PDF Retrieval Assistant
Project Proposal 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 the initial phase of a project aimed at developing an interactive Retrieval-Augmented Generation (RAG) system. The system is designed to retrieve and interpret information from PDF documents, enabling users to pose domain-specific questions and obtain accurate, contextually relevant responses. The literature review examines the crucial components of the RAG system. The discussion further delves into essential tools such as vector stores (e.g., FAISS, Pinecone, ChromaDB) and open-source RAG frameworks like Haystack, LangChain, and LlamaIndex. The study also evaluates datasets and metrics for benchmarking RAG systems, ensuring robust assessment of both retrieval and generation components. Through this groundwork, the project aims to create an efficient and user-centric RAG solution.

1 Introduction

This document outlines the initial phase of a project for the Natural Language Processing course. The primary goal is to design and implement a system for the automated retrieval of PDF documents. The ultimate objective is to develop an interactive Question Answering (QA) system that enables users to ask domain-specific questions and receive accurate responses. The subsequent sections provide a comprehensive literature review and present the conceptual framework for the proposed solution.

2 Literature Review

2.1 Question Answering Systems

Question Answering (QA) systems are a subset of Natural Language Processing (NLP) 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 [2].

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 (QA) systems is typically comprised of several key modules: question processing, document retrieval, and answer extraction. Figure 1 illustrates these essential components.

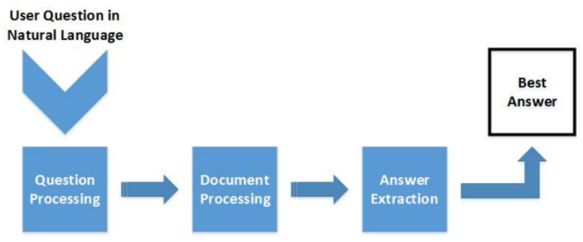

Figure 1: QA System Components. Source [23].

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 [23]. 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 [8] and GPT [17], have significantly enhanced QA system capabilities. These models leverage deep contextual embeddings to understand queries better and generate more accurate and human-like responses[13].

2.2 Retrieval-Augmented Generation (RAG)

Traditional LLMs, such as GPT [17] and BERT [8] 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 re-training 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 (RAG) addresses these issues by integrating external knowledge retrieval into the generative process.

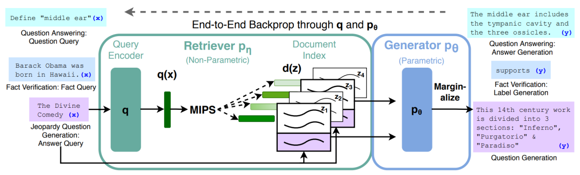
Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and p_θ
Query Encoder
Retriever p_η
(Non-Parametric)
Document Index
q
q(x)
MIPS
d(z)
z_4
z_3
z_2
z_1
Generator p_θ
(Parametric)
Margin-alize
p_θ
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

Figure 2: RAG overview. Source [10]

The concept of RAG was formalized by Facebook AI Research in [10]. 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 (LLMs)

Large Language Models (LLMs) are a class of deep learning models that have revolutionized the field of Natural Language Processing (NLP) 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 [12].

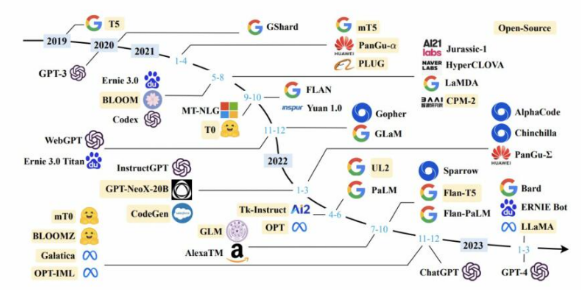

Figure 3: LLMs timeline.

Figure 3 presents a timeline of LLMs. Notable examples include OpenAI's GPT series (e.g., GPT-3, GPT-4), Meta's LLaMa [22], 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 (LLMs), 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[7].

2.5 Text Feature Embedders

Text feature embedders are crucial components in Natural Language Processing (NLP) 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 [24]. 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 [20].

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. [4]. Commonly used text embedding techniques include:

- **Word Embeddings:** Methods like Word2Vec [11] and GloVe [14] represent words as vectors, capturing semantic relationships such as synonyms and analogies.
- **Contextual Word Embeddings:** Models like ELMo [15] and BERT generate context-sensitive embeddings, representing words differently depending on their usage in a sentence.
- **Sentence and Document Embeddings:** Techniques like Sentence-BERT [19] 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. [4]. 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 (QA) datasets are crucial for evaluating the retrieval and generation capabilities of RAG models. Example QA sets are:

- **Natural Questions [9]** - a dataset with questions from real users paired with Wikipedia articles.
- **Stanford Question Answering Dataset (SQuAD) [18]** - dataset with questions posed by crowd workers on a set of Wikipedia articles.
- **MS MARCO [3]** - 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 [5]**, 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) [16]**, 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 [21]**, 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 [6]** 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 Solution Concept

The goal of this project is to develop an interactive research assistant designed to help university students and researchers quickly find relevant information in lecture materials. The system will enable users to query multiple PDF and PowerPoint files and retrieve specific answers, enhancing the research process and improving comprehension by providing targeted document previews. The interactive assistant will facilitate efficient navigation through the documents, show relevant sections, etc.

3.1 Key Features

The interactive research assistant will offer the following key features:

- **Query System:** Users can input questions in natural language. The system will use advanced Natural Language Processing techniques to extract relevant sections from the lecture materials, including PDFs and PowerPoints. The query system will support keyword-based searches as well as more complex semantic queries, providing a flexible interface for a variety of user needs.
- **Targeted Previews:** The system will retrieve and display relevant excerpts from the documents, providing a brief preview of each section along with a relevance score. The relevance score quantifies how closely the content of the retrieved section matches the user's query, helping users identify the most useful information at a glance.
- **Interactive Interface:** The user interface (UI) will be designed to be intuitive and minimalistic, enabling users to focus on querying

and retrieving results seamlessly. The UI will include:

  – A search bar for entering queries or keywords, allowing users to input their questions with ease.
  – A submit button to initiate the query and retrieve relevant results.
  – Future support for a feedback option, enabling users to provide insights on the retrieved results, helping to identify potential improvements and refine the system over time.

- **Feedback and Continuous Improvement:** The system will include a feedback mechanism allowing users to mark retrieved results as relevant or irrelevant. This feedback will be used to fine-tune the search algorithms, improving the system's performance over time through machine learning and reinforcement learning approaches.

3.2 Technical Approach

The technical approach to building this interactive research assistant involves the following key components. A detailed breakdown of the project timeline and its associated phases is provided in Table 1, while potential risks and their mitigation strategies are outlined in Table 2.

3.2.1 Document Embedding and Indexing

To enable efficient and meaningful search functionality, documents will be processed and transformed into embeddings before being indexed. This includes converting raw documents into structured formats compatible with advanced frameworks such as `llama_index`.

- **Embedding Generation:** NLP models such as Sentence Transformers or OpenAI's embeddings will be utilized to generate embeddings for document sections (e.g., paragraphs or slides). These embeddings will represent the semantic meaning of the content, enabling context-aware querying.

- **Efficient Retrieval:** For fast and scalable retrieval, a vector database such as Pinecone or FAISS will be used to store embeddings, enabling quick nearest-neighbor searches for the most relevant sections.

3.2.2 Data Preprocessing (Preliminary Step)

Before embedding and indexing, raw documents will undergo preprocessing to ensure compatibility with the system. This includes:

- **Text and Image Extraction:** Extracting textual and graphical content from PDFs and PowerPoints using specialized libraries.

- **Metadata Enrichment:** Capturing metadata (e.g., author, title, date) for better indexing and filtering.

These preprocessing steps ensure that all documents are ready for embedding and querying in a structured and consistent format.

3.2.3 Leveraging Large Language Models (LLMs)

Large Language Models (LLMs) will play a crucial role in enhancing the system's ability to understand user queries and generate precise responses. These models enable advanced natural language comprehension and support conversational interactions.

- **Probable Usage of LLMs:** The system is designed to integrate an LLM, such as `llama3`, to handle complex queries. Models like `llama3` are capable of interpreting nuanced user inputs and delivering contextually relevant results.

- **Instruction-Tuned Interactions:** User queries will be structured as prompts tailored to maximize the LLM's ability to comprehend intent, ensuring accurate and meaningful responses.

- **Multi-Step Reasoning:** For queries requiring sequential or logical reasoning, the system will leverage the LLM to chain multiple steps together, using tools like LangChain for orchestration.

By incorporating an LLM such as `llama3`, the system gains the potential to deliver enhanced semantic search capabilities and dynamic responses, ensuring users receive relevant and insightful results tailored to their research needs.

3.3 Expected Benefits

The interactive research assistant will offer numerous advantages for students and researchers:

- **Enhanced Efficiency:** The system allows users to quickly locate relevant sections within large lecture materials, significantly reducing search time.
- **Improved Comprehension:** By retrieving targeted previews and presenting relevant excerpts, users can better focus on the information they need.
- **Continuous Improvement:** A feedback-driven approach will ensure the system evolves to meet user needs more effectively, improving accuracy and user satisfaction over time.

4 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 (RAG) 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 lectures 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.

5 Proof of Concept

For the Proof of Concept stage, we developed a RAG system, which integrated FAISS for vector-based retrieval and Llama-3 for natural language generation. The architecture is as follows:

- **Vector Store**: The vector store is implemented based on FAISS and it uses the "BAAI/bge-base-en" [25] embedding model with 768-dimensional embeddings.
- **Retrieval**: The retival is performed based only on the dense approach. Indexing is done using IndexFlatL2 index, which measures the Euclidean distance between the query and all vectors loaded into the index. We retrieve only one chunk with the lowest Euclidean distance.
- **Generator**: For generation we use "meta-llama/Meta-Llama-3-8B-Instruct" [1] with a maximum response lenght equal to 350 tokens and temperature set to 0.1. The context window is set to 4096. As context we provide the whole slide, which is realted to retrieved chunk.
- **Data preprocessing**: We split documents into pages and extract text and additional metadata. The data is chunked into segments of 256 chunks with overlapping regions using sentence splitter.

Initial tests of our RAG yielded promising results, as the system was retrieving contextually relevant information and generated accurate responses. Run parameters can be found in Table 3.

References

[1] AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[2] 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.

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[4] Hongliu Cao. Recent advances in text embedding: A comprehensive review of top-performing methods on the mteb benchmark. *arXiv preprint arXiv:2406.01607*, 2024.

[5] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

[6] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

[7] Yang Gu, Hengyu You, Jian Cao, and Muran Yu. Large language models for constructing and optimizing machine learning workflows: A survey. *arXiv preprint arXiv:2411.10478*, 2024.

[8] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[9] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7: 453–466, 2019.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

[11] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781, 2013.

[12] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023.

[13] Kate Pearce, Tiffany Zhan, Aneesh Komanduri, and Justin Zhan. A comparative study of transformer-based language models on extractive question answering. *arXiv preprint arXiv:2110.03142*, 2021.

[14] 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.

[15] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[16] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

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

[18] P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[19] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[20] S Selva Birunda and R Kanniga Devi. A review on word embedding techniques for text classification. *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020*, pages 267–281, 2021.

[21] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[22] 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.

[23] FS Utomo, N Suryana, and MS Azmi. Question answering systems on holy quran: a review of existing frameworks, approaches, algorithms and research issues. In *Journal of Physics: Conference Series*, volume 1501, page 012022. IOP Publishing, 2020.

[24] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[25] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

A Work Plan Timeline

Phase	Tasks	Timeline
Preliminary Research and PoC Setup	• Research and evaluate existing NLP and embedding models for document indexing (e.g., Sentence Transformers, `llama_index`). • Experiment with vector databases (e.g., Pinecone, FAISS) for efficient embedding storage and retrieval. • Test LLMs (e.g., `llama3`) for query interpretation and response generation. • Collect and preprocess sample data, including text extraction and metadata enrichment from PDF/PowerPoint files. • Build a minimal working pipeline integrating LLMs, embedding generation, and retrieval for a Proof of Concept (PoC).	Nov 27 - Dec 11, 2024
PoC Validation and Expansion	• Validate PoC using a curated set of lecture materials, focusing on retrieval accuracy and relevance scores. • Refine preprocessing pipeline to handle edge cases (e.g., missing metadata, complex document layouts). • Enhance LLM query processing for semantic understanding and multi-step reasoning. • Test embeddings with diverse datasets to assess generalization and scalability. • Begin UI prototyping, including a basic search bar and results display.	Dec 12 - Dec 26, 2024
Full System Development	• Implement a user-friendly interface with interactive features (search bar, relevance scores, feedback option). • Scale document embedding and indexing to support large datasets. • Optimize query-processing pipeline for speed and accuracy using advanced LLM features. • Test multi-step reasoning and retrieval chaining using LangChain or similar tools.	Dec 27, 2024 - Jan 16, 2025
Testing and Finalization	• Conduct extensive testing of the system with university lecture datasets, evaluating accuracy and usability. • Address any performance bottlenecks or critical issues. • Finalize documentation. • Deliver the fully functional interactive research assistant.	Jan 17 - Jan 22, 2025

Table 1: Detailed Work Plan Timeline

B Risk Analysis

Risk	**Impact**	**Probability**	**Mitigation Strategy**
Limited time for PoC	Reduced scope for PoC	High	Focus on core functionalities: query processing and result previews.
Integration challenges	PoC instability	Medium	Use modular development to isolate components.
Incomplete preprocessing pipeline	PoC performance issues	Medium	Use small, clean datasets for the PoC phase.

Table 2: Risk Analysis for Project Timeline

C Reproducibility

	Description
Model Description	• FAISS with ”BAAI/bge-base-en” embedding model with 768-dimensional embeddings • meta-llama/Meta-Llama-3-8B-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 $\sim$ 1.5 minutes
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 3: Reproducibility