial: this knowledge base cannot consist solely of old articles, as knowledge evolves and is enriched over time. To be truly useful and up to date, new articles need to be integrated continuously, something that chatbots cannot do.

However, a pragmatic approach is possible, to collect a database of articles that is rich enough to be useful. Researchers usually maintain their own personal library, with hundreds to thousands of articles stored on their own computer. If they consult with their colleagues or students, the number of articles can become substantial, and the database thus constituted is 'tailor-made', as the articles in one's personal library are precisely those that relate to one's particular theme (for example, Phylogeography of *Mycobacterium africanum*, a subspecies of *Mycobacterium tuberculosis*). Since not all the knowledge in one's personal library is mastered, and since manually searching for information in a potentially large PDF database is time-consuming and not very exciting, this is the first advantage of having a high-performance information retrieval tool for the articles of most interest.

A good practice then consists of, when embarking on a new research project (for example, on *Mycobacterium abscessus*), manually collecting relevant articles via the digital portal of a university library, and enriching one's PDF base with these articles. This is not an additional step in the work of researchers, for whom an update of current knowledge is a prerequisite for any work, but simply to get into the habit of enriching one's article knowledge base with each new literature review.

This initial database can be enriched with articles on other subjects relating to the *M.tuberculosis* bacterium, to create a chatbot with a broad knowledge base. This can be done by integrating a variety of open access articles semi-automatically. PubMed Central (PMC) is an open access database managed by the National Institutes of Health (NIH) in the United States. It provides free access to full-text research articles in the biomedical and life sciences fields. Using Entrez, a federated search system developed by the National Center for Biotechnology Information (NCBI), one can automatically query PMC, and then use the NCBI FTP server to access the archive of these open access articles. Interestingly, in addition to the PDF, this archive contains the supplementary materials and the text of the article in xml format.

In the same spirit, arXiv [6] contains freely accessible preprints, with bioRxiv [4] and medRxiv [5] serving as specialized variants for the biological and medical fields respectively. Setting up code to fetch search results for 'tuberculosis' in both PDF and text formats is straightforward, requiring only an HTML parser to retrieve these documents along with any supplementary materials, however keeping in mind that these articles are not yet peer-reviewed.

Regarding the ongoing integration of the latest articles on *M.tuberculosis*, the most effective approach identified involves receiving daily alerts from NCBI about new articles related to this keyword, and then manually downloading them each day through a university portal. This is certainly a tedious task, but it only involves around twenty articles a day, and this latter can be partially automated, for example using a python script. A systematic and pragmatic approach of this kind provides access in a reasonable time to almost 20% of the world's output of articles on tuberculosis, with a focus on recent articles and those specific to one's research.

Last, but not least, databases may be worth integrating: the bacterio.net website provides a list of prokaryotic names with standing in nomenclature [16]; the Mycobrowser website [10], another example, contains detailed 'cards' for each of the 4,000 *M. tuberculosis* genes, as well as for those of around ten associated species. There are also similar databases for bacteriophage viruses and insertion sequences, for example (see, e.g., ISFinder [17]), which might be useful to integrate in some way. Furthermore, the usefulness of such a knowledge base would be enhanced by receiving articles on other related topics. For instance, to extend the research articles to the entire *Mycobacterium* genus, to which the *M.tuberculosis* bacterial species belongs, can clearly enrich the database by broadening its spectrum, ultimately providing a more global vision. Finally, thesis manuscripts, courses and books on bacteria, biology, bioinformatics, statistics, and even history, demography, and the study of civilization, can be relevant too.

3 Information extraction

The challenge of extracting information from the knowledge base will be addressed in this section, with a focus solely on research articles. The more specific cases of databases and other information sources will be considered in a future article.

3.1 PDF Loader

Research articles are predominantly available in PDF format, often accompanied by supplementary materials. It is necessary to extract information from these PDF files and store it in a database to facilitate subsequent information retrieval. Various PDF document loaders are available and generally perform adequately. For example, LangChain integrates several loaders based on tools such as PyPDF, PyMuPDF, MathPix (based on the OpenAI API), Unstructured, PDFMiner, etc. Some loaders utilize neural networks, while others are designed for OCR, image extraction, or even structure extraction (e.g., headings, sections).

However, tests have indicated a preference for Grobid [1], a machine learning library tailored for extracting, parsing, and restructuring raw documents such as PDFs into structured TEI-encoded[4] documents, particularly focusing on technical and scientific publications. This preference is based on the library's long-term project development for neural networks specifically designed for extracting texts from PDF research articles. Grobid has demonstrated enhanced performance,

[4] https://tei-c.org

notably in extracting tables and images along with their captions. It accurately recognizes both the text they contain and their placement on the page, allowing for the extraction of tables as images, for instance, which are then processed using a neural model specifically trained for reading tables. Additionally, Grobid provides fine-grained reference management, crucial in scholarly articles, and automatically links labels to their corresponding references. Although LangChain offers a Grobid PDF loader, a standalone version has been adopted to allow for more precise configuration.

3.2 Title, Abstract, List of Authors, and Publication Year

Research articles possess specific characteristics that necessitate careful and precise extraction to enhance the performance of information retrieval and, consequently, the RAG pipeline. These characteristics include the title, the list of authors, the publication year, and the abstract. Each of these elements is significant for several reasons.

- Titles can be highly informative and are often necessary to index the paper.
- Abstracts summarize key facts and findings of the articles, providing crucial background information.
- The list of authors is critical information that needs to be accurately detected and read.
- The publication year is essential for discerning between conflicting information, with preference often given to more recent articles where there is consensus.

For instance, our discovery of a tenth lineage in the *Mycobacterium tuberculosis* complex this year [9] is an example where the title of the publication ('Newly Identified *Mycobacterium africanum* Lineage 10, Central Africa') provided essential information for retrieval systems to identify relevant content correctly.

The extraction of these fields, however, poses considerable challenges. Titles, for example, often use distinctive fonts and sizes, and their extraction is complicated by smaller spaces that are frequently misdetected. Abstracts vary significantly in their placement from one publication to another, sometimes being positioned in margins or colored frames, which leads to their misidentification as figures or their incorporation into body text. Author lists also present unique challenges as they need to be recognized as distinct from other textual content, despite often being in the same font as the rest of the text. The extraction of authors is further complicated by the varied formatting of names across different publications and countries.

To address these issues, we have developed a four steps method:

- First, we extract potential titles via multiple methods including metadata, Grobid, and pdfminer.layout [8].
- Second, the Crossref API [7] is utilized to retrieve DOIs for these titles, and the associated BibTeX is then obtained using doi2bib [3].

– Third, if a consensus on the DOI is reached and the title in the BibTeX closely matches the potential titles (based on Levenshtein distance [12]), it is concluded that the correct BibTeX—and thus a clean title, a well-formatted list of authors, and a verified publication year—has been found.
– Last, the abstract is consolidated from this BibTeX by employing LangChain's PubMed retriever, which ensures that the retrieved document's title closely matches the queried title, accounting for potential discrepancies.

3.3 The body of the text, figures, tables and references

Managing the body of the text has presented no particular difficulties, as Grobid has performed adequately in this regard. Custom post-processing operations have been introduced to expand genus and species names, as explained later in this article. For figures and tables, a decision was made to temporarily set aside their extraction.

Figures are mainly well-detected by Grobid. Some of these figures are provided for illustrative purposes only, and do not contribute substantially to the understanding of the text. While other figures, such as phylogenetic trees or diagrams of bioinformatics pipelines, or complex multilayers maps, contain valuable information, the existing multimodal LLMs have not yet achieved a level of image comprehension comparable to text comprehension for complex research diagrams. Therefore, the extraction of figures has been deprioritized in favor of focusing on text-based information, which is generally more reliably informative.

Tables frequently contain critical information, such as lineage data with associated genetic markers. Despite testing various state-of-the-art neural network models designed to extract text from table images, the decision was made not to include tables at this stage due to the occasional inaccuracies in the extracted data, which could potentially mislead the LLMs. The assumption is made that if a table contains essential information, it will likely be reiterated in the text, either in the same article or elsewhere, and can thus be more reliably extracted from textual content. These issues however point to one of the limitations of this approach that will have to be corrected in the future: it remains difficult to translate some existing knowledge into a text-mining format, and procedures to translate complex tables or complex maps should also be developed to translate this knowledge into text format for more efficient RAG.

4 Text splitting

Although recent LLMs have significantly large context sizes—far exceeding that of a single article, such as GPT-4 at 128k tokens or Gemini/Claude at 1M tokens—performance degradation with increased context size has been documented [13]. Strategies have been proposed to position the most relevant articles at the beginning and end of the context, based on findings that suggest the central part of the context is where issues predominantly arise [13]. However, other studies have yielded different conclusions, identifying the initial half of the context as problematic and thus casting doubt on the effectiveness of context reordering. More critically, when responses depend on multiple facts within a single document (referred as multi-fact retrieval), it has been observed that the efficacy of long context LLM approaches diminishes with the increase in fact numbers, a limitation not present in $1,000$ token approaches, which successfully retrieve all facts. Given that long context LLMs are costly, and both query price and execution time escalate with context size, the decision was made to continue with the conventional method of splitting articles into overlapping chunks.

The optimal granularity for segmentation remains to be determined. An initial approach might involve segmenting at the section or paragraph level. Experiments have revealed that current PDF loaders often struggle to accurately detect the structure of an article based on its sections, resulting in a significant proportion of potentially valuable articles being poorly processed. The paragraph-by-paragraph approach, which simply involves searching for double line breaks (denoted by \n\n), has been found easier to implement. However, tests tend to indicate that this granularity is too coarse for embedding calculations, as a single paragraph may contain multiple semantically distant pieces of information, especially in sections like the abstract, introduction, and discussion.

Conversely, reducing the text to 2-3 sentences risks severing these sentences from their contextual framework. For instance, if the GC content in a specific gene is calculated alongside a comparison for a neighboring species, the extracted text might lose the part indicating that the rate pertains to another species, potentially misleading the LLM. More precisely, when calculating embeddings, shorter texts of 2-3 sentences are sometimes preferable to capture specific semantics; however, for the LLM to formulate its response, providing texts of one or two paragraphs is advantageous to ensure no specific context is missed. These seemingly contradictory objectives can be achieved simultaneously by storing in the vectorstore the embedding of each pair or triplet of successive sentences, but associating it not with the text of these 2-3 sentences, but with that of the paragraph or paragraphs containing them. Thus, the search for information remains specific (semantics captured at the sentence level), while the LLM is provided with the necessary context (returning to paragraph level).

Various methods for text segmentation were considered possible. Clearly, improvements could be made beyond merely cutting every X characters. One promising approach involved semantic chunking, as implemented in LangChain library. This method entails calculating the embeddings for each sentence, then segmenting the text where two consecutive embeddings are sufficiently distinct. Although intriguing, this approach requires an embedding function capable of distinctly identifying the meanings specific to *Mycobacterium tuberculosis* research area, a capability not demonstrated by the best available OpenAI embeddings. Depending on the criteria used to determine whether two embeddings are sufficiently distinct, this method could result in segmentation by individual sentences, or in two or three large blocks of text, but nothing intermediate. This limitation could be attributed to the need for embeddings specifically fine-tuned to encode the semantics of *M. tuberculosis* for such an approach to be effective.

The most successful solution for our corpus has been to implement a classical recursive character text splitter, set with a chunk size of 500, an overlap of 200, and separators including ['\n\n', '\n', '.']. This splitter we considered, operates as follows:

- text is initially split along the first separator;
- each resulting extract is then processed:
  - if it is smaller than the chunk size, it is left as is;
  - otherwise, it is further split using the second separator;
  - this recursive chunking continues, utilizing the separators sequentially, until either the size is appropriate or it is the last separator used.

This approach highlights the challenges posed by Linnaeus' naming convention in text processing. Indeed, biologists traditionally adhere to the naming convention introduced by Carl von Linné, employing the genus and species to name bacteria (e.g., *Mycobacterium tuberculosis*, *Mycobacterium leprae*, *Yersinia pestis*), and italicizing these names. It is also common practice to use only the initial of the genus (e.g., *M. tuberculosis*, *Y. pestis*). However, this convention becomes problematic when articles are segmented into chunks for inclusion in databases.

Indeed, the use of the first letter of the genus (*M. tuberculosis* instead of *Mycobacterium tuberculosis*) risks splitting the text between genus and species, which is problematic in several ways. Firstly, such splitting occurs in the middle of a sentence, which contradicts the aim of recursive text splitting and inevitably leads to a loss of some semantics. Secondly, there is a risk of losing information regarding the bacterium's name to which the facts pertain, which is nonetheless valuable. Finally, segments of text referring to two bacteria (e.g., comparisons between *M. tuberculosis* and its probable closest neighbor *M.canetti*) are likely to be separated. To circumvent these issues, the text has been pre-processed to transform all condensed forms of species names into their extended form (from *M.leprae* to *Mycobacterium leprae*). This pre-processing has been applied to a large collection of bacteria that are either related to tuberculosis (particularly all bacteria of the *Mycobacterium* genus) or are commonly studied and thus may be mentioned periodically (*Yersinia pestis*, *Pseudomonas aeruginosa*, etc.). Finally, and more anecdotally, writing these species names in italics leads to a smaller space downstream, and therefore to the absence of detection of this space: the species names are often stuck to the words that follow them. All this has to be taken into account.

5 Selection of the vectorstore

Finally, the choice of a vectorstore is not arbitrary when the objective is to build systems embedding large language models in a research context. While the extraction of relevant text snippets is a priori the sole use of a vectorstore within the framework of retrieval-augmented generation, the search for articles requires the ability to query other fields. Various types of searches would be useful, such as:

- finding articles written by one or two given authors, potentially with misspellings in their names,
- specifying the date, the journal, potentially with errors or a "fuzzy" search (e.g., "an article written by a certain author around the 2010s"),
- searching for an article based on its approximate title, or based on its subject matter (e.g., "the recent article redefining the taxonomy of *M. tuberculosis* affecting cattle").

Feature	**ChromaDB**	**FAISS**	**Weaviate**	**Milvus**
Additional Fields Handling	No	No	Yes	Yes
Metadata Management	Limited	No	Yes	Yes
Support for Lists (e.g., Authors)	No	No	Yes	Yes
Hybrid Search (Embed. & Metadata)	No	No	Yes	Yes
Multiple Embeddings per Record	No	No	Yes	Yes
Sparse Embeddings Handling	No	No	Yes	Yes
Ease of Use for Researchers	Moderate	Moderate	Complex	Moderate
User-Oriented Documentation	Moderate	Moderate	Limited	Moderate
Industrial Target Suitability	No	No	Yes	Yes

Table 1: Comparison of Various vectorstores for research article storage

Therefore, it is not merely a matter of performing semantic search based on embedding similarity, but a similarity search that is contextualized and potentially complex. For the "publication date" field, a search for temporal proximity would be required; for the authors' field (a list of character strings), a high "edit distance" similarity with at least one element of the list would be needed (cross-similarity if more than one author is specified in the user's query); for the title field, several complementary approaches could be interesting: by edit distance (when the title is almost exactly provided in the query), by semantic similarity (when the query describes the subject of the article they are looking for), or by the presence of rare keywords ("The article defining lineage 10 of *M. africanum*", "The article explaining the role of the Rv0024 gene"); and for the abstract field, searching for an article based on its content (semantic similarity).

This implies several constraints on the choice of vectorstore. Firstly, it must be possible to store more than just text and its embedding, as the publication date, etc., is also needed. The author field must be able to contain a list of variable length (some specific articles have over a hundred authors). Storing the BibTeX, for instance in JSON format, would be appreciated. Most importantly, it should also be possible to perform searches based on criteria other than embedding similarity. Finally, it should be possible to integrate multiple different embeddings, typically dense and sparse, with the latter capable of outperforming TF-IDF for searches involving the importance of potentially rare specific terms (e.g., a gene name).

Classical vectorstores like ChromaDB, for instance, do not simply handle additional fields and do not allow document retrieval based on a combination of embedding similarity and other criteria (such as dates, etc.). Only one embedding per record can be stored, and it does not handle advanced types such as lists, which are useful for storing authors. Another commonly used vectorstore, FAISS (Facebook AI Similarity Search), a library developed by Facebook AI Research to facilitate efficient similarity searches among vectors in large datasets, is also unsuitable: it lacks native metadata management, as FAISS is ultimately specialized only for vector search.

In contrast, Weaviate and Vespa are far more advanced and allow the storage of complex objects and advanced querying in a hybrid manner (metadata and embeddings). However, these are complex libraries to manipulate, designed for industrial targets with needs that are more advanced (in terms of capacity, latency, deployment, etc.) than those of a researcher seeking to deploy tools enriched with LLMs. The complexity of these systems and their lack of user-oriented documentation make them unsuitable for this task. Among all popular vectorstores, only Milvus seems to have both the simplicity and advanced usage modes useful for the intended applications, see Table 1. It notably allows the integration of multiple embeddings, optimized handling of sparse embeddings, metadata management (possibly arrays), and hybrid search capabilities, as we will detail in the second article in this series, focusing on information retrieval itself.

6 Conclusion

In this article, the first in a planned series, the foundations for information retrieval in the context of LLM-enriched tools for tuberculosis researchers are established. The approach is designed to be general, with *M. tuberculosis* serving merely as an illustrative example. Based on our experience, the methodology for building a PDF database while respecting researcher constraints (such as not placing heavy demands on servers) is explained, including what information to extract and the techniques for extraction. Post-processing methods and effective text segmentation strategies are detailed, and the selection of the most suitable current vectorstore for these tasks is specified.

In future articles, the selection of embeddings and associated LLMs will be addressed, followed by the construction of graphs with nodes as agents equipped with LLMs, allowing iteration to provide optimal answers for researchers. Methods for integrating various embeddings and keyword searches will be specified, as well as the association of multiple LLMs with ad hoc prompts to achieve better results. The development of specific tools for certain tasks (e.g., obtaining precise information on a particular gene or bioproject through precise queries in an ad hoc SQL database or on the NCBI site) will be explored, along with the creation of custom agents equipped with these tools. Finally, the coupling of these agents to perform tasks significantly more complex than a simple RAG on a scientific corpus will be demonstrated.

Bibliography

[1] Grobid. https://github.com/kermitt2/grobid, 2008–2024.

[2] Anthropic. Claude. Chatbot platform powered by artificial intelligence, 2024. Accessed on May 22, 2024.

[3] Pau Casanova. doi2bib. https://github.com/PauCasanova/doi2bib, 2024.

[4] Cold Spring Harbor Laboratory. BioRxiv. The preprint server for biology, 2024. Accessed on May 13, 2024.

[5] Cold Spring Harbor Laboratory. MedRxiv. The preprint server for health sciences, 2024. Accessed on May 13, 2024.

[6] Cornell University. ArXiv. Repository for electronic preprints, 2024. Accessed on May 13, 2024.

[7] CrossRef. Crossref services. https://www.crossref.org, 2024. Access to DOI and metadata services for academic literature.

[8] Yusuke Shinyama et al. pdfminer: Tool for extracting information from pdf documents. https://github.com/pdfminer/pdfminer.six, 2024. Last access on 13 May 2024.

[9] Christophe Guyeux, Gaetan Senelle, Adrien Le Meur, Philip Supply, Cyril Gaudin, Jody E Phelan, Taane G Clark, Leen Rigouts, Bouke de Jong, Christophe Sola, et al. Newly identified mycobacterium africanum lineage 10, central africa. *Emerging Infectious Diseases*, 30(3):560, 2024.

[10] Adamandia Kapopoulou, Jocelyne M Lew, and Stewart T Cole. The mycobrowser portal: a comprehensive and manually annotated resource for mycobacterial genomes. *Tuberculosis*, 91(1):8–13, 2011.

[11] Le Chat Mistral. Le chat mistral. Chatbot platform powered by artificial intelligence, 2024. Accessed on May 13, 2024.

[12] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[13] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[14] National Center for Biotechnology Information, U.S. National Library of Medicine. Pubmed, 2024. Last access on 13 May 2024.

[15] OpenAI. Chatgpt-4. Language modeling software based on the GPT-4 architecture, 2023. Accessed on May 13, 2024.

[16] Aidan C Parte. Lpsn—list of prokaryotic names with standing in nomenclature. *Nucleic acids research*, 42(D1):D613–D616, 2014.

[17] Patricia Siguier, Jocelyne Pérochon, L Lestrade, Jacques Mahillon, and Michael Chandler. Isfinder: the reference centre for bacterial insertion sequences. *Nucleic acids research*, 34(suppl_1):D32–D36, 2006.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.