PDF Retrieval Assistant
Project Report for NLP Course, Winter 2024

Hubert Bujakowski
Warsaw University of Technology
01161404@pw.edu.pl

Jan Kruszewski
Warsaw University of Technology
01161492@pw.edu.pl

Łukasz Tomaszewski
Warsaw University of Technology
01161604@pw.edu.pl

supervisor: Anna Wróblewska
Warsaw University of Technology
anna.wroblewska1@pw.edu.pl

Abstract

This paper presents an interactive research assistant based on Retrieval-Augmented Generation (RAG), developed to address the challenges of domain-specific knowledge retrieval and query generation in Natural Language Processing (NLP). The system combines dense vector embeddings with scalable vector stores and generative models to deliver precise, contextually relevant answers from a dataset of 155 academic lecture PDFs. Using FAISS for retrieval and Llama-based models for generation, the system demonstrates strong performance across multiple evaluation metrics, including Context Precision and Faithfulness. While the results are promising, further enhancements, such as dataset expansion and multilingual support, are proposed. This project highlights the potential of RAG frameworks to advance educational and research-oriented applications, bridging gaps between user queries and knowledge-rich data.

1 Introduction

Natural Language Processing (NLP) has become a pivotal technology in enabling intuitive and efficient interactions between humans and machines. Question Answering (QA) systems, a key application of NLP, facilitate access to relevant information by translating user queries into accurate, contextually appropriate responses. This capability is vital in domains such as education, healthcare, and customer service, where rapid access to precise information is essential.

Retrieval-Augmented Generation (RAG) frameworks represent a significant advancement in QA systems by integrating external knowledge retrieval with generative modeling. Traditional pre-trained models, such as GPT or BERT, are often constrained by static knowledge bases and require costly retraining to stay updated. RAG systems address this limitation by dynamically retrieving and incorporating domain-specific data, offering improved performance for specialized contexts.

In this project, we propose an interactive research assistant leveraging RAG principles. The assistant processes user queries using state-of-the-art retrieval and generative techniques, designed to efficiently extract and present information from an extensive dataset of academic materials. By combining advanced vector embeddings, scalable storage systems like FAISS, and cutting-edge LLMs, this system aims to enhance educational and research workflows, offering robust and adaptable capabilities for various user needs.

2 Literature Review

2.1 Question Answering Systems

Question Answering systems are a subset of Natural Language Processing applications designed to provide precise and contextual responses to user queries by leveraging structured or unstructured data sources. These systems aim to bridge the gap between human language and machine understanding, enabling intuitive interaction with large volumes of information [1].

There are two main QA Systems categories

- **Closed-Domain QA Systems:** These systems are designed to answer questions within a specific domain or context. They rely on a pre-defined dataset or knowledge base, offering highly accurate and domain-specific responses. Examples include medical QA systems or legal advisory tools.
- **Open-Domain QA Systems:** These systems can handle various questions across multiple

topics. They often use large-scale datasets, such as web pages or encyclopedias, to generate answers, prioritizing breadth over depth.

The architecture of Question Answering systems is typically comprised of several key modules: question processing, document retrieval, and answer extraction. Figure 1 illustrates these essential components.

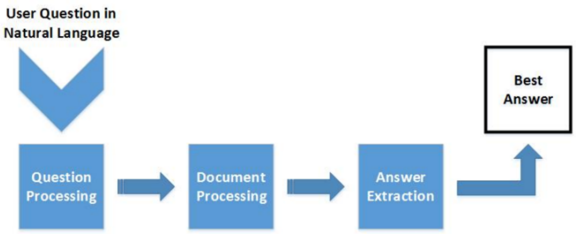

Figure 1: QA System Components. Source [18].

Question processing module is responsible for interpreting the user's natural language query and translating it into a structured format that can be effectively utilized by the **document retrieval module**. The document retrieval module identifies and retrieves candidate documents or information sources that are most likely to contain the relevant answers to the query. Finally, **the answer extraction module** analyzes the retrieved content, identifies the most relevant passages, and determines the best possible answers to present to the user [18]. This modular architecture ensures a systematic approach to transforming user queries into precise and contextually appropriate responses, facilitating the development of efficient and accurate QA systems.

Recent advancements in NLP, particularly in transformer-based architectures such as BERT [3] and GPT [12], have significantly enhanced QA system capabilities. These models leverage deep contextual embeddings to understand queries better and generate more accurate and human-like responses.

2.2 Retrieval-Augmented Generation

Traditional LLMs, such as GPT [12] and BERT [3] operate only on pre-trained parameters. These models cannot dynamically incorporate new information after training, which can lead to outdated and inaccurate outputs, especially in domain-specific context. Another issue is that retraining or fine-tuning of LLMs is costly, making them impractical for use cases like querying company-specific documents or other specialized knowledge bases, where the required information is not included in the data, on which the model was pre-trained. Retrieval-Augmented Generation addresses these issues by integrating external knowledge retrieval into the generative process.

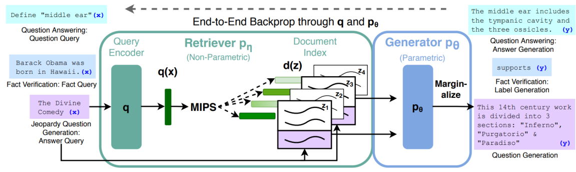

Figure 2: RAG overview. Source [6]

The concept of RAG was formalized by Facebook AI Research in [6]. This framework introduced a novel architecture that combined a dense retriever module with a generative model (Fig. 2). The retriever obtains relevant documents from data storage based on the user's query input. The obtained documents are then fed into a generative model that provides coherent and contextually relevant responses. This approach ensures the output is based on retrieved data and does not rely only on pre-trained knowledge.

2.2.1 Retrievers in Retrieval-Augmented Generation

Retrievers play a crucial role in RAG by fetching relevant information from external knowledge sources and providing domain-specific and up-to-date data to models. The retriever processes the user's query, searches a vector store, and identifies the most relevant documents to pass to the generative model.

Retrievers usually use sparse or dense retrieval techniques. Sparse retrievers, such as BM25, rely on traditional keyword-based matching, utilizing term frequency and inverse document frequency (TF-IDF) to rank documents based on the overlap of query terms with document terms. Dense retrievals, like Dense Passage Retrieval (DPR), use neural networks to embed both queries and documents into high-dimensional vector spaces, facilitating more context-aware matching. The retriever's ability to quickly select the most relevant documents has a major impact on the overall performance of the RAG system.

In addition to sparse and dense methods, there are hybrid retrieval approaches, which combine the strengths of both to improve retrieval performance. These methods combine keyword-based and semantic search through weighted aggregation to ensure that both lexical and semantic matching are considered. This hybrid approach is particularly beneficial when some query-document pairs rely heavily on exact term matches (e.g. company name), while others benefit from deeper semantic understanding.

2.3 Vector Store

Vector stores are an important component of RAG as they provide storage and retrieval mechanisms for documents that are used to enhance the generative process. Vector stores keep the embeddings of documents in a high-dimensional vector space, where each document is represented by a dense vector. The primary function of a vector store is to enable fast retrieval of documents based on the similarity between the query vector and the stored embeddings, which is calculated using, e.g. cosine similarity.

FAISS

FAISS (Facebook AI Similarity Search) is an open-source vector search library. It is designed to handle large-scale nearest neighbor search and is optimized for searching high-dimensional vectors.

Pinecone

Pinecone is a fully managed vector database designed for high-performance retrieval. It offers automatic scalability and is optimized for low-latency real-time search.

ChromaDB

ChromaDB is an open-source vector database designed for managing embeddings and performing similarity searches.

2.4 Large Language Models

Large Language Models (LLMs) are a class of deep learning models that have revolutionized the field of Natural Language Processing by demonstrating remarkable proficiency in understanding, generating, and manipulating human language. These models, primarily based on transformer architectures, leverage self-attention mechanisms to capture intricate relationships and dependencies within the text, enabling them to process long-range context and handle diverse linguistic structures.

LLMs are typically pre-trained on vast and diverse text corpora, often sourced from books, websites, and other large-scale data repositories. This extensive training allows them to learn a broad range of language patterns, grammar, facts, and reasoning abilities, which enables them to perform a variety of NLP tasks with little to no task-specific supervision. Some of the common NLP tasks that LLMs excel at include:

- **Translation:** Translating text between different languages while maintaining meaning and fluency.
- **Summarization:** Condensing large volumes of text into concise and meaningful summaries.
- **Information Retrieval:** Extracting relevant information from large datasets or text corpora based on user queries.
- **Conversational Interactions:** Engaging in an interactive dialogue with users, providing contextually appropriate responses.

The effectiveness of LLMs stems from their ability to filter large volumes of text data, summarize key points, and find patterns that would take humans much longer to identify. They operate by encoding input text into high-dimensional embeddings, which capture the semantic meaning of words and phrases. These embeddings are then decoded into output text, whether in the form of direct answers, summaries, or translations.

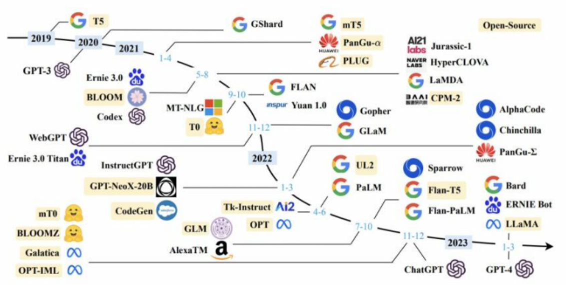

Figure 3: LLMs timeline.

Figure 3 presents a timeline of LLMs. Notable examples include OpenAI's GPT series (e.g., GPT-3, GPT-4), Meta's LLaMa [17], and T5 (Text-to-Text Transfer Transformer). These models have set new benchmarks across a variety of NLP tasks and have significantly advanced the capabilities of AI in natural language understanding and generation.

Large Language Models, with their ability to capture intricate language patterns through training on vast textual datasets, have demonstrated remarkable flexibility in performing zero-shot and few-shot learning. These models can generalize to new tasks with minimal task-specific data or examples, significantly reducing the reliance on annotated datasets. This capability makes LLMs particularly valuable for real-world applications, especially in domains where obtaining annotated data is scarce or expensive.

2.5 Text Feature Embedders

Text feature embedders are crucial components in Natural Language Processing systems. These embedders convert raw text into numerical dense vectors of fixed-length and low-dimension, called embeddings, which capture the semantic meaning of words, phrases, or entire documents. The primary goal of text embeddings is to map text into a high-dimensional vector space where semantically similar pieces of text are represented by vectors that are close to each other. Word embeddings, in particular, are fundamentally a form of word representation that links human understanding of knowledge meaningfully to machine interpretation. These embeddings can be seen as a set of real numbers (a vector) that represent the scattered depiction of text in an n-dimensional space, attempting to capture the meanings of words [15].

Text embeddings allow models to process and compare text efficiently by providing a dense and continuous representation of language, as opposed to sparse, discrete representations like bag-of-words. The key advantage of text embeddings is that they encode rich semantic information, which is essential for various NLP tasks. Commonly used text embedding techniques include:

- **Word Embeddings:** Methods like Word2Vec [7] and GloVe [9] represent words as vectors, capturing semantic relationships such as synonyms and analogies.
- **Contextual Word Embeddings:** Models like ELMo [10] and BERT generate context-sensitive embeddings, representing words differently depending on their usage in a sentence.
- **Sentence and Document Embeddings:** Techniques like Sentence-BERT [14] provide vector representations for entire sentences or documents, enabling tasks like semantic search and document retrieval.
- **Transformer-based Embeddings:** Models such as BERT and GPT provide robust embeddings across different levels—word, sentence, and document—achieving state-of-the-art results for diverse NLP tasks.

The effectiveness of text feature embedders lies in their ability to represent complex linguistic structures in a compact vector form, which can then be used by downstream models for tasks like text classification, text clustering, sentiment analysis, information retrieval, question answering, dialogue systems, semantic textual similarity or item recommendation. In QA systems, text embeddings are used to represent both the user's query and the documents or knowledge base from which answers are retrieved. By comparing the embeddings of the query and the documents, the system can determine the most relevant information to answer the query accurately.

2.6 Datasets for RAG

Retrieval-Augmented Generation models combine the strengths of information retrieval and generative models. To effectively evaluate and benchmark these models, it is essential to use datasets that challenge both the retrieval and generative parts. Datasets for testing RAG systems may be selected from datasets created for different tasks, such as question answering, conversation, or knowledge-intensive tasks.

Question Answering Datasets

Question answering datasets are crucial for evaluating the retrieval and generation capabilities of RAG models. Example QA sets are:

- **Natural Questions [5]** - a dataset with questions from real users paired with Wikipedia articles.
- **Stanford Question Answering Dataset (SQuAD) [13]** - dataset with questions posed by crowd workers on a set of Wikipedia articles.
- **MS MARCO [8]** - dataset with real Bing questions and human answer.

Conversational Datasets

Conversational datasets test RAG's ability to perform retrieval and generation in the context of dialogue. An example of such a dataset is **QuAC [2]**, which contains information-seeking QA dialogs.

Knowledge-Intensive Datasets

Knowledge-intensive tasks involve the usage of external knowledge to generate answers. A benchmark from this domain is **KILT (Knowledge Intensive Language Tasks) [11]**, which consists of 11 datasets representing 5 types of tasks: fact-checking, entity linking, slot filling, open domain QA, and dialog generation. An example dataset from this benchmark is **FEVER [16]**, which is an example of fact-checking dataset

2.7 Open-source RAG tools

Several open-source tools and frameworks have been developed to enable the implementation of RAG.

Haystack

Haystack is an open-source framework for RAG. It supports multiple retrievers, like DPR and BM25, and generators, for example, GPT or BART.

LlamaIndex

LlamaIndex provides an interface for connecting LLMs with external knowledge bases.

LangChain

LangChain offers a modular framework for constructing LLM pipelines, including RAG systems.

2.8 RAG evaluation methods

Evaluation of the RAG model performance requires methods that assess both the retrieval and generation components. The most commonly used evaluation methods are presented below.

Retrieval quality evaluation

The first step in evaluating RAG is assessing the quality of the retrieval, which is responsible for identifying relevant documents. Standard retrieval metrics include:

- **Precision at K** - the proportion of correctly identified relevant items within the top-K retrieved results.
- **Recall at K** - the ratio of correctly identified relevant items in top-K results to the number of all relevant results.
- **Mean Reciprocal Rank (MRR)** - evaluates how quickly the system can show the first relevant item in the top-K results.

Generation quality evaluation

To evaluate the quality of the generated text, **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** can be used, which measures the quality of the summary by comparing it to summaries created by humans.

End-to-End performance evaluation

To evaluate the performance of the Retrieval-Augmented Generation pipeline, **RAGAs [4]** can be used. This framework provides a few evaluation metrics, which assess RAG. Example metrics are listed below.

- **Faithfulness** checks if the claims that are made in the answer can be inferred from the context.
- **Answer relevance** checks if the answer directly addresses the question.
- **Context relevance** checks if the context contains only relevant information to answer the question.
- **Answer Correctness** compares the generated answer to ground truth.
- **Semantic similarity** compares the semantic resemblance between the generated answer and the ground truth.

3 Methodology

The methodology for developing the interactive research assistant is structured to ensure robust, efficient, and user-centric functionality. This section outlines the key stages of the system design, implementation, and refinement, with a focus on the technical components and workflows that enable effective query processing and result retrieval.

3.1 System Architecture

The system architecture consists of two primary modules: (1) embedding generation and indexing and (2) query handling. Each module is designed to work cohesively to ensure seamless and accurate document search capabilities.

3.1.1 Embedding Generation and Indexing

To enable efficient retrieval, the system generates vector embeddings for the document content and stores them in a scalable vector store. The following processes are involved:

- **Embedding Generation:** To encode the semantic meaning of the provided documents, `BAAI/bge-base-en` model is used. The embeddings generated by this model enable context-aware querying.
- **Indexing in a Vector Database:** A high-performance vector database, FAISS, is utilized to store the embeddings. This database supports fast and accurate nearest-neighbor searches based on the Euclidean distance, enabling rapid retrieval of relevant document sections based on user queries.

The embedding generation and indexing processes are critical for achieving high-quality semantic search capabilities within the system.

3.1.2 Query Processing and Role of Large Language Models

The query processing module, in conjunction with Large Language Models, is central to understanding user queries and retrieving relevant document sections. The following processes ensure accurate and meaningful responses:

- **Semantic Query Understanding:** Users input natural language queries via an intuitive interface. The system leverages an LLM, such as `llama3`, to interpret these queries and match them against the document embeddings for relevant results.
- **Retrieval-Augmented Generation:** The system retrieves relevant document sections based on semantic similarity and uses the LLM to generate contextually appropriate responses. This process combines the precision of embedding-based retrieval with the flexibility of natural language generation.
- **Targeted Previews and Relevance Scoring:** Retrieved results are presented as brief previews, accompanied by relevance scores that quantify the match quality between the query and the document content. This helps users quickly identify useful information.

By integrating LLMs into the query handling process, the system achieves advanced natural language comprehension and dynamic interaction capabilities, ensuring users receive relevant and insightful search results.

3.2 User Interface

The interactive research assistant's user interface (UI), as shown in Figure 4, was designed to enhance usability and engagement while supporting efficient exploration of search results. Built using Streamlit, the UI offers a clear, functional layout that caters to both intuitive navigation and detailed exploration of retrieved content.

The UI consists of three main sections: an answer panel on the left, a results exploration area on the right, and a feedback/comment section at the bottom.

Left Panel: Answer Display

On the top side of the interface, users can input their queries in natural language. After processing the query through the Retrieval-Augmented Generation pipeline:

- The system displays a concise and contextually relevant answer derived from the retrieved document sections.
- Answers are presented in an easy-to-read format, allowing users to focus on the information most pertinent to their query.

Right Panel: Search Results Exploration

The right-hand side of the UI serves as a detailed results viewer. This section enables users to:

- **Scroll through Retrieved Results:** Each result corresponds to a relevant document section retrieved from the vector store. These results are presented alongside relevance scores, enabling users to gauge the quality of the match.
- **View Document Metadata:** For each result, metadata such as document title, page number, and source is displayed to provide additional context.
- **Preview Document Content:** The UI includes a preview of the document page associated with the retrieved result, allowing users to quickly verify the relevance and depth of the information.

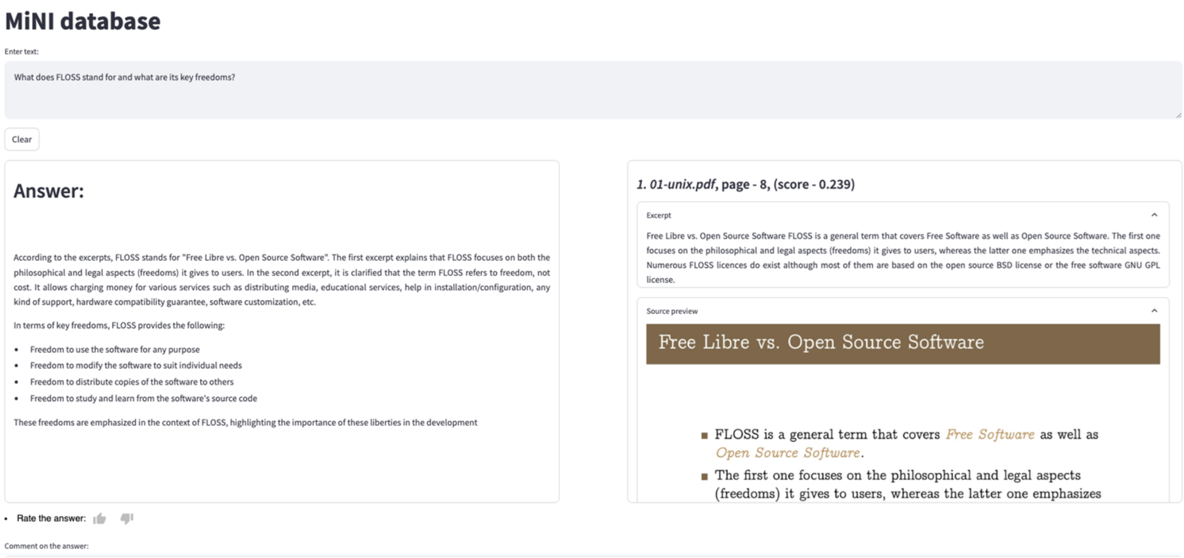

Figure 4: User Interface of the interactive research assistant.

Bottom Panel: Feedback and Commenting System

The feedback and commenting system, located at the bottom of the interface, plays a vital role in improving the system's performance and user satisfaction. Users can interact with the system's outputs in the following ways:

- **Reaction Options:** Users can react to the provided answers using predefined options, such as thumbs up or thumbs down. These reactions are logged for later analysis, providing insights into the perceived quality and relevance of the results.
- **Commenting Feature:** A text box allows users to leave detailed comments regarding the quality or relevance of the results. This feedback is also stored for potential use in refining the system.

While the current model is not directly fine-tuned based on the provided feedback, the collected data forms a valuable resource for future improvements. It can guide enhancements to the embeddings and optimization of search algorithms. By iteratively incorporating user feedback, the system is designed to evolve and better align with user expectations over time, ensuring long-term effectiveness and satisfaction.

3.3 Dataset

The dataset utilized in the project comprises lecture materials in PDF format, sourced from courses at the Warsaw University of Technology. These materials serve as the foundational knowledge base for the Retrieval-Augmented Generation system. By leveraging real-world educational content, the project aims to simulate domain-specific scenarios relevant to academic and professional contexts. In total, we collected 155 lecture pdf files.

The following courses provided the lecture PDFs used in the study:

- Big Data Analytics
- Fuzzy Reasoning
- Social Networks and Recommendation Systems
- Optimization in Data Analytics
- Deep Learning
- Data Storage in Big Data Systems
- Data Warehouses and Business Intelligence Systems
- Data Transmission
- Databases
- Introduction to Machine Learning
- IT Systems Engineering
- Operating Systems in Data Engineering

These lecture materials encompass a diverse range of topics, providing a rich and varied dataset for testing and benchmarking the system. The focus on technical and specialized content ensures that the RAG system is evaluated against challenging real-world data. This approach not only validates the system's retrieval and generation capabilities but also highlights its potential applications in educational and research settings.

3.4 Data Preprocessing

The data preprocessing module ensures that raw documents are transformed into structured formats suitable for embedding and querying. In the preprocessing pipeline, documents are sliced into one-page segments. Python libraries such as PyPDF2 and python-pptx are utilized to extract the text from the provided documents. The obtained text is divided using a sentence splitter and combined into chunks of a given size. These preprocessing steps ensure the documents are consistently formatted and prepared for subsequent embedding and indexing operations.

4 Experimetal Setup

4.1 Evaluation dataset

The evaluation dataset was created to test how well the Retrieval-Augmented Generation system performs. It includes 125 questions that were designed based on content from lectures. The questions cover a variety of topics and levels of difficulty to accurately evaluate the system's abilities.

To create reliable answers for comparison, ChatGPT was used to generate responses for each question. These answers were then manually reviewed to ensure they were accurate, clear, and relevant to the lecture content. This step was important to make sure the dataset was dependable and matched the information from the lectures.

In the end, the dataset consisted of question-answer pairs, where each question had a clear and verified answer. This dataset was used to test the RAG system's performance, helping us measure how well it retrieves and generates accurate responses.

4.2 Architecture Components Setup

Below are presented the key parameters for the main components of the architecture, chosen to keep the system running efficiently.

Faiss Vector Store

- Embedding Model: "BAAI/bge-base-en" on cuda with a dimension of 768.
- Indexing: Documents are chunked (size 256, overlap 25.6) and indexed using Faiss with a FlatL2 distance metric.
- Retrieval: Retrieves the top 2 most similar documents for each query.

Text Generation Model

- Model: "meta-llama/Llama-3.2-3B-Instruct" with 4-bit quantization on cuda.
- Generation Settings:
  - Max Length: 512 tokens (response).
  - Sampling: Enabled with top_p=0.9, top_k=50, and temperature 0.1.
  - Repetition Penalty: 1.1.

5 Results

To evaluate the performance of our RAG system, we calculated the RAGAs score. For this study, the calculations were performed on a subset of 30 queries extracted from the evaluation dataset. The metrics for the RAGAs score were calculated using the llama3 model, which served as the foundation for evaluating the system's outputs. Figure 5 presents the key metrics obtained from this evaluation.

These results reveal several key insights into the performance of our RAG system. First, the high Context Precision and Context Recall scores demonstrate the system's ability to retrieve relevant context with remarkable accuracy and completeness. The Faithfulness metric, while slightly lower, indicates a strong alignment between the generated answers and the retrieved information, suggesting that the outputs were predominantly factual and aligned with the source content. Lastly, the Answer Relevancy score, though slightly lower than the other metrics, reflects that the system generates answers that are mostly relevant to the user queries.

In summary, these results highlight the robust performance of our RAG system across multiple dimensions. However, there remains room for

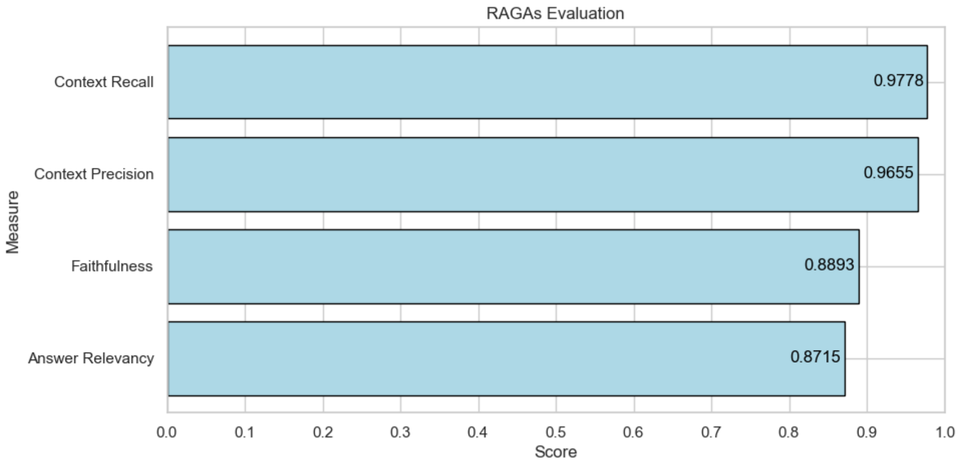

Figure 5: RAGAs score of our RAG system.

improvement in enhancing Faithfulness and Answer Relevancy, which could be addressed by further fine-tuning the generation component of the pipeline.

6 Future Work

The current project presents several opportunities for future development and improvement. The following key areas are proposed for further exploration:

1. **Fine-tuning the model:** Incorporate users' feedback to adapt the model for specific use cases, improving query relevance, precision, and overall user satisfaction.

2. **Dataset expansion:** Broaden the scope of the dataset to include a wider variety of academic topics and domains, ensuring applicability across disciplines. Additionally, introduce datasets in multiple languages to enhance multilingual support.

3. **Multimedia integration:** Develop capabilities for incorporating multimedia elements such as images and diagrams into the system, enabling richer and more engaging responses tailored to visual and interdisciplinary contexts.

7 Conclusions

This project has demonstrated the feasibility and effectiveness of integrating Retrieval-Augmented Generation frameworks into an interactive research assistant. By employing state-of-the-art technologies such as dense vector retrieval, scalable vector stores, and powerful generative models, the system achieved promising results in delivering contextually relevant and accurate responses.

Evaluation metrics, such as Context Precision and Faithfulnes, highlighted the system's strengths in retrieving and generating information aligned with user queries. However, areas for improvement, such as Answer Relevance, underscore the need for further fine-tuning and optimization. Expanding the dataset to cover broader domains and integrating multilingual support are key next steps to enhance applicability and inclusivity.

The interactive interface, designed with user feedback in mind, ensures an intuitive experience, fostering effective exploration of retrieved content. Future improvements will incorporate multimedia elements and leverage user feedback to iteratively refine the system, paving the way for more versatile and intelligent research assistance solutions.

References

[1] Sarah Saad Alanazi, Nazar Elfadil, Mutsam Jarajreh, and Saad Algarni. Question answering systems: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, 12(3), 2021. doi: 10.14569/IJACSA.2021.0120359.

[2] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Meth-*

ods in Natural Language Processing, pages 2174–2184, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1241.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

[4] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta, 2024. Association for Computational Linguistics.

[5] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7: 452–466, 2019. doi: 10.1162/tacl_a_00276.

[6] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[7] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12, 2013.

[8] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. 2016.

[9] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[10] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202.

[11] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.200.

[12] Alec Radford. Improving language understanding by generative pre-training. 2018.

[13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

[14] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410.

[15] S. Selva Birunda and R. Kanniga Devi. A review on word embedding techniques for text classification. In *Innovative Data Communication Technologies and Application*, pages 267–281, Singapore, 2021. Springer Singapore. ISBN 978-981-15-9651-3.

[16] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[18] F S Utomo, N Suryana, and M S Azmi. Question answering systems on holy quran: A review of existing frameworks, approaches, algorithms and research issues. *Journal of Physics: Conference Series*, 1501 (1):012022, 2020. doi: 10.1088/1742-6596/1501/1/012022.

A Reproducibility

	Description
Model Description	• FAISS with "BAAI/bge-base-en" embedding model with 768-dimensional embeddings • meta-llama/Llama-3.2-3B-Instruct
Link to Code	GitHub Repository
Infrastructure	• CPU: AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz • RAM: 16GB • NVIDIA GeForce RTX 3050 (CUDA 12.6) • Windows 11 • Python 3.11
Runtime Parameters	• Files to .pkl $\sim$ 15 seconds • Vector Store creation $\sim$ 8 minutes • Query $\sim$ 30 seconds
Parameters	• FAISS: embd_model_dim = 768, chunk_size = 256, chunk_overlap = 10% of chunk size • LLama: llm_context_window = 4096, max_length = 350, temerature = 0.1, top_p = 0.9, do_sample = True, top_k = 50, repetition_penalty = 1.1 • RAG: top_k = 1
Data Stats	• 155 pdf files from courses at the WUT • The number of pages vary from 6 to 98 pages. On average 38 pages
Data Processing	• Convert PDF files into `.pkl` format. • Use the `PyMuPDFReader` package to extract textual content from the raw data. • Split data into chunks using Sentence Splitter
Data Download	Lectures data
Data Languages	English

Table 1: Reproducibility

B Review answer

B.1 Answer to Jakub Piwko, Grzegorz Zakrzewski, Michał Gromadzki, Kacper Skonieczka

Our primary aim was to create a question-answering system, not a conversational chatbot. While the implementation allows single-turn interactions, this design aligns with the original purpose of the tool, which is to assist users in efficiently retrieving specific information from educational PDFs without introducing unnecessary complexity.

We fully agree with your suggestion to adopt newer LLM models, as advancements in this space have provided significant improvements in performance and context length. We will incorporate updated models like Llama 3.2 in our future releases to enhance system capabilities.

While we appreciate the suggestion to explore other embedding models, our current implementation has demonstrated excellent performance in terms of retrieval relevance and efficiency. Thus, we plan to continue with the current embedding model, as it meets the system's needs effectively.

B.2 Answer to Mikołaj Gałkowski, Mikołaj Piórczyński, Julia Przybytniowska

We acknowledge that the evaluation strategy requires further clarification. Although the paper discusses standard metrics such as Precision at K, Recall at K, and ROUGE, their application to assess our system's performance was not explicitly detailed.

However, we disagree with the statement that frameworks such as RAGAS were not mentioned. RAGAS was indeed referenced as part of the evaluation methodology, but we recognize that its integration into the evaluation process could have been elaborated further.

Additionally, we agree that testing variations in system components, such as chunking methods, embedding models, and generation LLMs, could yield valuable insights. However, given the time constraints of this project, we chose to focus on fine-tuning one set of methods to achieve optimal performance.

While the current system effectively handles the provided dataset, which primarily consists of PDF documents, we agree that handling more diverse formats or complex documents (e.g., PDFs with multimedia content like figures or charts) may pose challenges.

C Division of work

Task	**Team member**
Acquiring data and creating data set	Jan Kruszewski
Vector database creation	Hubert Bujakowski
Integration of LLM with vector database	Łukasz Tomaszewski
Model quantization	Hubert Bujakowski
UI design and implementation	Hubert Bujakowski
Add document presentation to UI	Łukasz Tomaszewski
Add feedback to UI	Jan Kruszewski
Create evaluation dataset	Jan Kruszewski
RAG evaluation	Łukasz Tomaszewski

Table 2: Work division