Project Mid-Progress Report

Development of an AI Agent for Research Paper Analysis and Question Answering

Name: Roshini Talluru, Tushar Jayendra Mhatre, Rahul Kataram

Semester: Spring 2025

Course: DSA 5900

Credit Hours: 4

Faculty Supervisor: Dr. Theodore Trafalis

Company and Sponsor (if applicable): N/A

Chapter 1: Introduction

Every year, with millions of scientific publications being issued daily, it has become incredibly challenging for researchers to summarize, explore, and gather the necessary information efficiently. One of the most significant hurdles is contextual analysis. Existing tools like ScholarAI, ChatPDF, and ExploreAI were able to solve this issue, but they still have notable limitations.

One of the primary concerns with publicly available AI models is data privacy and security. They require researchers to upload their documents to third-party servers, raising security risks when handling sensitive information such as unpublished research, proprietary datasets, or confidential academic material. Additionally, most large language models (LLMs) deployed in these tools are designed for general-purpose use. While they can handle many queries, they often fail to provide deep, structured, and research-focused insights.

Recent advancements in Natural Language Processing (NLP) and Large Language Models (LLMs) have introduced promising solutions to automate research analysis and improve information retrieval efficiency [1,2]. To address the shortcomings of existing tools, we will be developing an AI-based Research Paper Chatbot that uses Retrieval-Augmented Generation (RAG) to deliver accurate, contextual, and research-focused responses.

Unlike keyword-based search engines, by incorporating RAG methodology, our application can pull out semantically relevant passages from documents before generating responses, which ensures higher accuracy and contextual relevance [3]. Furthermore, integrating FAISS (Facebook AI Similarity Search) improves document retrieval efficiency, and using LLMs enhances intelligent response generation [4]. This blend of technology creates a user-friendly experience for users looking to analyze research papers efficiently.

Unlike cloud-based AI solutions, which require internet connectivity and expose data to external servers, we plan to design a system which can be operated fully offline or deployed on a local server, ensuring complete data privacy and user control. Another key aspect is using smaller yet efficient LLMs, which ensures seamless execution on user hardware without expensive infrastructure.

The system follows a structured pipeline to transform the data. Initially, research papers are ingested in PDF format, preprocessed, and embedded using SHA-256 hashing. This ensures that identical or highly similar content is not redundantly stored or processed. After preprocessing, we chunk the text into small parts and tokenize the chunks using pre-trained tokenizers, breaking them into smaller subword units compatible with transformer models. These tokens are then passed through Sentence Transformers [5] to generate dense vector representations that capture semantic meaning. FAISS, a high-performance similarity search library, stores these embeddings and enables fast retrieval based on vector similarity [6]. When a user submits a query, the system retrieves the most semantically relevant tokenized segments and uses a Large language model (LLM) to generate an accurate, context-aware response.

We used metrics like ROUGE, BERT Score, processing time, and model size to evaluate the system's performance. This will help us determine which metrics best enhance response accuracy and measure textual overlap [9,10]. As part of our experimentation, we plan to benchmark various open source LLMs—including Gemma3, Llama 3.1, Llama 3.2, Mistral, Qwen 2.5, and DeepSeek-R1-—to identify the best-performing models under different constraints.

Our system offers a secure, efficient, and research-oriented alternative to traditional AI tools. Combining on-device execution, advanced semantic retrieval, and customizable LLM selection enables researchers to extract structured insights from academic literature while maintaining full control over their data and computational environment. The following sections of this report will further elaborate on the dataset, methodology, evaluation framework, and final deliverables.

Chapter 2: Objectives

To implement this project, we established primary and specific objectives that help us understand our achievements and areas for improvement. The primary objective of this project is to enhance literature analysis, question-answering, and document summarization through the integration of Retrieval-Augmented Generation (RAG), FAISS-based document retrieval, and large language models (LLMs). By leveraging these advanced AI techniques, the system will generate contextually relevant and fact-related answers, streamlining the research process and improving information accessibility for researchers. To accomplish this, several technical key objectives have been set.

2.1 Technical Learning Objectives

Developing an AI assistant for ease in literature review: The first objective is to develop an AI-powered system that enhances research efficiency by automating the retrieval and summarization of scientific literature. The system will employ advanced NLP techniques to extract, clean, and index research papers into a vector-based storage system using FAISS, enabling high-speed similarity-based retrieval. Optimization techniques will be applied to improve the retrieval process through approximate nearest-neighbor search, ensuring fast and accurate document retrieval.

Generating high-quality research summaries: The second objective is to create accurate and concise research summaries. This is done by extracting meaningful text from documents and using retrieval techniques to find the most relevant content. These methods ensure summaries are coherent, factually correct, and focused on key research insights.

Comparative evaluation of large language models: The third objective is to conduct a benchmarking study comparing multiple LLMs, including DeepSeek R1, Gemma 3, Mistral, Llama 3.1, and Llama 3.2, to determine their effectiveness in scientific document analysis and question answering. The evaluation will incorporate metrics such as ROUGE and BERTScore to assess the accuracy and relevance. The insights from this evaluation will help optimize model selection for different research applications.

Designing an intuitive user interface for researcher interaction: The fourth objective is to design a user-friendly interface that allows researchers to upload research papers, submit queries, and receive structured answers in an accessible format. The interface will streamline interaction with the AI system, ensuring researchers can efficiently retrieve and summarize research findings without requiring deep technical knowledge.

Optimal Results through a smaller model: Our goal is to develop an optimal solution by focusing on the model's size and the number of parameters it uses. We will evaluate how efficiently a model can perform with a comparatively smaller number of parameters and a reduced model size. While many online models tend to be large and computationally expensive, we prefer creating a system that can provide high-quality responses using a smaller and less powerful model. This approach is more

cost-effective and reduces time consumption. This can be achieved through efficient prompt engineering and implementations of RAG methodology.

2.2 Personal Learning Objectives

This project provided us with the opportunity to explore new technologies. Previously, our projects focused primarily on supervised and unsupervised machine learning models. However, this project introduced us to various new concepts that required us to learn everything from scratch. We gained an understanding of the domain of generative AI and its implementation through various natural language preprocessing techniques and retrieval-augmented generation (RAG).

We expanded our understanding of different LLM architectures and honed our research skills. One of the standout lessons for us was the importance of effective communication. It's crucial for sharing insights and explaining technical concepts within our team. This emphasis on communication has helped us identify potential limitations and consider small adjustments that could strengthen our project, ultimately helping us connect with a broader audience. By integrating technical knowledge with practical application, this project equips us for real-world uses of AI in tasks like research assistance and automated literature reviews.

By achieving these objectives, we could implement a complete end-to-end approach and understand how to build and deploy an industry-level application. We aim to advance AI-driven research workflows, making literature reviews more structured, efficient, and scalable, while contributing to the broader field of AI-powered academic research tools.

Chapter 3: Data

3.1 Data Sources

The dataset for this project is a combination of research papers collected from publicly available repositories and academic databases, including arXiv, PubMed, and Nature. These sources provide a comprehensive range of scientific analysis across multiple domains, ensuring a diverse dataset. This selection of sources ensures the dataset includes papers from multiple domains, providing information retrieval and summarization robustness. The metadata available in these repositories aids in structuring the dataset for efficient query-based search and summarization tasks.

3.2 Raw Data

The dataset consists of 15 structured research papers collected from various academic databases mentioned above, along with a human-written summary for each paper, which we will set as a benchmark and reference for the model summarization. Among these, five papers are listed in our references, and the rest are taken from other sources, so our dataset will consist of papers from different domains. We wrote and verified the summaries with our project supervisor, Dr. Theodore Trafalis, an experienced researcher with immense literature writing and review knowledge. We created summaries in a structured format, which segmented the research paper into key sections. This includes a detailed breakdown of the abstract in simple terms, key references and how they relate to this work, how this work improves upon existing literature, a concise summary of the methodology used, and the main results and conclusions.

The table below gives a basic overview of the papers we have included in this study:

Table 3.1 Literature Paper Information

Paper No	Title	No. of Pages	No. of Words	PDF Size(KB)	Paper Source
1	Overlaps and distinctions between attention deficit/hyperactivity disorder and autism spectrum disorder in young adulthood	23	21854	822	PubMed Central (PMC)
2	Generative Adversarial Nets	9	4715	530	arXiv

3	Deep Residual Learning for Image Recognition	12	9085	819	arXiv
4	Quantum Machine Learning	24	9173	547	Nature
5	Attention Is All You Need	15	5644	2200	arXiv
6	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	16	9507	775	arXiv
7	A Comprehensive Survey on Graph Neural Networks	22	19986	1643	arXiv
8	A Survey on Transfer Learning	15	12684	2539	IEEE
9	Observation of Gravitational Waves from a Binary Black Hole Merger	16	11123	863	arXiv
10	THE FAISS LIBRARY	24	15956	1528	arXiv
11	Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report	37	9079	768	arXiv
12	Optimizing Domain-Specific	14	9289	1186	arXiv

	Image Retrieval: A Benchmark of FAISS and Annoy with Fine-Tuned Features				
13	CoFE-RAG: A Comprehensive Full-Chain Evaluation Framework for Retrieval-Augmented Generation with Enhanced Data Diversity	11	6395	536	arXiv
14	Retrieval-Augmented Generation for Large Language Models: A Survey	21	15405	1624	arXiv
15	Highly accurate protein structure prediction with AlphaFold	12	10214	3567	Nature

3.3 Exploratory Data Analysis and Evaluation Metrics

To enhance the performance of our application, we have ensured that the subset of papers it processes varies in structure, size, encoding, and format. This approach allows us to confirm that our application is robust enough to handle documents from diverse sources. We have visualized certain properties of the selected literature PDFs to provide an overview of the data we are working with. Figure 3.1 and Figure 3.2 showcase how we have selected PDFs of varying length, number of pages, and average words per page to comprehend how these factors affect the parsing time of these PDFs.

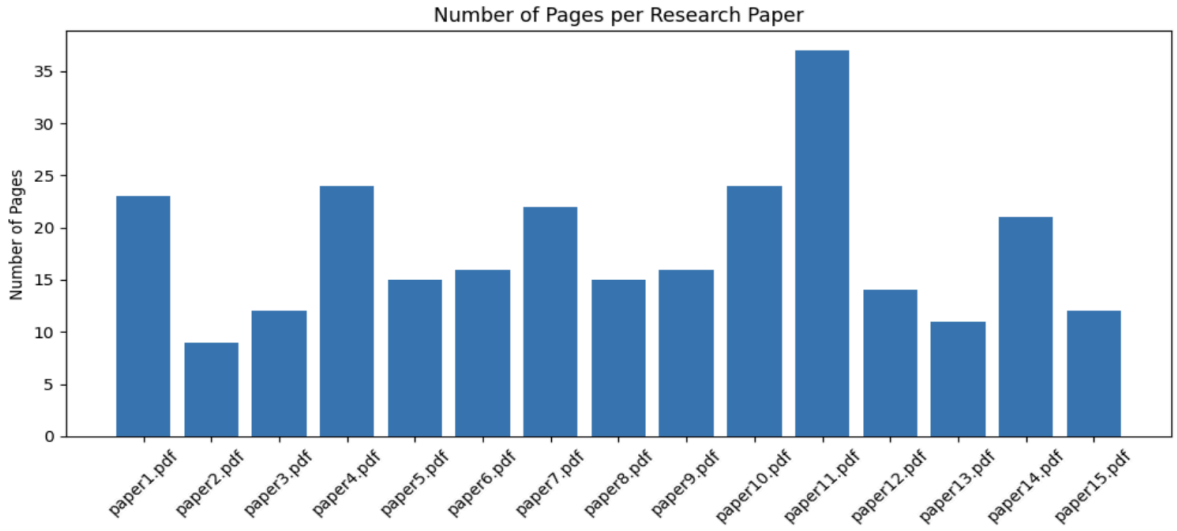


Figure 3.1 Number of Pages per Research Paper

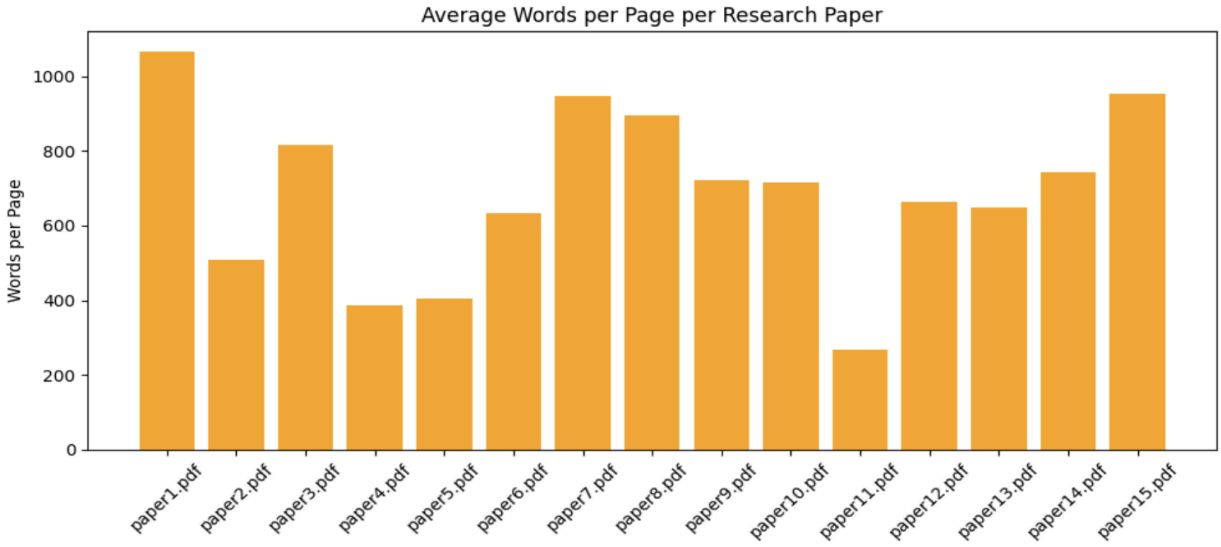


Figure 3.2 Average Words per Page per Research Paper

The scatter plot (Figure 3.3) below compares the PDF file size with the average words per page, giving us an idea of the tradeoff between their size and efficiency.

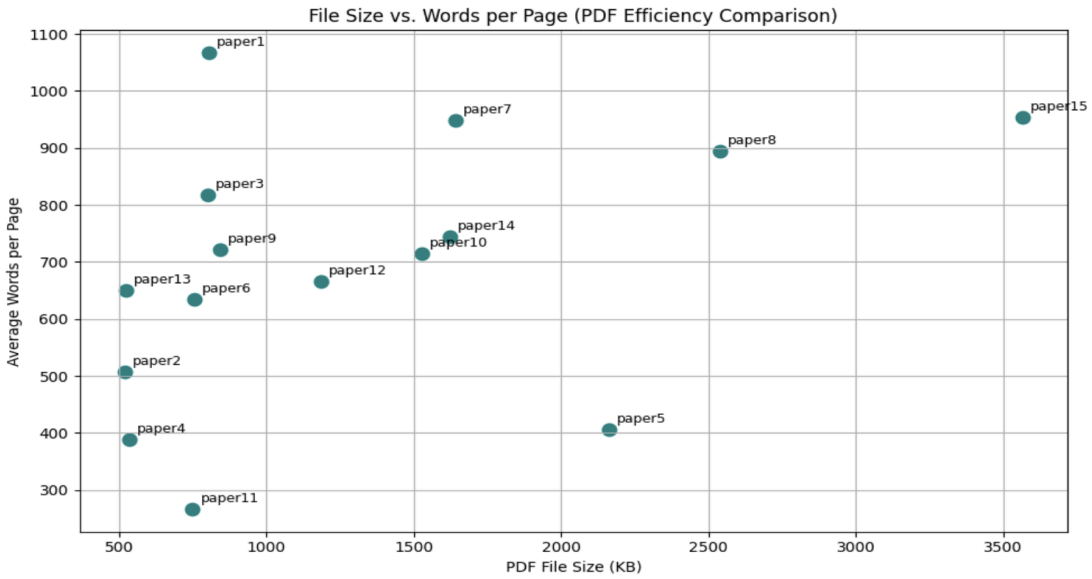


Figure 3.3 File Size vs. Words per Page (PDF Efficiency Comparison)

In Figure 3.4 below, we can see the parsing time of each PDF, which is the time the Python compiler takes to extract all the text from the PDF file. It is evident that the papers with high word density(number of words per page) generally have a higher parsing time, and a higher file size does not necessarily mean inefficient parsing. For example, Paper 15 is over 3.5 MB, 5 times larger than Paper 1, with an over 700 kb file size, but both PDFs have almost identical parsing times. Also, its parsing time is significantly lower than that of paper 14, despite being more than twice its size.

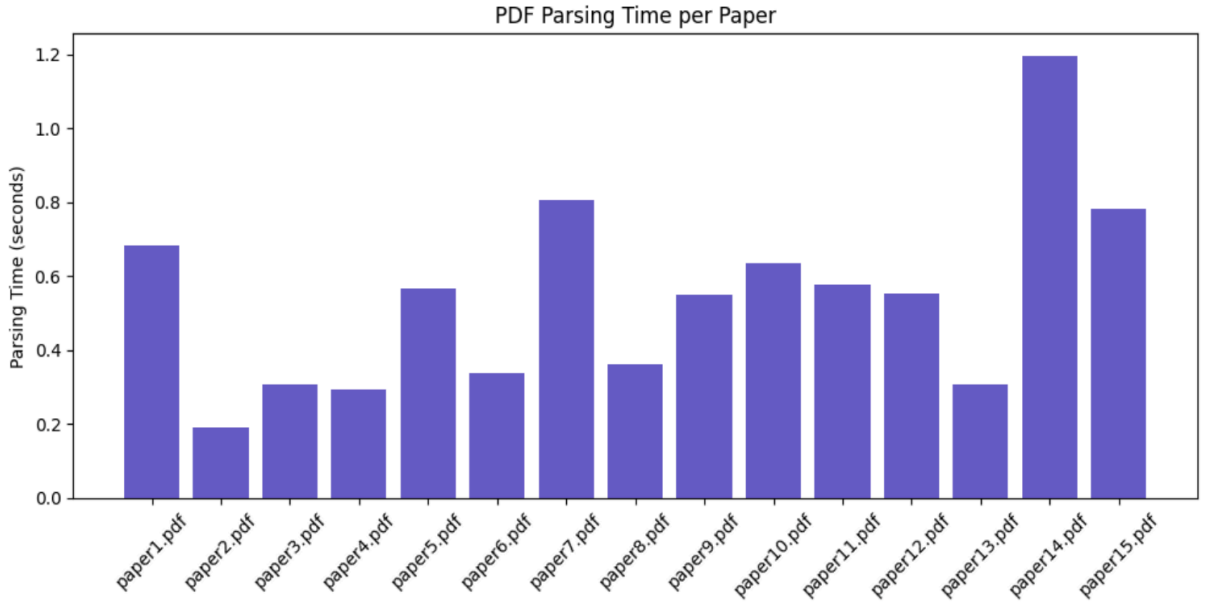


Figure 3.4 PDF Parsing Time per Paper

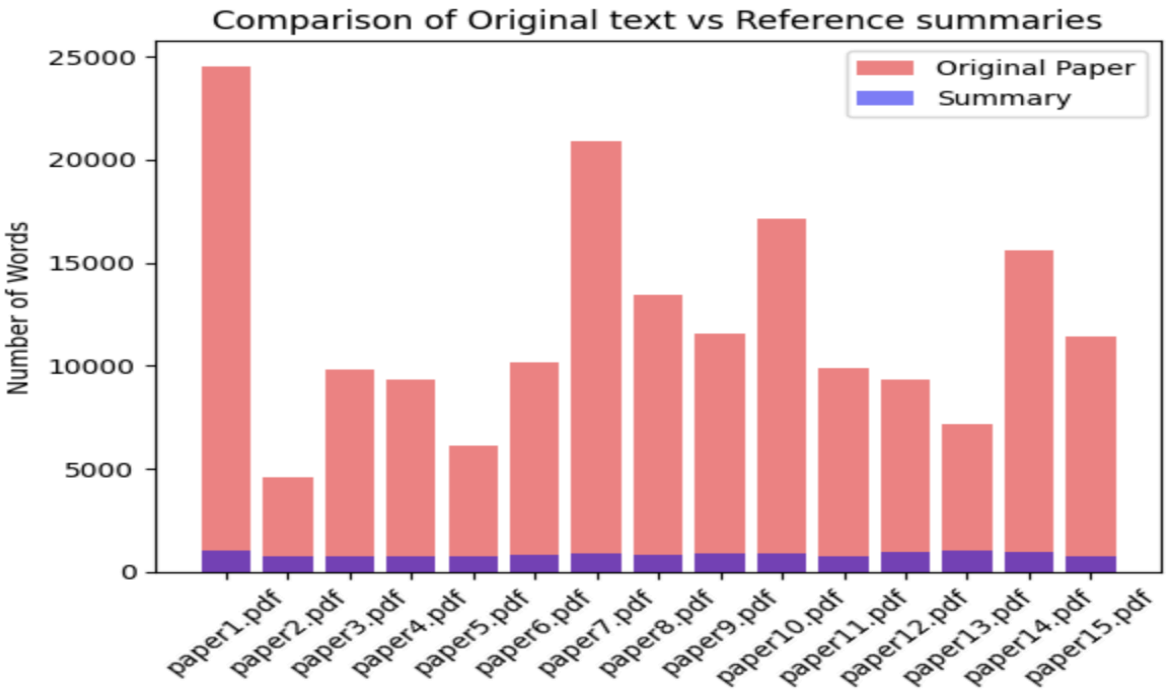


Figure 3.5 Comparison of original text vs Reference summarises

Figure 3.5 barplot gives us an overview of the length of the summaries compared to the length of the original text. The summaries were written while maintaining only a small proportion of the original text, as the goal was to focus only on key takeaways rather than include full discussions.

The following four metrics are evaluated to assess model performance:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE is a recall-based metric widely used for evaluating text summarization. It measures how much of the reference summary's content is captured in the system-generated summary. This is particularly useful in summarization tasks, where capturing the **essence** of the original text matters more than exact word matches. ROUGE is especially effective for extractive and abstractive summaries because it considers the words used and how they are arranged in meaningful sequences[8].

In our project, we focused on three commonly used ROUGE variants — **ROUGE-1**, **ROUGE-2**, and **ROUGE-L** — to comprehensively evaluate the quality of the generated summaries.

ROUGE-1: ROUGE-1 calculates the overlap of **unigrams** (individual words) between the candidate and the reference summary. It is a simple yet powerful measure of how much of the essential vocabulary from the reference is present in the generated output. The more unigrams that match, the better the recall[8].

$$\text{ROUGE-1}=\frac{Number\ of\ overlapping\ unigrams}{Total\ number\ of\ unigrams\ in\ reference\ summary}$$

This metric helps quantify basic content coverage and gives a clear idea of whether the generated summary includes the keywords from the reference.

ROUGE-2: ROUGE-2 evaluates the overlap of **bigrams** — sequences of two consecutive words between the generated and reference summaries. This allows us to assess whether the summary captures important **word pairs** or phrases, which often carry more meaning than individual words.

$$\text{ROUGE-2}=\frac{Number\ of\ overlapping\ biigrams}{Total\ number\ of\ bigrams\ in\ reference\ summary}$$

ROUGE-2 is more sensitive to word order than ROUGE-1, making it a better indicator of the **fluency** and **local coherence** of the summary.

ROUGE-L: ROUGE-L is based on the **Longest Common Subsequence (LCS)** between the candidate and reference summaries. It captures the most extended sequence of words that appear in the same order in both summaries, without requiring the words to be consecutive. This makes it ideal for evaluating **sentence-level structure** and logical flow[8].

To compute ROUGE-L, we calculate recall, precision, and an F1 score based on the LCS:

Recall: $R_{LCS} = \frac{LCS(X,Y)}{Length\ of\ reference\ summary}$

Precision: $P_{LCS} = \frac{LCS(X,Y)}{Length\ of\ candidate\ summary}$

F1 Score: $F_{LCS} = \frac{(1+\beta^2).R_{LCS}.P_{LCS}}{R_{LCS}+P_{LCS}.\beta^2}$

Typically, the β value is set high (such as 8) to give more weight to recall, ensuring that the generated summary retains the structure of the original text.

By using ROUGE-1, ROUGE-2, and ROUGE-L together, we can better understand and compare model performance in terms of both content inclusion and summary quality. ROUGE-1 tells us how much key vocabulary is included, ROUGE-2 shows whether meaningful word pairs are preserved, and ROUGE-L helps maintain the overall structure and flow of information.

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model developed by Google that has significantly advanced natural language understanding. Its core innovation lies in its deep bidirectional processing of text, where the model simultaneously considers both left and right context at every layer. This is enabled by the Transformer architecture and a pre-training strategy that includes two key tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)[9].

In the MLM task, random tokens within the input sequence are masked, and BERT learns to predict these masked tokens using the surrounding context. Formally, for a masked token tit_iti, BERT maximizes the probability:

$P(t_i|t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_n)$

This allows the model to understand word meaning in context robustly. The NSP task, on the other hand, trains BERT to determine whether two given sentences are sequential in the original text, thereby enabling it to capture sentence-level relationships and coherence.

BERT's ability to learn rich, contextual representations makes it well-suited for abstractive summarization and chatbot applications. Unlike models that rely on surface-level word matching, BERT captures the semantic meaning behind the text, allowing it to understand and generate summaries that preserve the original message even when rephrased.

Moreover, BERT underpins BERTScore, a modern evaluation metric that compares generated and reference texts based on contextual embeddings rather than exact word matches. In BERTScore, precision and recall are computed using cosine similarities between token embeddings, where cosine similarity measures how close two vectors are in direction, reflecting their semantic closeness.

Recall (R): R= $\frac{1}{|Y|}\sum_{y\epsilon Y}\sum_{x\epsilon X} max(cosine\_similarity(y,x))$

Precision (P): P= $\frac{1}{|X|}\sum_{x\epsilon X}\sum_{y\epsilon Y} max(cosine\_similarity(x,y))$

The **F1 score** is then calculated as the harmonic mean of precision and recall: $F1=2\times\ [\frac{(P\times R)}{(P+R)}]$

This embedding-based evaluation aligns especially well with the goals of our chatbot system, which aims to interpret and summarize complex academic texts in a human-like and semantically accurate manner. Overall, BERT's architecture and pre-training methodology make it a cornerstone for building intelligent, context-aware NLP systems.

Processing Time measures how quickly the model retrieves relevant document sections and generates summaries[10]. Lower processing times improve the real-time usability of the system, ensuring efficiency for large-scale applications.

Model Size (in GB) refers to the model's computational cost and memory footprint. Larger models may provide better accuracy but require higher computational resources, impacting scalability [2,3]. In our model evaluation phase, these models' size and system requirements were significantly evaluated in terms of their trade-offs between performance and efficiency.

3.4 Data Preparation and Preprocessing

The dataset undergoes a structured preprocessing pipeline for efficient document retrieval and summarization. The pipeline involves text extraction, duplicate removal, chunking and segmentation, tokenization, embedding generation, vector indexing, and text normalization. As explained below, each step is crucial in optimizing the data for accurate and scalable retrieval.

Text Extraction: Text extraction involves retrieving full-text content from research papers, particularly PDFs, and making it accessible for analysis. To extract the raw text, we primarily use the PyMuPDF package from Python, which extracts the whole text of the PDF without losing any information. However, before prompting the text into the Large Language Model, we modify the raw extracted text with more context by giving it an appropriate structure. For this, the algorithm first extracts the content. Then it identifies prominent features, such as titles, authors, references, and other section headings, using a combination of standard academic section names and numbering systems as references. We look for these headings in the raw text, and when a heading is found, the code checks the font size and text style and verifies whether the heading is followed by a numbering system, which will further refine the segmentation. Once each textual block is associated with its appropriate heading, with the help of Python dictionaries, we store the results in a structured format. Figures 3.6 and 3.7 below illustrate our extracted raw data, structure, and labeled text.

```
[25]: 'Deep Residual Learning for Image Recognition\nKaiming He\nXiangyu Zhang\nShaoqing Ren\nJian S
      un\nMicrosoft Research\n{kahe, v-xiangz, v-shren, jiansun}@microsoft.com\nAbstract\nDeeper neu
      ral networks are more difficult to train. We\npresent a residual learning framework to ease the
      training\nof networks that are substantially deeper than those used\npreviously. We explicitly
      reformulate the layers as learn-\ning residual functions with reference to the layer inputs, i
      n-\nstead of learning unreferenced functions. We provide com-\nprehensive empirical evidence s
      howing that these residual\nnetworks are easier to optimize, and can gain accuracy from\nconsi
      derably increased depth. On the ImageNet dataset we\nevaluate residual nets with a depth of up
      to 152 layers-8×\ndeeper than VGG nets [41] but still having lower complex-\nity. An ensemble
      of these residual nets achieves 3.57% error\non the ImageNet test set. This result won the 1st
      place on the\nILSVRC 2015 classification task. We also present analysis\non CIFAR-10 with 100 a
      nd 1000 layers.\nThe depth of representations is of central importance\nfor many visual recogn
      ition tasks. Solely due to our ex-\ntremely deep representations, we obtain a 28% relative im-
      \nprovement on the COCO object detection dataset. Deep\nresidual nets are foundations of our s
      ubmissions to ILSVRC\n& COCO 2015 competitions1, where we also won the 1st\nplaces on the task
      s of ImageNet detection, ImageNet local-\nization, COCO detection, and COCO segmentation.\n1.
```

Figure 3.6 Extracted Raw Data

```
Preamble: Deep Residual Learning for Image Recognition Kaiming He Xiangyu Zhang Shaoqing Ren J
ian Sun Microsoft Research { kahe, v-xiangz, v-shren, jiansun } @microsoft.com

Title: Deep Residual Learning for Image Recognition

Authors: Kaiming He, Xiangyu Zhang

Abstract: Deeper neural networks are more difficult to train. We present a residual learning fr
amework to ease the training of networks that are substantially deeper than those used previou
sly. We explicitly reformulate the layers as learn- ing residual functions with reference to t
he layer inputs, in- stead of learning unreferenced functions. We provide com- prehensive empi
rical evidence showing that these residual networks are easier to optimize, and can gain accur
acy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with
a depth of up to 152 layers—8 × deeper than VGG nets [41] but still having lower complex- ity.
An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result
won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10
with 100 and 1000 layers. The depth of representations is of central importance for many visua
l recognition tasks. Solely due to our ex- tremely deep representations, we obtain a 28% relat
ive im- provement on the COCO object detection dataset. Deep residual nets are foundations of
our submissions to ILSVRC & COCO 2015 competitions 1 , where we also won the 1st places on the
tasks of ImageNet detection, ImageNet local- ization, COCO detection, and COCO segmentation.

Introduction: Deep convolutional neural networks [22, 21] have led to a series of breakthrough
s for image classification [21, 50, 40]. Deep networks naturally integrate low/mid/high- level
features [50] and classifiers in an end-to-end multi- layer fashion, and the “levels” of featur
es can be enriched by the number of stacked layers (depth). Recent evidence [41, 44] reveals t
hat network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the ch
allenging ImageNet dataset [36] all exploit “very deep” [41] models, with a depth of sixteen [
41] to thirty [16]. Many other non- trivial visual recognition tasks [8, 12, 7, 32, 27] have a
lso 1 http://image-net.org/challenges/LSVRC/2015/ and http://mscoco.org/dataset/#detections-ch
```

Figure 3.7 Structured and Labelled Text

Hashing to avoid duplication: Duplicate detection is crucial for maintaining dataset integrity by eliminating redundant documents and improving storage efficiency. SHA-256 hashing generates a unique fingerprint for each document, ensuring even minor content changes produce distinct hash values[12]. Compared to other hashing algorithms like MD5, CRC32, or costly cosine similarity, which have higher collision risks, SHA-256 offers faster and scalable duplicate detection without embedding comparisons. It always creates the same hash for identical documents, enabling efficient indexing. This lowers the amount of computing resources needed and speeds up how quickly data can be retrieved.

```
        # Print the result
        print(f"{filename} → Hash: {file_hash}")
```

```
paper1.pdf → Hash: c25c5ed38c1d951a240b372c337188ff024958d9249cea5361ee64c2fae2862b
paper10.pdf → Hash: 376d9e8ce0bcbc0cbcdff9bbd081ec6425072a9affe9b49ffab42ec2232d40ac
paper11.pdf → Hash: 511afd0f2f8c48b7bcfd72711efa130d1621a20114ad50bd8887893284f2fc6e
paper12.pdf → Hash: 57f28047ebc78e38f16c617b67c24bd0d1b23ba1a82145f23067a7d5918f73fe
paper13.pdf → Hash: e5ba7e2ae7de6c5e9cf9e252884a0790638baf116590ed001ba7db8c8796e6c0
paper14.pdf → Hash: 396a0fadeb4cd40f5c8ccc36b73a0815f6cb4d7f6bfa53b6c48c1f9aba7c7e02
paper15.pdf → Hash: 6eae057a9faf4f671c3101e0745ed704460c6d3dec77243dfd3a9f2d2ab68970
paper2.pdf → Hash: ff5819e3a7b713c3bd3107b7de3d51fe0a347aa5d8444f0efdcf2345ef0a8b63
paper3.pdf → Hash: 1e0651b6810ecba34a3dbc5b5b0209226f889004607c1f203540a48d64e5a93a
paper4.pdf → Hash: f6d77381228b86d8609e7205129b39b1a7f7820db133b46b7dc0e4119846b69d
paper5.pdf → Hash: bdfaa68d8984f0dc02beaca527b76f207d99b666d31d1da728ee0728182df697
paper6.pdf → Hash: 5692a5514787a8c6727b4ff3b726a3385798bc68e12138d1d4af83947e2acf6e
paper7.pdf → Hash: de24ede0f541343b8bce39289120c835a9a2cc6a29e24877430f87f205e84280
paper8.pdf → Hash: bab71ea386b6b414cd0d863815baa31728e0925acf2329968337508d938c8da8
paper9.pdf → Hash: af436561e6c10988c206dddece92955d508377840473a6f2f22fd870c67106b7
```

Figure 3.8 Hash ID of all PDF files

Another significant advantage of using SHA-256 is that it strengthens data security and integrity due to its collision-resistant nature, preventing different documents from generating the same hash [12].

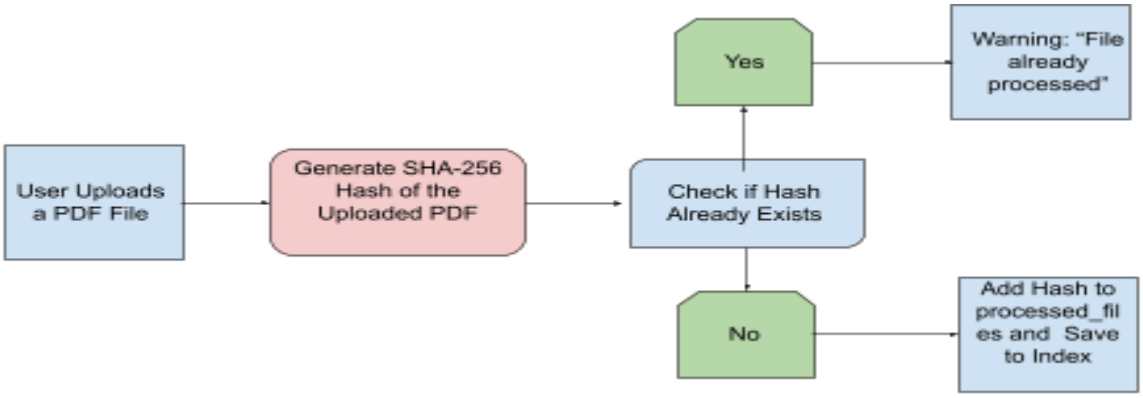


Figure 3.9 Flowchart of Hashing the PDF file

Chunking, Tokenization, and Sentence Embeddings: The whole text of PDF is then broken down into smaller chunks based on a chunk size measured in terms of the number of words. Tokenization converts these chunks into smaller units, such as subwords, making it suitable for processing by Large Language models. This project employs subword-based tokenization, which is highly effective for handling out-of-vocabulary words and maintaining semantic structure. Sentence Transformers generate dense vector representations of these tokens, ensuring semantic retention. Alternative approaches, such as word-based tokenization[14], are computationally lighter but fail to capture contextual relationships,

making them less effective for scientific text processing. The screenshot below (Figure 3.10) shows an example of text being broken down into tokens and converted into embeddings(Figure 3.11).

```
Original text:
This is an example sentence to illustrate tokenization and vectorization

Step 1: Tokenization:

Token IDs:
[101, 2023, 2003, 2019, 2742, 6251, 2000, 19141, 19204, 3989, 1998, 9207, 3989, 102]

Tokens:
['[CLS]', 'this', 'is', 'an', 'example', 'sentence', 'to', 'illustrate', 'token', '##ization', 'and', 'vector', '##ization', '[SEP]']

Step 2: Embeddings:

Embedding shape: (384,)
Sample embedding values (first 20 dimensions):
[ 0.02853777  0.08226844  0.03734801  0.0331188   0.04716726  0.00316423
  0.08714339  0.02005985  0.0131845   0.00073176  0.05827313 -0.00201833
  0.04761961  0.00513821  0.01119517  0.02200647 -0.02385613  0.08998881
 -0.08269478 -0.02824724]
```

Figure 3.10 Original, tokenized, and vectorized text

Sentence embeddings transform textual data into high-dimensional numerical vectors that encode semantic meaning, enabling similarity-based searches for efficient document retrieval. We used MiniLM architecture from the sentence transformers package[5] to generate embeddings with superior contextual understanding.

```
[59]: pdf_embedding

[59]: array([-1.22818775e-01, -3.09121367e-02,  2.99247559e-02,  1.39283817e-02,
              8.77714157e-02,  3.97068150e-02, -5.23743546e-03, -1.39394319e-02,
             -3.50674205e-02, -3.30992155e-02, -8.41207523e-03, -4.09878939e-02,
             -1.44968377e-02,  4.40649725e-02, -3.13341320e-02, -4.05359492e-02,
              5.88430874e-02,  8.79773051e-02, -1.60534650e-01, -1.48112904e-02,
             -6.98640617e-03, -6.38818019e-04,  2.74431780e-02, -5.58865396e-03,
             -4.77242144e-03, -2.89388858e-02, -1.25369085e-02, -7.63749853e-02,
              8.18719901e-03, -5.03617525e-02, -2.35850830e-03,  7.06209242e-03,
             -6.63494645e-03,  6.35635853e-02,  9.22131445e-03,  3.82343717e-02,
             -2.11167913e-02, -1.07168537e-02,  6.82470063e-03, -2.43803803e-02,
             -5.67892473e-03,  1.71989836e-02,  1.85554549e-02,  4.29035462e-02,
              8.95134509e-02,  4.93700206e-02, -1.17396303e-02, -7.23688230e-02,
              6.93843290e-02,  2.67110448e-02, -3.20169404e-02,  1.22534977e-02,
             -2.13434882e-02,  1.09122209e-01, -2.95895804e-02,  1.65140647e-02,
             -2.27539968e-02, -1.09549367e-03, -2.40635090e-02,  4.45314236e-02,
              6.25792518e-02, -7.94315711e-02, -2.01931372e-02, -3.30179632e-02,
             -2.65185647e-02,  2.69184019e-02,  6.54102070e-03, -4.49164324e-02,
             -4.85284813e-03, -1.86137427e-02, -9.66690294e-03,  8.57126862e-02,
             -9.31679904e-02,  2.79648658e-02,  7.00672269e-02,  3.59925255e-02,
              5.16618714e-02,  1.18886717e-02,  2.82961205e-02, -1.27114311e-01,
              8.34373757e-02,  8.91995952e-02,  9.94896442e-02, -3.18146572e-02,
              1.17015757e-01,  3.47067378e-02, -1.02243498e-01,  1.40814362e-02,
             -4.96036969e-02, -2.06258800e-02, -1.65918265e-02, -4.33785282e-02,
```

Figure 3.11 Embeddings for a whole chunk

Chapter 4: Methodology

4.1 System Architecture and Workflow

Our project follows a structured RAG pipeline.

Below is a workflow diagram illustrating the end-to-end process of the system :

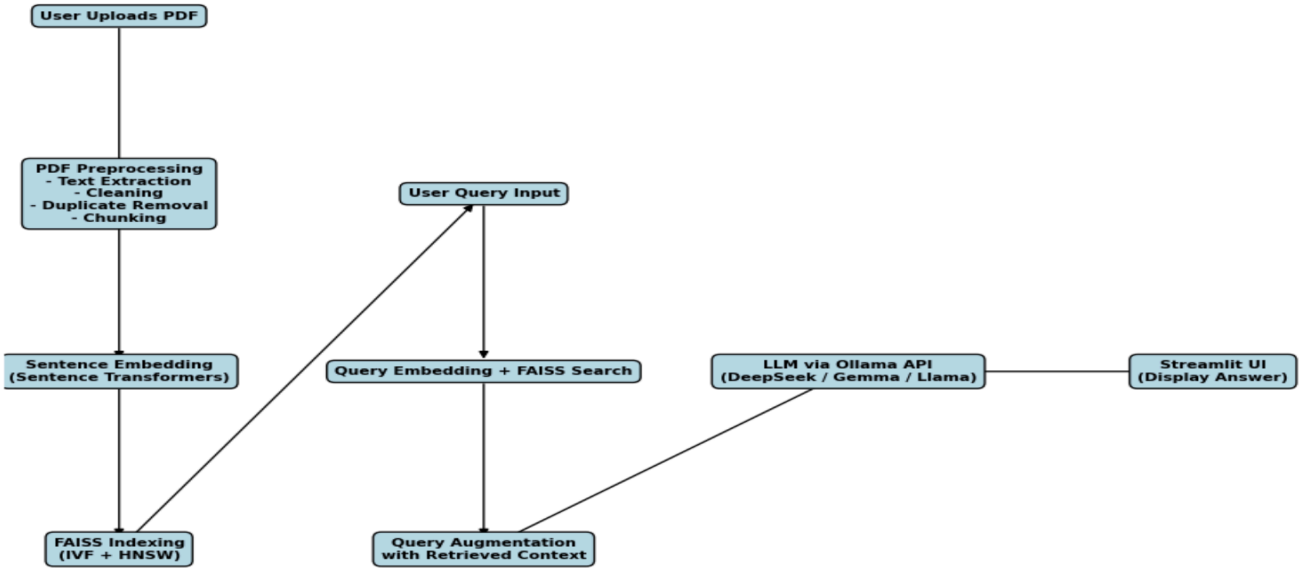


Figure 4.1 Workflow diagram

Our system workflow begins when a user uploads a research paper in PDF format. The document undergoes text extraction and preprocessing, is cleaned, tokenized, and converted into embeddings. These embeddings are then indexed in FAISS, enabling fast and efficient similarity-based retrieval. When a user submits a query, the system searches the FAISS index to find the most relevant document chunks. The retrieved text is then augmented into the query, ensuring context-rich responses before passing it through a large language model (LLM) via the Ollama API to generate a response. The final result is then displayed to the user through the Streamlit interface. This workflow ensures high retrieval accuracy, efficiency, and improved user experience in research analysis.

4.2 Techniques

4.2.1 Embedding & Vector Index(FAISS)

Once the raw text has been extracted from the uploaded PDF and cleaned to remove duplicates through hashing, chunked into smaller sections, tokenized into single tokens and converted into sentence embeddings using a sentence transformer-based model, the final step in the preprocessing pipeline is to

index these embeddings. This helps for fast and meaningful retrieval when a user submits a query. In traditional information retrieval systems, indexing often involves mapping keywords to document locations. However, our system is based on semantic search, not keyword search. Each chunk of text has been transformed into a dense vector that captures its meaning, so we need an indexing system that can operate in vector space, not keyword space. This is where FAISS(Facebook AI Similarity Search) becomes essential.

FAISS is a library specifically designed for efficient similarity search and clustering of dense vectors. FAISS enables approximate nearest neighbor (ANN) search in high-dimensional spaces, allowing the system to retrieve document chunks based on semantic similarity rather than exact keyword matches. It supports multiple indexing strategies and offers GPU acceleration, making it suitable for real-time and large-scale document retrieval tasks[4].

Our application used the IndexFlatL2 structure from FAISS, which organizes the vector space using Euclidean (L2) distance as the similarity metric. Each embedding is added to the FAISS index, and a mapping file is maintained to track metadata such as the original document name, section position, and source location. FAISS arranges all embedded segments so that semantically similar vectors are positioned close together. This organization plays a crucial role in ensuring that, at query time, the system can quickly identify and return the most relevant segments.

We evaluated several vector similarity search libraries during the system design phase. While Annoy is known for memory efficiency and ScaNN for high retrieval accuracy, they present limitations—Annoy struggles with large-scale scalability, and ScaNN lacks GPU acceleration, reducing its suitability for performance-intensive environments. FAISS emerged as the most suitable option due to its balance of speed, scalability, and GPU support. Although recent work[24] such as ScaDANN has introduced promising disk-based graph indexing methods for ANN under memory-constrained conditions, FAISS remains more mature and widely supported for our current application needs.

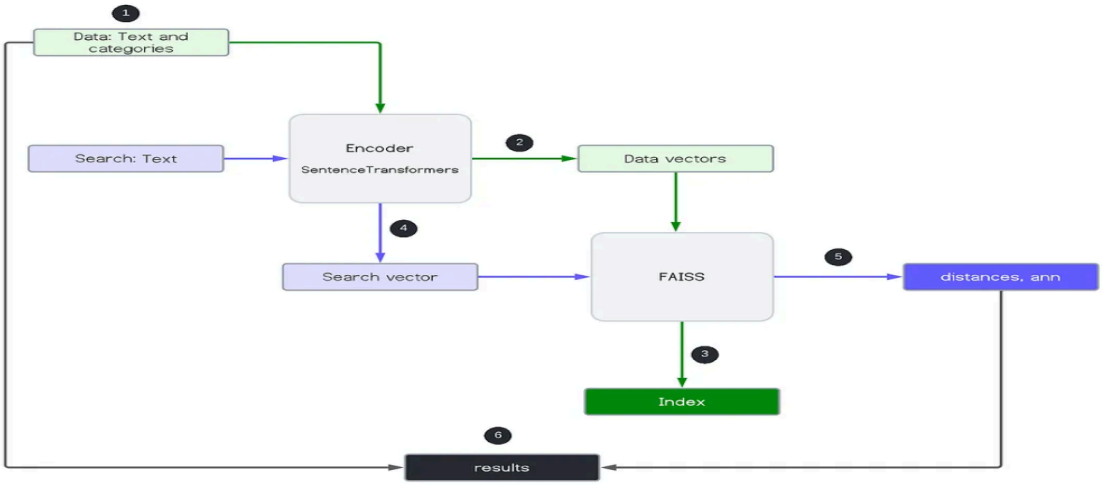


Figure 4.2 FAISS Workflow diagram

This indexing step completes the document ingestion pipeline. The system can handle user queries with all documents now embedded and indexed in FAISS. When a user submits a prompt, the system will use this FAISS index to identify relevant document segments, which are then passed into a Retrieval-Augmented Generation (RAG) pipeline to generate a meaningful response. The RAG component, described in Section 4.2.2, leverages this index to enhance language model outputs with real, grounded knowledge from the uploaded documents.

4.2.2 Query Time Retrieval & Generation(RAG)

Once the uploaded documents have been preprocessed and indexed using FAISS, the system is ready to process user queries. This is where Retrieval-Augmented Generation (RAG) comes into play. RAG enhances generation by dynamically retrieving relevant content from an external knowledge base—in our case, the indexed document embeddings from Section 4.2.1. RAG is highly effective for academic and research applications that require factual grounding. Enhancing user queries with real-time document content ensures that responses are contextually relevant and traceable to sources, addressing the limitation of standalone language models that often generate unverifiable information.

The RAG process(Figure 4.3) begins when a user submits a question or prompt through the system's user interface. This input is passed through the same Sentence Transformer encoder used during document indexing, converting the query into a high-dimensional vector embedding that represents its semantic meaning.

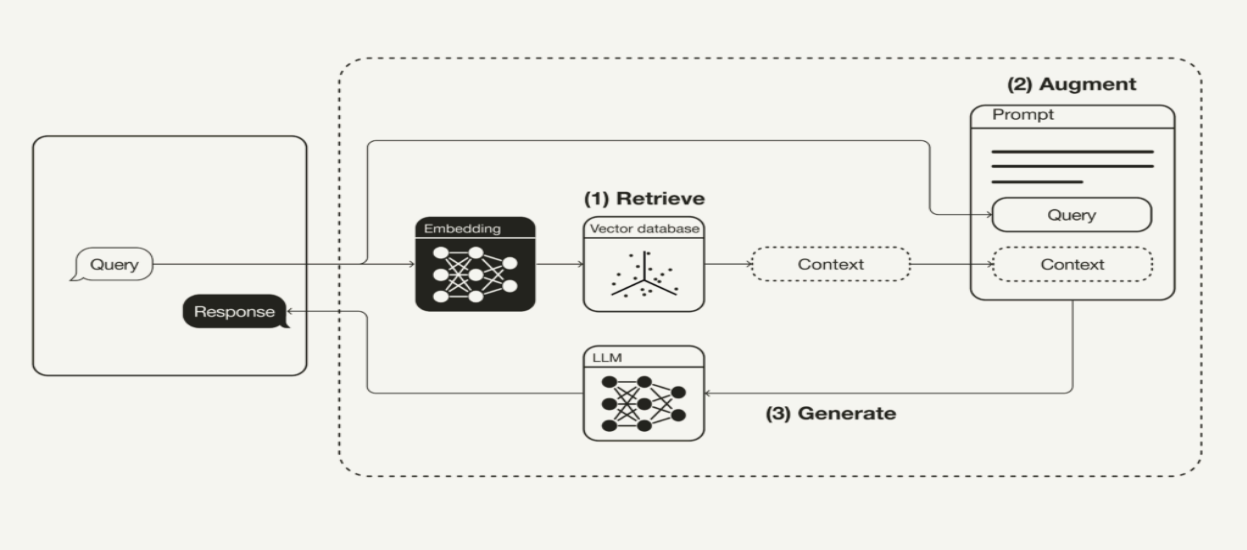


Figure 4.3 RAG Workflow – Retrieve, Augment, Generate

Next, the query embedding is used to search the FAISS index, which is compared against all stored document vectors. Using approximate nearest neighbor (ANN) search, FAISS returns the top-*k* most similar document segments—those most closely aligned in meaning with the user's query. Once retrieved,

these document chunks are collected and appended to the original query, forming what is known as an augmented prompt.

```python
# Suppose we have three short text entries to store in our knowledge base.
# You could replace these with actual text from your PDF extraction phase.
knowledge_base = [
    "Deep learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain.",
    "FAISS is a library for efficient similarity search and clustering of dense vectors.",
    "Sentence embeddings map entire sentences to a vector space in which semantically similar sentences are close."
]
```

Figure 4. 4 Sample Knowledge Based

```
[274]: user_query = "What do I need for similarity search in machine learning?"
       retrieved_docs, dists = retrieve_relevant_docs(user_query, knowledge_base, index, top_k=2)

       print(f"User Query: {user_query}\n")
       for i, (doc, dist) in enumerate(zip(retrieved_docs, dists)):
           print(f"Rank {i+1} (Distance: {dist:.4f})")
           print(f"Document: {doc}\n")

       User Query: What do I need for similarity search in machine learning?

       Rank 1 (Distance: 0.8856)
       Document: FAISS is a library for efficient similarity search and clustering of dense vectors.

       Rank 2 (Distance: 1.3834)
       Document: Sentence embeddings map entire sentences to a vector space in which semantically similar sentences are close.
```

Figure 4.5 Demonstration of Retrieval Mechanism

This prompt now contains the user's question and supporting evidence extracted from the uploaded PDFs. This augmented input is then forwarded to a Large Language Model (LLM) through the Ollama API, where a final answer is generated. Because the model now has immediate access to relevant source material, it can produce more accurate, contextually complete, and academically grounded results.

The generated answer and references to the supporting document segments are displayed through the system's Streamlit interface. This interface ensures that users not only receive a clear and concise answer but can also trace the response back to its evidence, enhancing transparency and trust.

By integrating real-time retrieval with generative capabilities, RAG creates a system that bridges the gap between search engines (which retrieve text) and language models (which generate text). Recent research[3] demonstrated that this hybrid approach significantly improves generated outputs' factual consistency and reliability, especially in knowledge-intensive tasks such as academic literature review and scientific QA systems.

4.2.3 Ollama API

The Ollama API plays a central role by enabling dynamic model selection based on query complexity and resource availability. This reduces latency and boosts real-time responsiveness, making it well-suited for our RAG-based research assistant [4]. Recent studies [23] show Ollama's effectiveness in local RAG setups, enhancing retrieval accuracy and data privacy by avoiding cloud-based dependencies. Its compatibility with FAISS allows efficient document chunk retrieval, while integration with Streamlit

enables real-time, citation-backed summaries. These features make Ollama a strong fit for secure, fast, and context-rich research analysis [4].

4.2.4 Model Selection

The selection of models is based on several key factors, including retrieval accuracy, efficiency, scalability, and response coherence. Among the models, we decided to work on six models—Llama 3.2, Llama 3.1, Mistral, Qwen 2.5, Gemma 3, and DeepSeek-R1, and each offers unique advantages and limitations based on their architectural capabilities and computational efficiency.

These Large Language Models are open-source, making them accessible, flexible, and reliable for experimentation. Large language models, also known as LLMs, are very large deep learning models that are pre-trained on vast amounts of data to understand and generate human-like language. It uses billions of parameters to identify patterns, predict words, and produce context-aware responses based on input text. The LLM parameters are essential elements that define the model's behaviour. They represent the values the model learns from training data and uses to make predictions. These parameters shape the model's understanding of language, directly influencing how it interprets input and generates responses. Each parameter acts like a piece of a massive puzzle, contributing to the model's ability to produce coherent, human-like text. Large Language Models typically contain millions or even billions of such parameters. Together, they form the foundation of the model's linguistic capabilities, enabling it to effectively comprehend, generate, and contextualize language**.** For instance, Llama 3.2 contains 3 billion parameters.

The table below summarizes the key differences between the selected models, highlighting their developers, parameters, model size, capabilities, limitations, and ideal use cases.

Table 4.1 Model Comparison Table

Model	Developer	Parameters (in Billions)	Model Size(in GB)	Advantage	Possible Limitations	Why It Can Be Ideal for Our Project
Llama 3.2	Meta AI	3	2.0 GB	Lightweight and highly efficient for fast retrieval	Due to less size it may have limited knowledge depth for complex research retrieval	Ensures low-latency performance for research paper analysis[16,17]

Llama 3.1	Meta AI	8	4.9 GB	High retrieval accuracy, strong multilingual capabilities	High computational demand, requiring optimized inference	Supports multilingual research retrieval and comprehensive document summarization[16,17]
Mistral	Mistral AI	7	4.1 GB	Low-latency response generation for efficient AI interactions	Limited reasoning depth	Can Enable fast query response in an interactive RAG-based system[16,17]
Qwen 2.5	Alibaba Cloud	7	4.7 GB	AI-driven automation, structured reasoning, and document processing	Requires fine-tuning for research document retrieval	Enhances structured document retrieval and research knowledge processing[16,17]
DeepSeek-R1	DeepSeek AI	8	4.7 GB	Optimized for retrieval-augmented reasoning and knowledge extraction	Complex fine-tuning and moderate GPU needed	Can ensure high-quality document retrieval and response generation[15,16,17]
Gemma 3	Google DeepMind	12	8.1 GB	Highly scalable, supports multimodal retrieval and reasoning	High computational cost, challenging real-time deployment	Considering it's large size and number of parameters, it can definitely handle a lot of complex content from literature[16,17]

4.3 Procedure

Below is a screenshot of our User Interface developed through Python library “streamlit”.

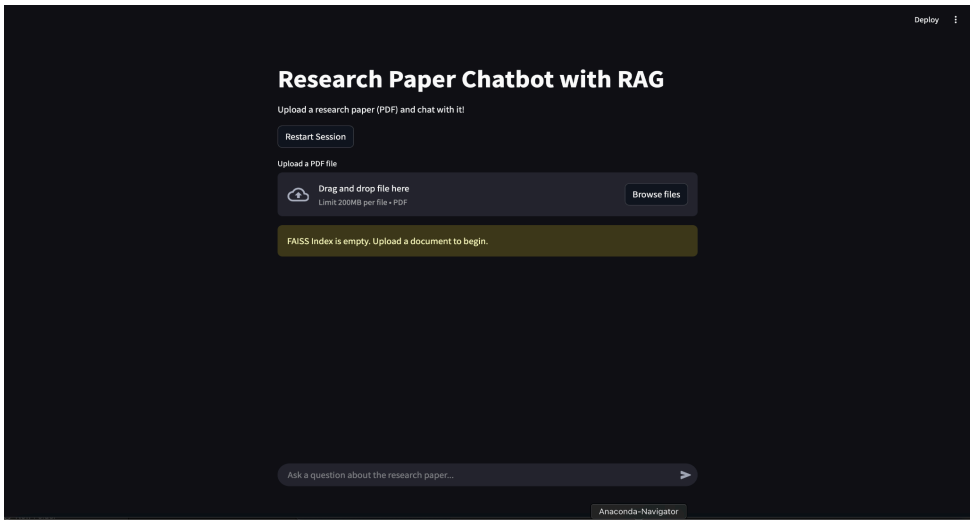


Figure 4.6 Screenshot of basic UI.

User Upload: The user uploads a PDF via the Streamlit interface.SHA256 hash of the file is computed to avoid re-processing duplicates. A Restart session button is added to reset the current session and start a new chat, which will reset the FAISS index and delete any previous stored chunks

Figure 4.7 Processing of PDF

Extracted Text Preview: The application also provides a window which can be expanded to preview the extracted text (the first 1000 characters) and immediately verify that the PDF’s content has been correctly captured without losing essential information. This is to ensure that it reads the right document.

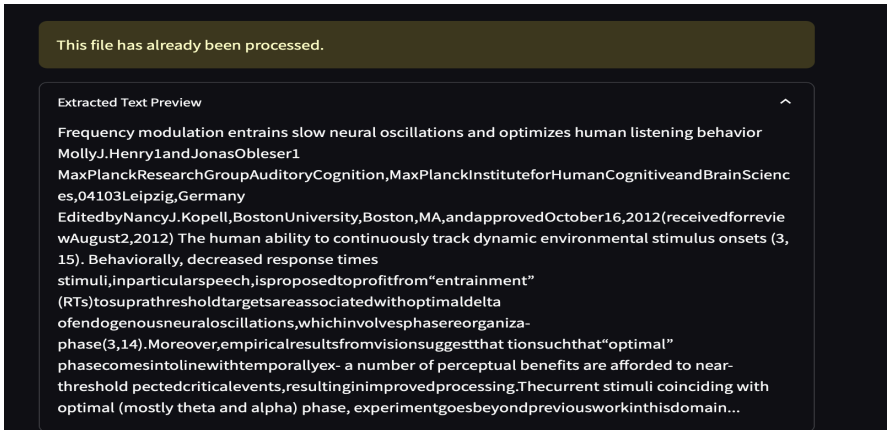


Figure 4.8 Expandable Preview of the text

Extraction & Storage: The system uses PyMuPDF to extract text from all pages of the uploaded PDF, concatenating the text into a single string and then structuring it for better comprehension. The extracted text is appended to a knowledge base stored in a JSON file (knowledge_base.json). Simultaneously, the extracted text is converted into a 384-dimensional embedding using embeddings = model.encode(text, convert_to_numpy=True).astype('float32'). These embeddings are added to the FAISS index for future retrieval.

Initial Summary: After embedding, the code forms a summary prompt by combining the original text with the retrieved context. This prompt is sent to the Ollama API to generate a summary, which is then displayed in the chat interface.

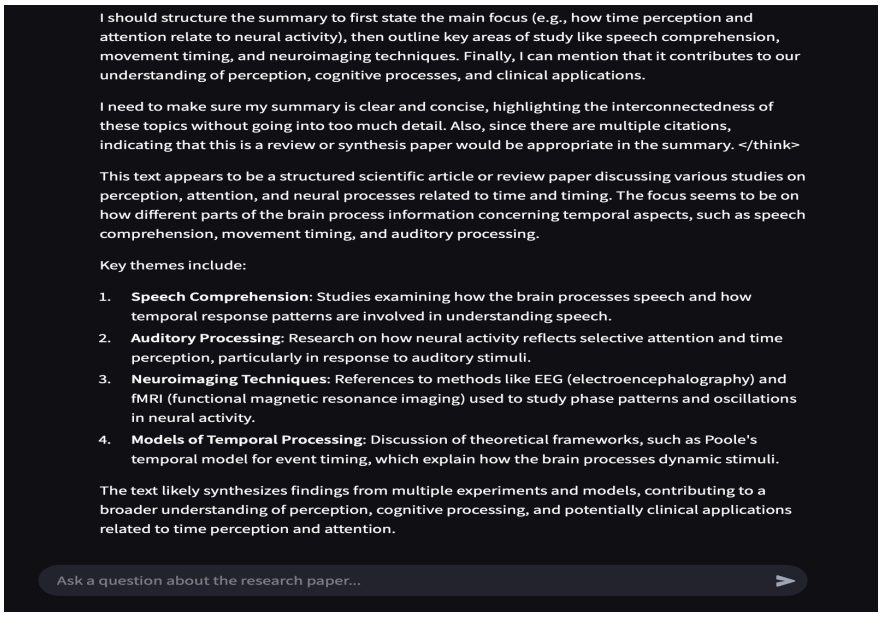


Figure 4.9 Preview of the summarized text

FAISS Index Monitoring: To confirm that each processed PDF contributes an embedding to the index and prevents duplicate uploads, the system displays the number of entries in the FAISS index using the " ntotal" attribute.

Figure 4.10: Indication for FAISS Index

User Query: When a user submits a question, the system encodes the query with the same model. The FAISS index is searched for the top 3 most relevant text chunks. The retrieved context is appended to the user's question to form a new prompt, which is then sent to the local LLM. The final response is displayed in the Streamlit chat window.

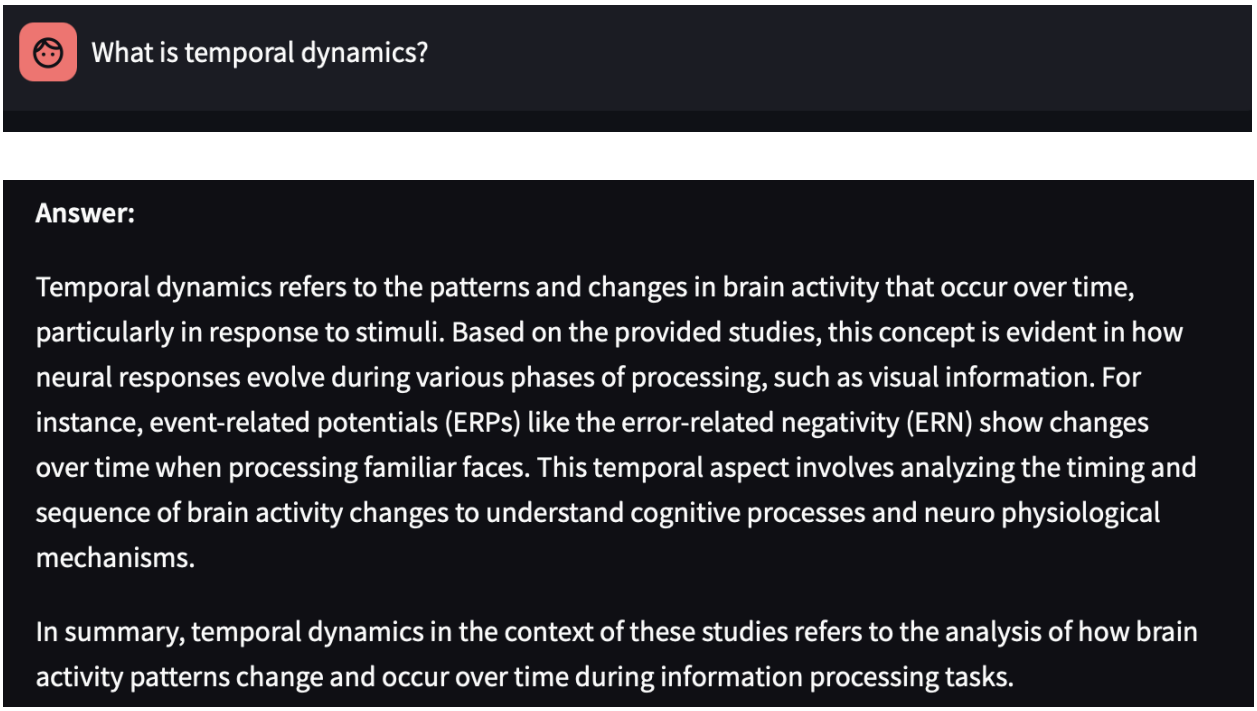


Figure 4.11:An Example query with relevant response

Evaluation: We will also test the query response system using manually created question-answer pairs to verify that the retrieval and answer generation produce correct and contextually relevant responses. To ensure accuracy, the system is tested using ROUGE, BERT Score, Latency, and size.

Chapter 5: Results and Analysis

5.1 Prompt engineering

For summarization, a prompt needs to be configured in the backend, which will be fed to the Large Language model to get a concise summary. This step is crucial as it ensures the generated summaries are accurate, relevant, and tailored to our reference summaries. A literature paper contains complex, domain-specific language, so a simple prompt like "summarize this paper" won't be able to comprehend and capture all the required details from the paper and may focus more on unnecessary and redundant sections. Without a tailored prompt, the LLM may generate unreliable or inconsistent summaries. The below plot showcases how efficient prompt engineering will improve the model response. We start from a basic prompt like "summarize this research paper" to an appropriately tailored prompt, and we observe a consistent improvement in the performance of all the models. The following table summarizes the prompts we experimented with.

We began the process by manually testing each model with our selected 15 research papers. During each summarization prompt, the summarization time by each model is noted, as this is an important metric for our analysis in later stages. We documented each summary generated by a model on different prompts for 15 different research papers in separate documents(.docx). A clear separator phrase was established to precede every summary. Later, it was converted into a structured .csv format, which includes the following sections: paper no, summary text, word count, and processing time. Similarly, the reference summaries were organized and structured in a consistent .csv format and aligned in order with the model-generated files. Once we gathered all the required data, we quantitatively evaluated the data by calculating BERTScore F1 scores between the reference and model-generated summaries.

Table 5.1 List of Prompts Applied for Model Evaluation

Prompt Number	Prompt Text
prompt1	Summarize the following research paper:\n\n{text}\n
prompt2	Provide a brief summary of this research paper, covering its problem statement, methodology used, the main results and conclusions\n\nPaper:\n{text}\n
prompt3	Analyze the following research paper:\n\n{text}\n\nPlease deliver:\n1. A simple breakdown of the abstract\n2. A one-sentence description of the methodology\n3. The key findings and their significance\n4. Any suggested future work\n
prompt4	Analyze the following research paper with its structured sections:\n\n{text}\n\nPlease provide:\n1. A detailed breakdown of the abstract in simple terms\n2. Provide the key references used in the paper along with their links and how they relate to the problem statement or solution\n3. Any existing solutions—if yes, what makes this project stand out and how it improves upon existing literature\n4. A concise summary of methodology or techniques used in brief\n5. Main results and conclusions and proposed future work\n

Table 5.1 starts with the basic prompt1 and moves to a well-tailored prompt4. Figure 5.1 showcases the BERTScore obtained for all the models under the usage of different prompts, where it is evident that prompt4 outperformed for almost all models. Then, we proceeded to structure and visualize these performance statistics.

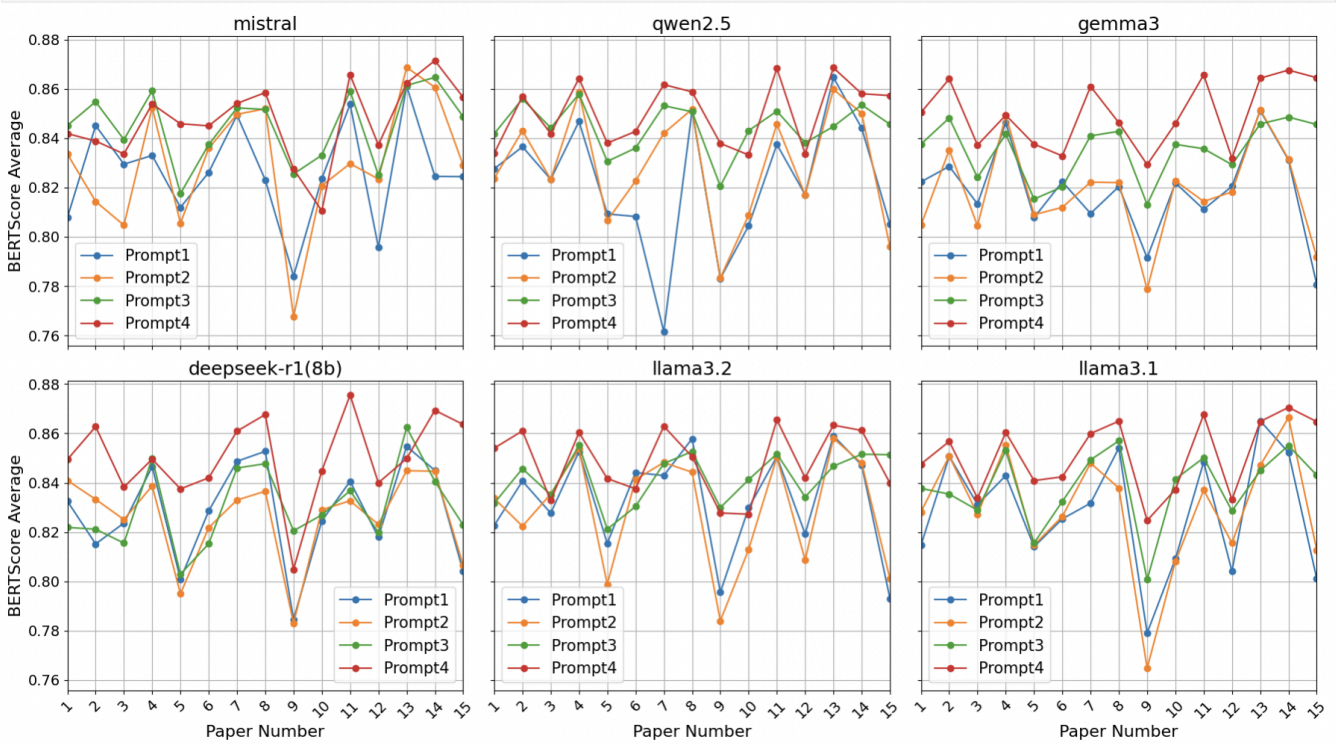


Figure 5.1 BERTScore Comparison of Four Prompts Across Models

5.2 Model Evaluation

We employed an analytical and data-driven approach to evaluate the models' performance that unfolded over multiple stages. After getting the best prompt through the stage of prompt engineering we used the responses generated from that prompt and again computed the other performance metrics across different models.

Table 5.2 Quantitative Evaluation of ROUGE and BERTScore scores over reference and model-generated summaries.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore	Summarization Time (sec)	Word Count

Deepseek-r1(8b)	0.4033	0.1516	0.2007	0.850337	**46.1213**	352.533
Gemma3	0.4127	0.1515	0.20462	0.849782	**47.632**	366.4
Lamma_3.1	0.38824	0.14501	0.1974	0.846775	81.265	330.066
Lamma_3.2	0.4053	0.15148	0.2011	0.850242	**48.5833**	371.2666
Mistral_Summary	0.40865	0.15363	0.2034	0.851195	110.3953	370.0666
Qwen_2.5	0.4088	0.1489	0.1980	0.848448	104.5426	362.533

Table 5.2 gives us an overview of average performance metrics across 15 papers for each model. The models, on average, performed very similarly on all metrics except for the summarization time, where we see some bifurcation. We also wanted to take a closer look at the performance scores using visualizations. This will help us analyze the data more thoroughly and understand patterns or trends.

A total of six line plots(Figure 5.2) were visualized using distinct colors to distinguish each model, allowing for a comprehensive comparison of their performance metrics across all the papers. From visual inspection, we observed a clear separation in the graph representing summarization time. For the other metrics, however, the performance was largely similar. We inferred that Deepseek-R1, Gemma3, and Llama 3.2 were superior to the different models, as they recorded significantly lower processing times while performing similarly on the other metrics. We conducted additional statistical evaluations to explore this further and validate our findings.

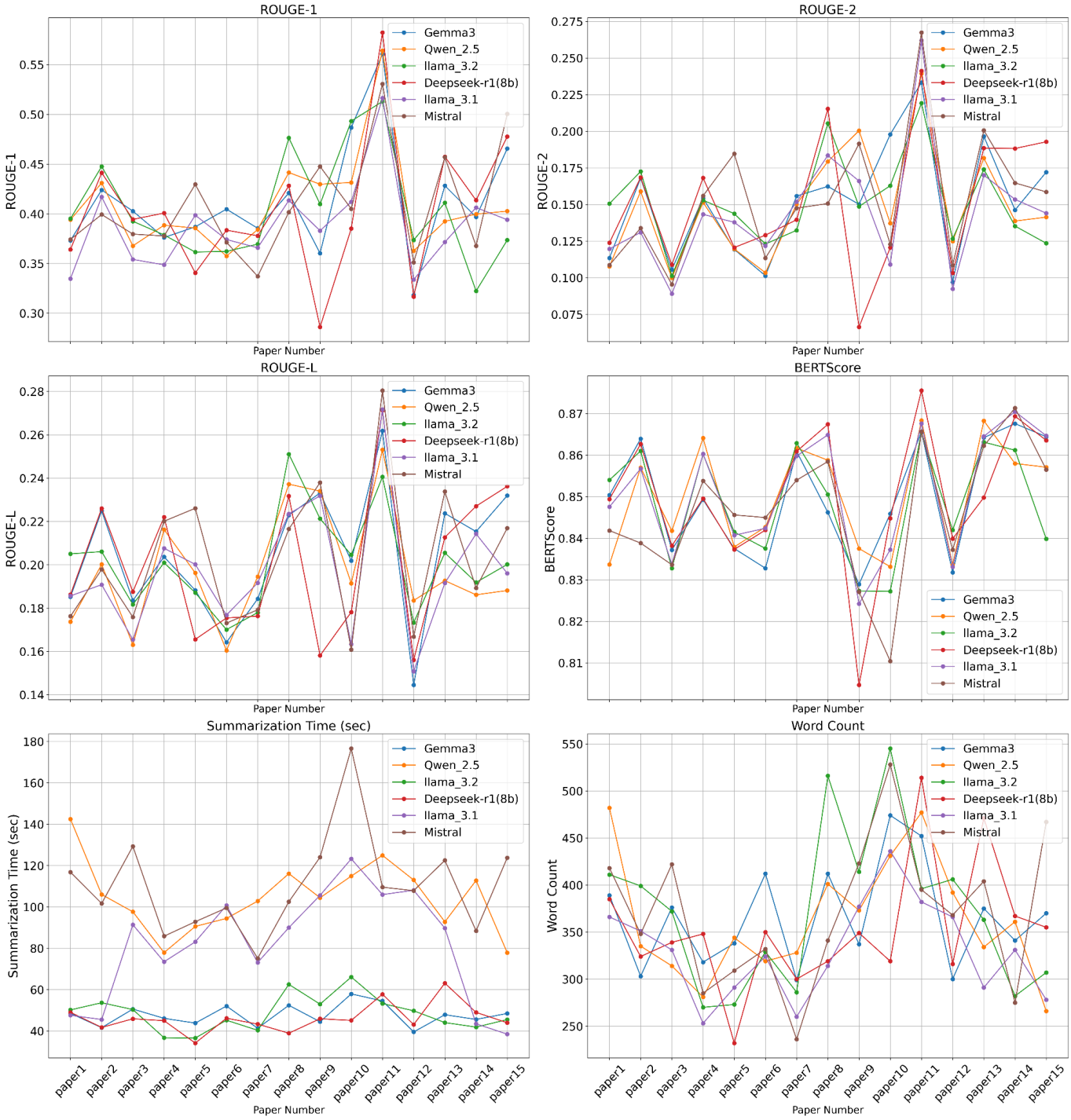


Figure 5.2 Comparative Evaluation of Models Across Multiple Metrics

Our primary goal for this analysis is to assess the trade-off between the summarization quality and computational efficiency. This interpretation can be understood better through the scatter plot (Figure 5.3) below, which plots the average BERTScore as a function of processing time.

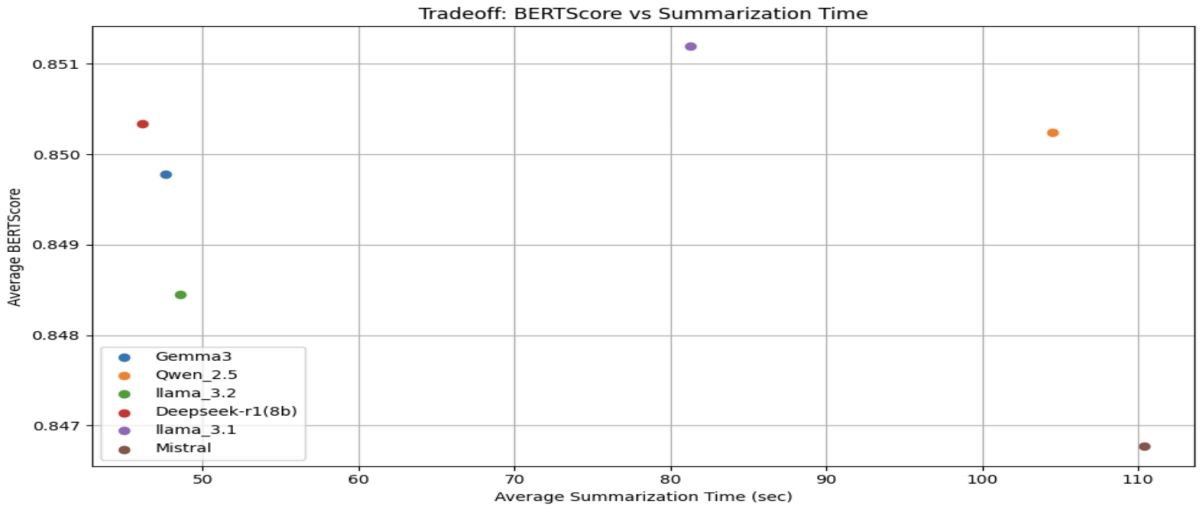


Figure 5.3 Average BERTScore vs Average Summarization Time(sec)

Since reasoning alone is insufficient to decide which model best suits our application, we need to do some statistical comparisons to select which model best suits our project. We conducted an ANOVA (Analysis of Variance) test(Figure 5.4) to determine whether the observed differences in each metric across the models were statistically significant. ANOVA is a statistical test used to compare the means of three or more groups to identify significant differences[20] among them. It divides the total variation in the data into two components: the variation between the groups and the variation within the groups. If the variation between the models is significantly larger than the variation within each model's results, it suggests that there is likely a significant difference between the models[20]. The results revealed that while there were no statistically significant differences in the variants of ROUGE and BERTScore scores, we obtained a p-value of less than 0.05 for processing time, indicating a significant difference that solidifies our previous finding.

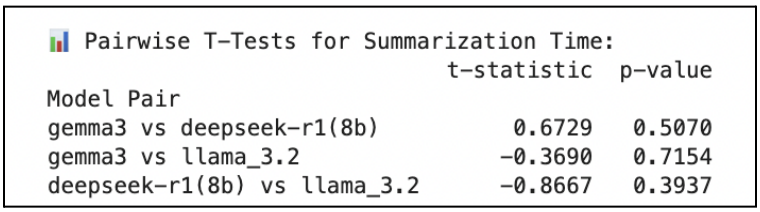
📊 ANOVA Results:

Metric	ANOVA p-value
ROUGE-1	0.8796
ROUGE-2	0.9950
ROUGE-L	0.9836
BERTScore	0.9732
Summarization Time (sec)	0.0000
Word Count	0.5492

Figure 5.4 ANOVA Test Result over Metrics

To explore the differences in summarization time among the top-performing models—Gemma 3, DeepSeek-R1, and llama 3.2, we conducted pairwise t-tests. The t-statistic[22] helped us assess the magnitude of differences between the models, with positive values indicating that the first model performed better and negative values showing the opposite. The results(Figure 5.5) revealed that DeepSeek-R1 had the shortest processing time compared to the others. However, the differences were not

statistically significant, suggesting that all three models are feasible options for the final task based on the metrics we analyzed.



📊 Pairwise T-Tests for Summarization Time:

Model Pair	t-statistic	p-value
gemma3 vs deepseek-r1(8b)	0.6729	0.5070
gemma3 vs llama_3.2	-0.3690	0.7154
deepseek-r1(8b) vs llama_3.2	-0.8667	0.3937

Figure 5.5 T-Test Results

Based on the results, we realized that model size would play a more influential role than we had previously anticipated. The size difference among the three models is quite evident. Gemma-3 is noticeably the largest, with a size of over 12 GB, while deepseek-r1 ranks in the middle at 4.7 GB. For our use case, **llama 3.2** emerged as the most preferable model due to its smaller size of 2 GB.

5.3 RAG Response Evaluation

After selecting llama 3.2 as the best model for summarization, we also tested the model's response ability in a Retrieval-Augmented Generation (RAG) setup. The evaluation of the model responses to different prompts as a follow-up to the summarization would be quite different from evaluating the summaries. Because now we are essentially verifying if the system can extract the correct information from the knowledge base for the prompt given to the model. The responses need to be factually accurate, so the metrics used for summary evaluation, like ROUGE or BERTScore, may not be that effective here, as they could lead to a misleading interpretation.

We figured out that human evaluation is required and more reliable than metrics alone. For that, we created a questionnaire of over 225 questions, forming 15 questions from each paper, and drafted reference answers for each question by our own. The nature of the questions we came up with was more factual rather than interpretative (e.g., *"What dataset was used in this Experiment ?"*). We already got a good understanding of the model's ability to interpret text in the summary evaluation. So, our primary focus was on the model's ability to retrieve accurate responses to the questions asked. As llama 3.2 is our best-performing model, we concentrated on only analyzing that model. Analysing other models would be time-consuming and would not yield significant insights. Also, the system's response ability depends on how well the RAG pipeline is designed rather than the model itself.

Figure 5.6 gives us a basic idea of the model response compared to a basic human summary.While the answers we drafted are mostly fact-based and to the point in one line, the model gives a very elaborate answer and reflects on the rationale behind the answers in many cases.

Question	Manually written Answer	llama 3.2_responses
What neurodevelopmental conditions does the paper focus on?	ADHD and ASD in young adulthood.	The paper focuses on Attention Deficit/Hyperactivity Disorder (ADHD) and Autism Spectrum Disorder (ASD), two neurodevelopmental conditions that frequently co-occur.
What is the main brain imaging method reviewed in the study?	Electroencephalography (EEG) especially ERPs and oscillatory analyses,positron emission tomography (PET) and functional magnetic resonance imaging (fMRI)	The main brain imaging methods reviewed in the studies include functional magnetic resonance imaging (fMRI), electroencephalography (EEG), and positron emission tomography (PET).
What age group does the paper specifically review?	Young adults aged approximately 16–26.	The paper specifically reviews the ages 16-26, falling within the emerging adulthood period.
Are ADHD and ASD commonly co-occurring?	Yes, with ADHD in 30–80% of ASD cases and ASD in 20–50% of ADHD cases.	Yes, ADHD and ASD frequently co-occur, with studies suggesting that individuals with ADHD are more likely to also have ASD, and vice versa, with estimates ranging from 20% to 80% of individuals with one disorder also having the other.

Figure 5.6 Comparison of Model-Generated and Human-Written Responses

In some instances, it is even able to generate a better answer than ours, like the example shown in Figure 5.7 below, where instead of just answering the question, it also elaborates on the source from which it was able to retrieve its information, despite not explicitly mentioning it in the prompt.

What are the four main neurocognitive domains reviewed?	Attention processing, performance monitoring, face processing, and sensory processing.	The text does not explicitly state that there are only four main neurocognitive domains, but it mentions that the studies were grouped into "broad categories reflecting key neurocognitive domains" (Tables 2A-2G), which implies a comprehensive review of various domains. However, based on the context provided in Table 3, the main neurocognitive domains discussed appear to include: 1. Attentional domain 2. Performance monitoring 3. Face processing 4. Memory and language

Figure 5.7 Comparison of Well-Generated Model Responses vs Human Response for Accuracy

Among the 225 Questions asked, the model was able to generate an efficient response for over 195 questions, which can easily qualify as an acceptable response to the asked question. This implies an accurate response rate of over 87.5%, which showcases the selected model's accuracy in giving accurate responses.

Chapter 6: Discussion

This project demonstrates the effectiveness of combining Retrieval-Augmented Generation (RAG), FAISS indexing, and prompt engineering to build a secure, efficient, and research-focused AI assistant. Through comprehensive benchmarking across multiple large language models (LLMs), we observed that model performance remained consistent across ROUGE and BERTScore metrics but varied significantly in processing time.

Prompt engineering played a crucial role in shaping the quality of generated summaries. By designing tailored prompts, we ensured that models generated outputs closer in structure and content to human-written references. Our analysis confirmed that optimized prompts consistently improved performance across all tested models, making prompt design an essential step in real-world LLM deployment. While DeepSeek-R1, Gemma3, and Llama 3.2 demonstrated competitive accuracy, Llama 3.2 emerged as the most practical option due to its smaller model size. Therefore, only Llama 3.2 was selected for RAG response testing and achieved an accurate response rate of over 87.5% in retrieval-augmented question answering, reinforcing its effectiveness for research-focused tasks.

This system effectively met both our technical and personal learning objectives. Technically, we created a fully functioning end-to-end application that enables secure, offline document ingestion, summarization, and question-answering—an achievement that aligns with our initial goal of improving literature review workflows. We gained practical experience working with modern NLP techniques, LLM architectures, vector databases, and UI development, which enhanced our understanding of how research and development intersect in real-world applications. However, we also encountered certain challenges. Running Ollama models locally introduced memory constraints, especially with larger models like Gemma 3. Despite this, our system could handle these constraints by focusing on smaller yet optimized models like Llama 3.2, proving that effective summarization and retrieval can be achieved without relying on large-scale cloud infrastructure.

Future Improvements will focus on enhancing the system's flexibility, performance, and usability. Fine-tuning large language models with domain-specific datasets will be explored to improve the contextual accuracy of generated responses. The user interface will be enhanced to support features like multi-query history and citation-aware highlighting, making interactions more seamless and insightful. Additionally, advanced prompt engineering strategies will be investigated to ensure the system adapts well across a broader range of research domains.

Furthermore, the application can be containerized using Docker by creating a standardized Docker image that bundles all dependencies and configurations. This image can be shared through a container registry like Docker Hub, allowing others to easily pull and run the application across different environments and ensuring portable deployment. In summary, this project benchmarked the effectiveness of various models under different prompt structures and highlighted the importance of combining smart prompt engineering with lightweight yet capable large language models (LLMs).

Chapter 7: Contributions

This project involved collaborative work, and everyone played a key role, leading to its success. We, Roshini Talluru, Tushar Jayendra Mhatre, and Rahul Kataram, worked together under the guidance of Dr. Theodre Thrafails. As we worked on the project together, each of us gave individual contributions. The following section discusses the individual and key responsibilities of each individual.

Roshini Talluru took the lead in data collection and preprocessing steps. She contributed to collecting the research papers, as this is a crucial step, and the data needs to be well structured as we proceed to the next main steps. She also handled text extraction, chunking, and vector indexing processes. Apart from the technical contribution, her focus is on project documentation. She played a significant role in editing and organizing the report.

Tushar Jayendra Mhatre took the lead in methodology, implemented base LLMs, and performed statistical evaluation. This step is significant as it decides which model best suits our project and which metric improves the system's overall performance. Apart from that, he even implemented the FAISS vector database for efficient document retrieval. He and Roshini Talluru contributed to the document edition.

Rahul Kataram took the lead in UI development. This is the last and significant step, which uses a Streamlit user interface where users can upload their PDFs, ask questions, and receive answers related to the document. He also took part in data summarization, where he was primarily responsible for collecting and organizing the dataset of our research papers for system testing.

Our project supervisor, Dr. Theodore Trafails, supported and guided our project. His timely guidance helped us refine the project, and he took the lead in guiding and evaluating the manual summary we generated. Furthermore, our practicum coordinator, Dr. Matthew Beattie and Triet Tran, our TA's feedback on our report, which also helped us refine our project further and go beyond our expectations.

Chapter 8: References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021, April 12). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. ArXiv.org. https://doi.org/10.48550/arXiv.2005.11401

[2] Liu, J., Ding, R., Zhang, L., Xie, P., & Huang, F. (2024). CoFE-RAG: A Comprehensive Full-chain Evaluation Framework for Retrieval-Augmented Generation with Enhanced Data Diversity. ArXiv.org. https://doi.org/10.48550/arXiv.2410.12248

[3] Khan, A. A., Hasan, M. T., Kemell, K. K., Rasku, J., & Abrahamsson, P. (2024). Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report. ArXiv.org. https://doi.org/10.48550/arXiv.2410.15944

[4] Rahman, M. S., Syed, R., & Rashid, M. M. (2024). Optimizing Domain-Specific Image Retrieval: A Benchmark of FAISS and Annoy with Fine-Tuned Features. ArXiv.org. https://doi.org/10.48550/arXiv.2412.01555

[5] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. ArXiv.org. https://doi.org/10.48550/arXiv.1908.10084

[6] Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs. ArXiv:1702.08734 [Cs]. https://doi.org/10.48550/arXiv.1702.08734

[7] Izacard, G., & Grave, E. (2021, February 3). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. ArXiv.org. https://doi.org/10.48550/arXiv.2007.01282

[8] Lin, C.-Y. (2004, July 1). ROUGE: A Package for Automatic Evaluation of Summaries. Aclanthology.org. https://aclanthology.org/W04-1013/

[9] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805. https://arxiv.org/abs/1810.04805

[10] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., & Wang, H. (2023, December 18). Retrieval-Augmented Generation for Large Language Models: A Survey. ArXiv.org. https://doi.org/10.48550/arXiv.2312.10997

[11] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., & Jégou, H. (2024, January 16). The Faiss Library. ArXiv.org. https://doi.org/10.48550/arXiv.2401.08281

[12] Deshpande, R., & Bodkhe, S. (2017). Analysis of an Efficient Approach for Duplicate Detection System.

[13] Guo, S., Mao, X., Sun, M., & Wang, S. (2023). Double Sliding Window Chunking Algorithm for Data Deduplication in Ocean Observation. IEEE Access, 11, 70470–70481. http://dx.doi.org/10.1109/ACCESS.2023.3276785.

14. Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. ArXiv:1808.06226 [Cs]. https://doi.org/10.48550/arXiv.1808.06226
[15] Feng, X., Dou, L., & Kong, L. (2025). Reasoning Does Not Necessarily Improve Role-Playing Ability. ArXiv.org. https://doi.org/10.48550/arXiv.2502.16940
[16] Aydin, O., Karaarslan, E., Safa, E. F., & Bacanin, N. (2025). Generative AI in Academic Writing: A Comparison of DeepSeek, Qwen, ChatGPT, Gemini, Llama, Mistral, and Gemma. ArXiv.org.https://doi.org/10.48550/arXiv.2503.04765
[17] Ollama. (2025). Ollama, https://ollama.com/search
[18] Cachola, I., Lo, K., Cohan, A., & Weld, D. S. (2020). TLDR: Extreme Summarization of Scientific Documents. ArXiv:2004.15011 [Cs]. https://doi.org/10.48550/arXiv.2004.15011
[19] T Borkovits, S A Rappaport, T G Tan, R Gagliano, T Jacobs, X Huang, T Mitnyan, F-J Hambsch, T Kaye, P F L Maxted, A Pál, A R Schmitt, The compact triply eclipsing triple star TIC 209409435 discovered with TESS, Monthly Notices of the Royal Astronomical Society, Volume 496, Issue 4, August 2020, Pages 4624–4636. https://doi.org/10.1093/mnras/staa1817
[20] Kim, T. K. (2017). Understanding one-way ANOVA using conceptual figures. Korean Journal of Anesthesiology, 70(1), 22–26. NCBI. https://doi.org/10.4097/kjae.2017.70.1.22
[21] Yasunaga, M., Kasai, J., Zhang, R., Fabbri, A. R., Li, I., Friedman, D., & Radev, D. R. (2019). ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks. ArXiv (Cornell University). https://doi.org/10.1609/aaai.v33i01.33017386
[22] Rainio, O., Teuho, J. & Klén, R. Evaluation metrics and statistical tests for machine learning. Sci Rep 14, 6086 (2024). https://doi.org/10.1038/s41598-024-56706-x
[23] Liu, F., Kang, Z., & Han, X. (2024). Optimizing RAG Techniques for Automotive Industry PDF Chatbots: A Case Study with Locally Deployed Ollama Models. arXiv. https://arxiv.org/abs/2408.05933
[24] Gu, H., & Kim, M. (2025). ScaDANN: A Scalable Disk-Based Graph Indexing Method for ANN. Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference, Barcelona, Spain.