Quilliachat:

A Retrieval-Augmented Generation System for PDF Q & A

Sapienza University of Rome

Degree: **Applied Computer Science and Artificial Intelligence**

Author: **Joan Joseph Thomas** (2106939)

Email: **thomas.2106969@studenti.uniroma1.it**

Course: **AI Lab – NLP & CV**

Professor: **Prof. Daniele Pannone**

Date: **11/06/2025**

Abstract

This report defines the design, implementation and systematic evaluation of Quilliachat, a modular Retrieval Augmented Generation (RAG) system for question answer over pdf. It integrates chunking, retrieval and a user-friendly Streamlit interface. The evaluation was done as a comprehensive grid search over retrieval configurations on both a custom-labeled dataset and BEIR/nfcorpus benchmark, focusing on accuracy, speed and resources usage. Results show that dense retrieval with *all-MiniLM-L6-v2* achieves the best balance between accuracy and latency. Following the evaluation, this configuration was set as the default setup in the application.

1. Introduction

1.1 Motivation

The sheer number of digital documents have created a need for systems capable of providing information through natural language questions from large unstructured collection of texts. Traditional search methods are often keyword based and very primitive and are not context aware while the use of large language models may cause hallucinations and fail to provide grounded answers. A RAG (Retrieval Augmented system) solves this issue by combining the information retrieval along with LLMs so providing a hybrid balance of answers that are both relevant, easy to obtain and source grounded.

1.2 Project Contribution

This report details the design and subsequent evaluation of Quilliachat (a RAG pipeline for PDF question answering). -

- Designing a robust, modular pipeline for chunking, indexing and retrieval.
- Developing a GUI using Streamlit that enables users to upload pdf and get answers with the ability to select different retrieval configurations.
- Systematic evaluation of all major retrieval configurations using both a custom labeled dataset and the BEIR(nfcorpus) benchmark.
- Identification and setting of default configuration based on optimal balance between speed, accuracy and resource usage.

2. System Design and Implementation

2.1 System Architecture

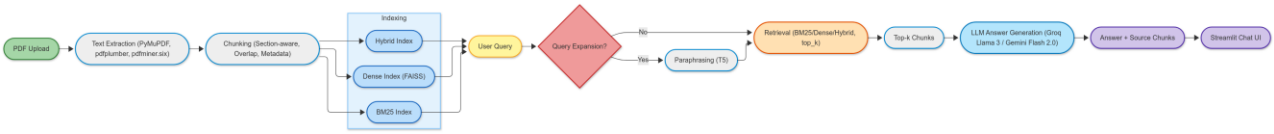

Figure 1 Quilliachat RAG pipeline

- **PDF upload:** using Streamlit interface, PDFs are uploaded.
- **Chunking:** Texts are extracted from PDFs and then split into logical, section aware chunks using a custom chunking script.
- **Indexing:** Chunks are indexed using various methods such as BM25, SentenceTransfomers(dense) and Hybrid mode.
- **Query Expansion:** user queries can be paraphrased using a T5 model optionally.
- **Retrieval:** using the selected configuration, system retrieves the top_k most relevant chunks.
- **Evaluation:** Retrieval accuracy, MRR, recall@k, latency and memory usage can be logged using evaluation script.

2.2 Core Components

2.2.1 Chunking

- **Libraries:** PyMuPDF, pdfplumber and pdfminer.six for extraction.
- **Strategy:** Section-aware chunking, max 300–500 tokens, 50-token overlap.
- **Metadata:** Each chunk contains metadata such as section, page, chunk ID, and document name.
- **Improvements:** additional pdf extraction libraries for fallback, improved section pattern recognition, and overlap logic were implemented to handle diverse PDFs.

2.2.2 Indexing

- **BM25:** *rank_bm25* tokenizes and indexes chunks for keyword-based retrieval.
- **Dense:** Embeds chunks using *SentenceTransformers* (all-MiniLM-L6-v2, all-mpnet-base-v2) with *FAISS* for fast vector search.
- **Hybrid:** Combines BM25 and dense scores for better retrieval.
- **Caching:** Chunks and embeddings are cached to improve speed by reducing the number of actions.

2.2.3 Retriever Implementation

- **BM25Retriever:** Fast, keyword-based using *rank_bm25*.
- **DenseRetriever:** Semantic, contextual, implemented with *SentenceTransformers* and *FAISS*.
- **HybridRetriever:** computation is weighted for both BM25 and dense to fetch better ranks.
- **Modularity:** The program is designed with modularity, each retriever is a separate class and new ones can be easily added.

2.2.4 Query Expansion

- **Model:** paraphrasing using T5.
- **Integration:** Query expansion can be optionally enabled by the user.

2.2.5 User Experience and Modularity

- **Streamlit UI:** - Users can upload PDFs select desired retrieval configs (retriever, dense model, top_k, query expansion) and chat with the pdf.
    - Results are fetched in a chat style interface along with the display of sources.
    - Retrieval settings can be user tuned.
- **Codebase organization:** - chunking/, retrievers/, indexers/, llm/, utils.py, main.py are all organized with modularity with the use of object-oriented programming. New retrievers, chunkers, or LLMs can be added or modified easily.

2.2.6 Evolution of Approach

- **Initial approach:** Considered the use of Word2vec for embeddings but was dropped due to the lack of semantic similarity and context aware retreival.
- **Improvement:** *Sentence-transformers* were adopted for context awareness and better results.
- **Chunking:** Chunking logic was improved to handle edge cases and real-world PDFs.

3. Evaluation Methodology

3.1 Custom Dataset

To validate and test the retrieval pipeline before testing with large-scale benchmark, a custom dataset was constructed. Five PDFs were carefully selected with a diverse range of domains –

- A call for applicants (*Call for applicants_Lazio_DIsco.pdf*),
- a legal regulation (*EU's data protection regulation.pdf*, i.e., the GDPR),
- a scientific climate report (*IPCC_AR6_WGI_SPM_final.pdf*),
- a programming book (*Python Crash Course_...pdf*),
- a research paper (*Attention is all you need_paper.pdf*),

to ensure the system was tested on academic, legal, scientific and research writing domains.

Each PDF was processed by the system's chunker and then five questions per document were manually labeled using a custom script. The labeling process included manually reading each chunk and selecting the chunk that contained the answer ensuring high quality ground answer.

A full gird search was run on this custom dataset evaluation all the possible combination of the configurations i.e (BM25, dense, hybrid, dense models - MiniLM-L6-v2, mpnet-base-v2), query expansion and different top_k values (3, 5, 10). This enabled a very detailed analysis of the performance across different configurations.

3.2 BEIR/nfcorpus Evaluation

To test the performance and validate the pipeline on a large dataset, the nfcorpus dataset from the BEIR benchmark was used with 3,633 passages, 3,237 queries and 324 relevant queries. Only the queries with qrels were evaluated. A grid search on all three retrievers, two dense models, query expansion enabled/disabled and top_k set at 5 were tested. The accuracy, MRR, recall@k, avg_rank, latency, memory and cosine similarity were checked. Queries were processed in large batches (100) for speed, accuracy and a specific number to prevent memory error.

3.3 Dataset Comparison

Dataset	**# PDFs**	**# Queries**	**# Chunks**	**Avg. Chunk Length**	**Domain**
Custom	5	25	~200	350 tokens	Academic, legal, scientific, technical
BEIR/nfcorpus	-	324	3,633	200–300 tokens	Biomedical

4. Results

4.1 Custom Dataset Results

A summary of the test done on the custom dataset with the grid search (total 36 configurations) few example data. The full results table is provided with the zip file containing code.

Retriever	Model	Query Expansion	top_k	Accuracy	MRR	Latency (s)	Memory (MB)
BM25		FALSE	5	0.92	0.86	0.0003	346
Dense	MiniLM-L6-v2	FALSE	5	0.72	0.55	0.85	1577
Hybrid	MiniLM-L6-v2	FALSE	5	0.84	0.9	0.98	2034
.....	.....	.....	.....	.....	....	.....	.....

PLOTS

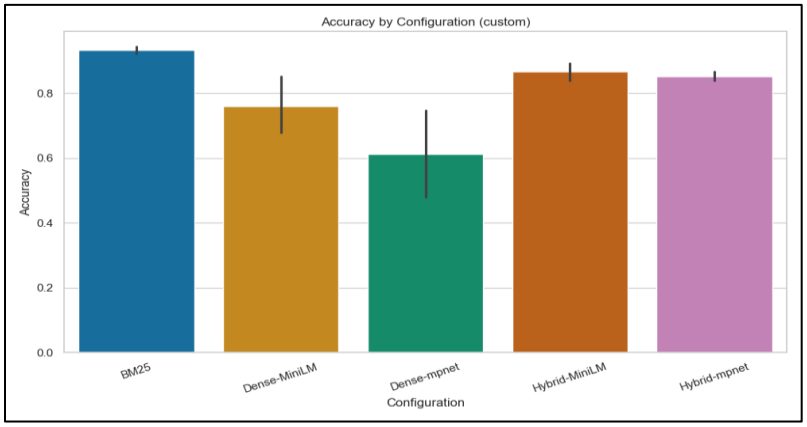

Figure 2 Accuracy by Configuration on custom dataset

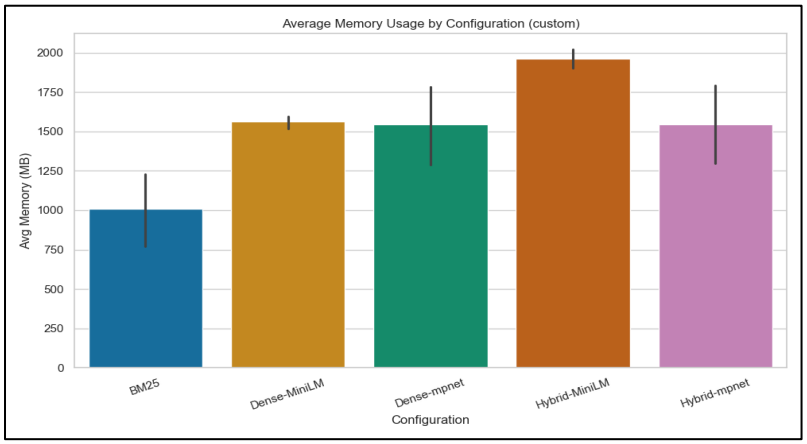

Figure 3 Average memory usage by configuration on custom dataset

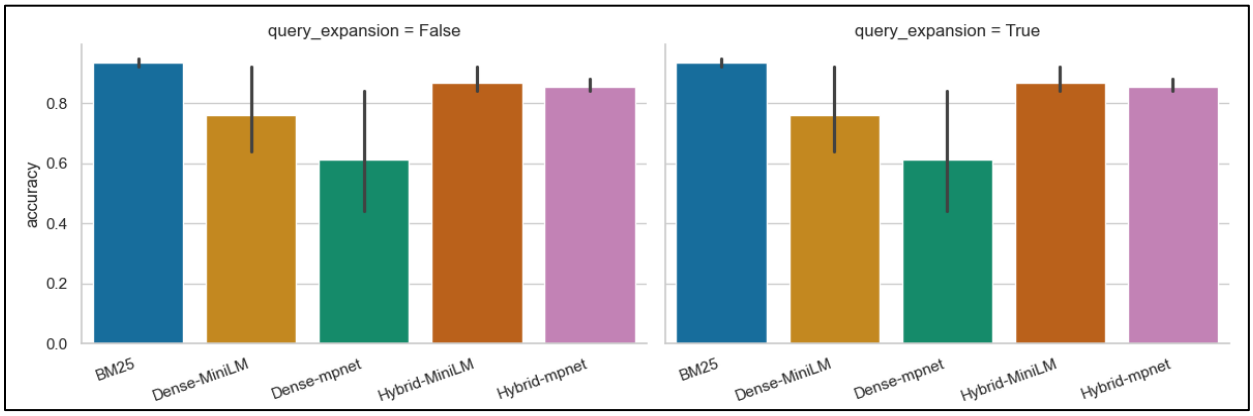

Figure 4 Accuracy by configuration faucet with query expansion

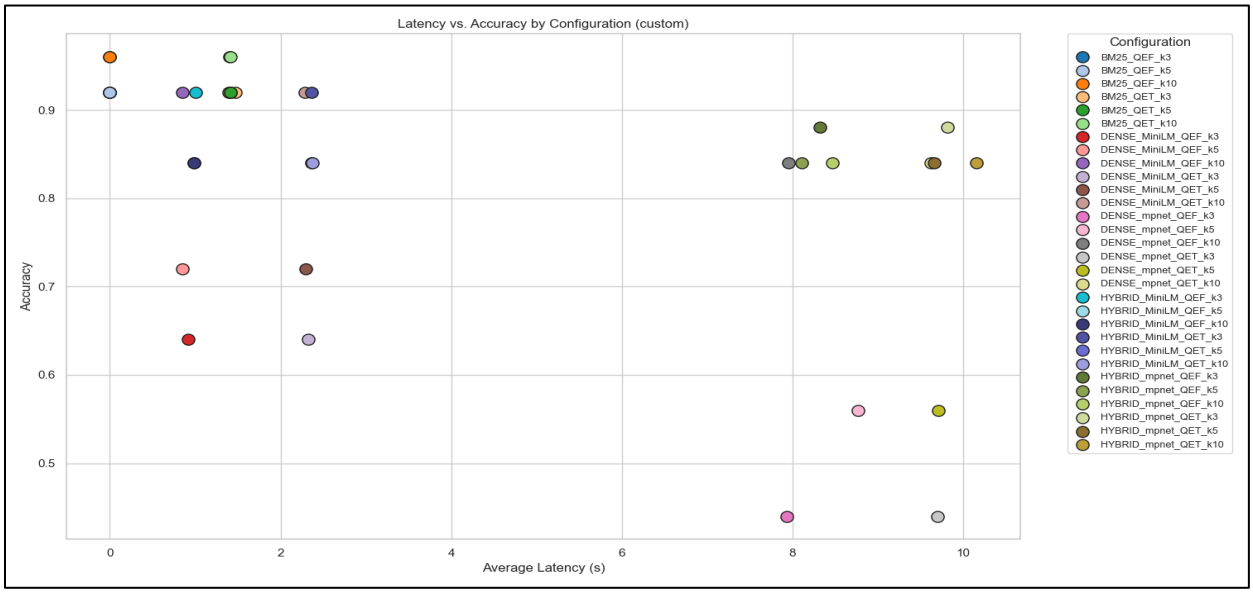

Figure 5 Latency vs Accuracy for custom dataset

Interpretation:

BM25 performed very well with accuracy around 0.92 – 0.96 which is likely due to the well-structured nature of the custom dataset. **Dense and hybrid** configs showed improvements at higher top_k values for more semantic or technical queries. **Query expansion** did not provide a significant benefit, but increased latency and memory usage. The grid search revealed that the best configuration for the custom dataset was hybrid retrieval with **MiniLM-L6-v2 at top_k=10**. As the number of top_k values increase the answer seems to be getting more accurate, but the resource usage will increase. So top_k value = 5 was selected as the default as a balance between speed and accuracy.

4.2 BEIR/nfcorpus Results

A summary of the test done on the Beir/nfcorpus dataset with the grid search (10 configurations). The grid search was done on the three different retriever options, two different dense models, query expansion enabled and disabled and top_value = 5.

retriever	dense_ model	query_ expansion	to p_ k	accuracy	mrr	recall_at _k	avg_rank	avg_ latency	avg_ cosine_ sim	avg_ memory _mb
bm25	-	FALSE	5	0.582043	0.783156	0.582043	1.728723	0.003628		683.9014
bm25	-	TRUE	5	0.582043	0.783156	0.582043	1.728723	0.00451		1198.964
dense	all-MiniLM-L6-v2	FALSE	5	0.625387	0.790017	0.625387	1.668317	0.009963	0.365242	1262.972
dense	all-MiniLM-L6-v2	TRUE	5	0.625387	0.790017	0.625387	1.668317	0.010008	0.365242	1267.781
dense	all-mpnet-base-v2	FALSE	5	0.625387	0.791502	0.625387	1.613861	0.040233	0.423283	1295.841
dense	all-mpnet-base-v2	TRUE	5	0.625387	0.791502	0.625387	1.613861	0.024063	0.423283	1427.923
hybrid	all-MiniLM-L6-v2	FALSE	5	0.603715	0.818547	0.603715	1.569231	0.017929	0.373255	1414.771
hybrid	all-MiniLM-L6-v2	TRUE	5	0.603715	0.818547	0.603715	1.569231	0.016254	0.373255	1462.19
hybrid	all-mpnet-base-v2	FALSE	5	0.613003	0.800758	0.613003	1.621212	0.047325	0.433706	2536.842
hybrid	all-mpnet-base-v2	TRUE	5	0.613003	0.800758	0.613003	1.621212	0.029694	0.433706	2694.791

PLOTS

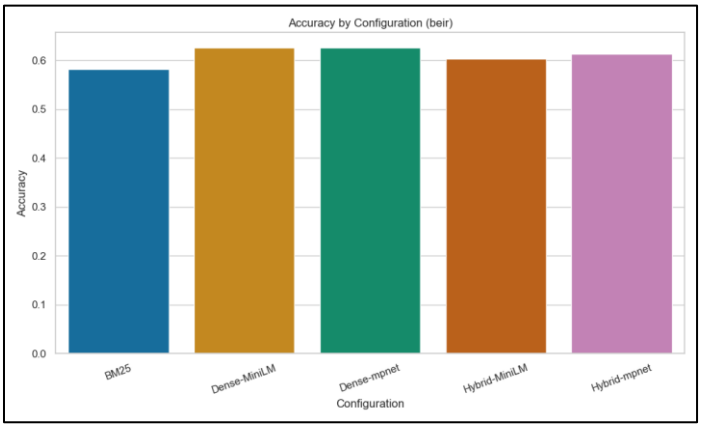

Figure 6 Accuracy by Configuration on BEIR dataset

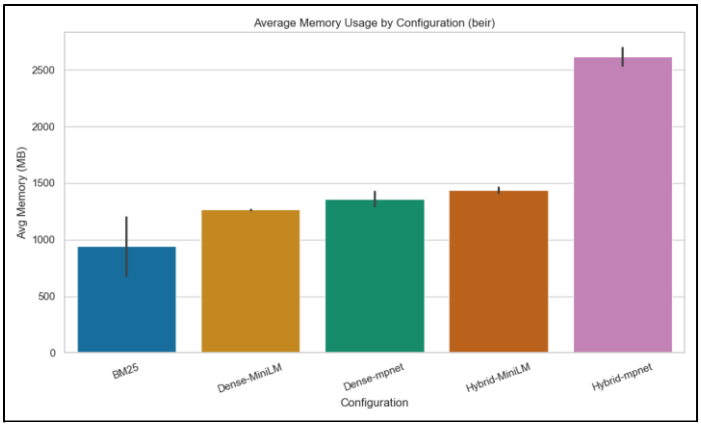

Figure 7 Average memory usage by configuration on BEIR dataset

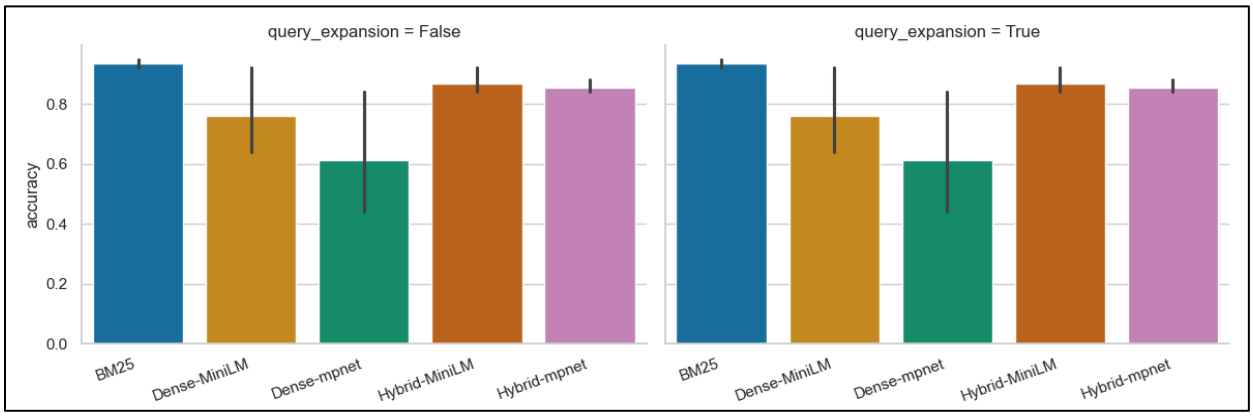

Figure 8 accuracy by configuration faucet by query expansion

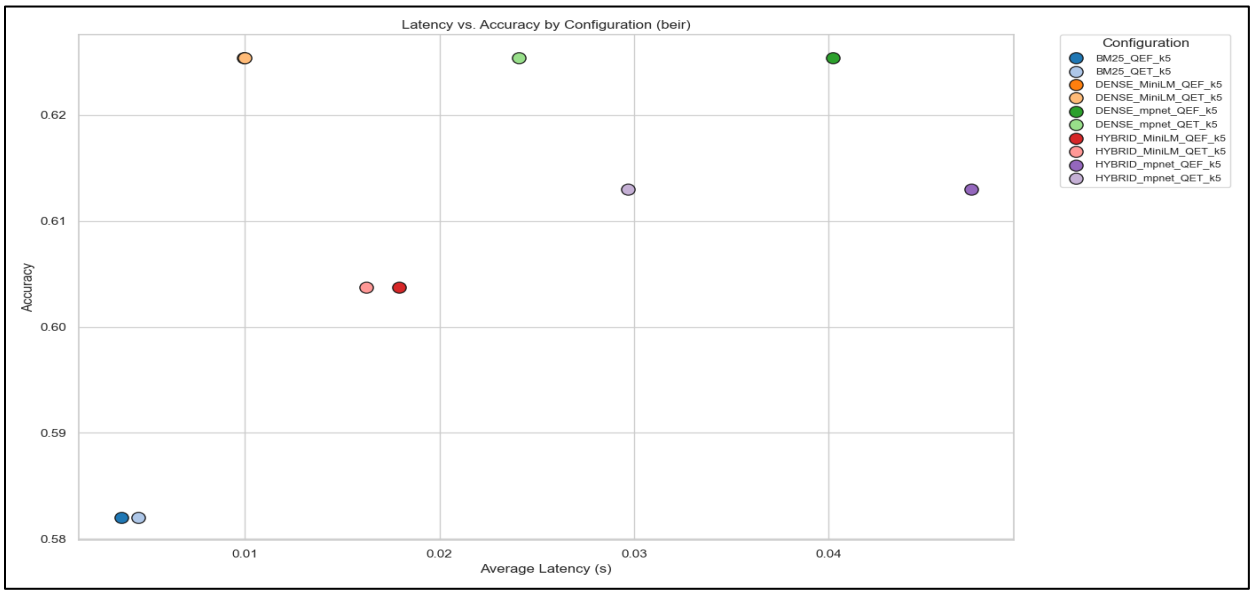

Figure 9 Latency vs Accuracy for BEIR dataset

Interpretation:

Dense and hybrid retrieval outperform **BM25**, especially for semantic queries. **Query expansion** does not improve accuracy for this dataset but increases resource usage. Latency is lowest for BM25, higher for dense/hybrid, but still acceptable for real-time use on GPU. Memory usage is highest for hybrid with mpnet-base-v2. The best configuration with a balance of speed, accuracy and system resource is **dense retrieval with MiniLM-L6-v2, top_k=5, no query expansion**. Based on this result, this configuration is set up as the default configuration in the Quilliachat with the ability to change it as per user needs.

5. Challenges, Limitations, and Future Work

5.1 Challenges & Solutions

During the development and evaluation of Quilliachat, several challenges were encountered and subsequently addressed. The extraction of clean structured text from diverse set of PDFs proved to be a very non-trivial task with many of them containing complex layouts, embedded images and inconsistent formatting. To address this, multiple extraction libraries were made use of such as PyMuPDF with fallback to pdfplumber, and pdfminer.six. The chunking logic was iteratively improved to handle overlapping texts, section headers etc. The evaluation of dense retrieval with large models and large datasets led to memory bottlenecks and out-of-memory errors which was solved by leveraging query batches, explicit cleaning of gpu memory after each configuration.

The project initially considered the use of word2vec for embeddings, but preliminary experiments and literature showed its various limitations and low accuracy especially in case of semantic retrievals. Thus, it was migrated to the use of sentence-transformers which provided better performance and flexibility.

5.2 Limitations

No fine tuning of the dense model was done, and all the retrievals are done using off-the-shelf models. Evaluation on a large dataset was limited to BEIR's nfcorpus dataset and results may differ slightly on other domains and document types. The LLM answer evaluation wasn't carried out and the focus was mainly on the evaluation of information retrieval. The system at the moment doesn't handle scanned image-based PDFs requiring OCR very well.

5.3 Future Work

The models could be fine tuned on specific domain to increase its performance and accuracy. A feedback model can be implemented to improve the models on the go based on user feedback and add chunk highlighting and retrieval score display to improve user trust and explainability. Testing on additional BEIR datasets and real-world documents be done to assess the generalizability. OCR support for parsing and extracting image-based pdfs can be implemented.

A full web-based interface (such as react and other front-end web development) with additional features such as pdf viewing with source highlighting can increase user experience.

6. Discussion & Conclusion

The comprehensive evaluation of Quilliachat across both a custom labeled dataset and BEIR's nfcorpus benchmark provides several key insights. **Dense retrieval with MiniLM-L6-v2, top_k=5**, and **no query expansion** consistently achieved the best balance of accuracy, speed, and memory usage. The configuration was thus set as default, but users retain the flexibility to change it anytime according to their preferences.

The results demonstrate that while **BM25** remains a strong baseline, especially for well structured, keyword rich documents, the **dense and hybrid methods** offer superior performance for more technical and semantic queries. Query expansion, despite on theory being better, did not yield any significant improvements for the tested datasets but increased latency and memory usage. Thus, it proves to be a bad option for the tests datasets.

The modular design of the Quilliachat codebase with clear separation of chunking, indexing, retrieval and UI scripts ensures that system is both practical and can be improved and extended easily.

In conclusion, Quilliachat represents a modular, extensible, user-centric RAG for PDF question answering. Its user-friendly interface, flexible configurations and display of source chunks for each answer promotes user transparency and trust. The systematic evaluation and testing of the retrieval system ensures that the best configuration with balance of speed and accuracy is enabled by default.

7. References

[1] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401.
[2] Thakur, N., et al. (2021). BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663.
[3] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084.
[4] Ma, X., et al. (2021). A Replicable Analysis of Hybrid Search. arXiv:2106.00882.
[5] Nogueira, R., et al. (2019). Document Expansion by Query Prediction. arXiv:1904.08375.
[6] https://github.com/beir-cellar/beir
[7] https://github.com/UKPLab/sentence-transformers
[8] https://github.com/joanjosephthom/Quilliachat