Scientific Research Paper Summarization System

Submitted to:

Prof. William Pourmajidi

Submitted By:

(Group 4)

Bibek Shiwakoti (C0898100)

Jumana Lightwala (C0892358)

Pratikkumar Mishra (C0891298)

Ronak Shah (C0891297)

Guneet Singh (C0882945)

August 12, 2024

Acknowledgement

We extend our heartfelt gratitude to Prof. William Pourmajidi for his invaluable guidance and supervision throughout the Capstone project, "Scientific Research Paper Summarization System." His insights and guidence were crucial to the successful completion of this project.

We also express our appreciation to the authors whose works were referenced and cited in our research. Their contributions provided a foundational basis for our study.

Recognition is due to our classmates for their valuable suggestions and discussions, which enriched our research and helped us navigate challenges along the way.

A special thanks to each team member for their collective enthusiasm, creativity, and dedication to this project. Together, we gained invaluable experience across various stages, including locating and obtaining dataset, selecting appropriate models, comparing accuracies of models and fine tuning them and finally integrating with the frontend. Our journey involved exploring diverse Natural Language Processing techniques and latest technologies to enhance the summarization functionality of the system.

Table of contents

List of tables

List of figures

Abstract

This project aims to enhance the accessibility of scientific research papers by leveraging advanced machine learning techniques, including Natural Language Processing (NLP), Large Language Models (LLMs), and the LangChain framework. Our system allows users to select research papers and receive concise, accurate summaries through a user-friendly interface. This streamlined approach facilitates quick comprehension of extensive scientific documents, thereby promoting more efficient research and study.

In our work, we evaluated several state-of-the-art LLMs, both with and without fine-tuning, using a custom dataset of research papers published on the arXiv website. We assessed four different models based on established ROUGE metrics and feedback from human evaluators. The BART-large-CNN model emerged as the top performer. Fine-tuning this model on the arXiv dataset resulted in a highly effective summarization system, consistently generating concise and accurate summaries. Integrated with Streamlit, this system offers a user-friendly interface for easy summary generation, significantly reducing the time and effort required for researchers to stay updated with the latest developments in their fields.

1 Introduction

The rapid growth of scientific literature has made it increasingly challenging for researchers and academics to stay current with the latest developments in their fields. With thousands of new papers published daily, manually sifting through and digesting this vast amount of information is both time-consuming and overwhelming. Recognizing this challenge, we developed a cutting-edge scientific research summarization system that leverages advanced machine learning techniques, including Natural Language Processing (NLP), Large Language Models (LLMs), and LangChain framework.

Our system allows users to upload or select research papers and receive concise, accurate summaries. This functionality helps users quickly grasp the essence of extensive scientific documents, saving valuable time and effort. To achieve this, we evaluated various existing LLMs from platforms like Hugging Face. After rigorous performance testing, we identified the BART-large-CNN model as the most effective for our purposes. We have fine-tuned this model on the arXiv dataset, which consists of a wide range of scientific papers, to optimize its performance for summarization tasks.

The resulting system is highly effective in generating concise and accurate summaries, significantly reducing the time and effort required for researchers to stay updated with the latest advancements. Our system not only simplifies the process of information retrieval but also enhances the efficiency and productivity of researchers by allowing them to focus more on their core work rather than on extensive literature review.

Moreover, our vision extends beyond the current capabilities of the system. We aim to integrate it with real-time research paper publication websites, enabling the automatic summarization of newly published papers tailored to specific interests. This integration would ensure that researchers receive timely updates on relevant research, further reducing the risk of information overload and enabling them to stay at the forefront of innovation and discovery.

In summary, this project addresses a critical need in the academic and research communities by providing a tool that streamlines the process of generating summaries from scientific literature. By leveraging advanced NLP and LLM techniques, our summarization system offers a practical solution to the problem of information overload, ultimately fostering a more efficient and productive knowledge gain from research papers in short span of time.

1.1 Statement of Need

1.1.1 Issue and Importance

In today's fast-paced academic environment, researchers and academics are inundated with an overwhelming volume of scientific literature. Sifting through numerous lengthy research papers to find relevant information is time-consuming and often inefficient. This challenge not only delays the research process but also hinders the timely dissemination and application of new findings. The development of a scientific research summarization system is crucial as it addresses this pressing issue by providing concise, accurate summaries of research papers. This solution will significantly enhance the efficiency of researchers, allowing them to quickly grasp key insights and focus on advancing their work.

1.1.2 Necessity

The necessity for this system stems from the critical need to streamline the research process in an era where the volume of published scientific literature is growing exponentially. Traditional methods of manually reading and summarizing papers are no longer feasible given the sheer amount of information available. By leveraging machine learning technologies such as NLP and LLMs, our system can automatically generate summaries, reducing the cognitive load on researchers. This not only saves time but also ensures that researchers remain current with the latest developments in their fields, fostering a more dynamic and responsive academic environment

1.1.3 Beneficiaries

The primary beneficiaries of this system will be researchers and academics who are at the forefront of scientific innovation. By providing them with quick access to concise summaries of relevant research papers, our system will enhance their ability to stay informed and integrate new knowledge into their work. Additionally, academic institutions and research organizations will benefit from increased productivity and more efficient knowledge dissemination. Ultimately, the broader scientific community and society at large will gain from the accelerated pace of research and discovery facilitated by this advanced summarization system.

2 Methods

This section provides an in-depth overview of the methods and approaches utilized in the development of our scientific research summarization system. It includes the context and setting of the study, the study design, details about the dataset used, identification of the main study variables, data collection instruments and procedures, and the analysis methods employed.

2.1 Study Design

The study was conducted within the domain of academic research, focusing on the need for efficient summarization of scientific papers. The primary objective was to develop a system that could generate concise and accurate summaries of extensive research documents, thereby aiding researchers and academics in staying updated with the latest developments in their fields.

2.1.1 Main Study Variables

The main study variables included:

- **Input Text**: The full text of the research papers from the arXiv dataset.
- **Output Summary**: The generated summaries of the research papers.
- **Performance Metrics**: Evaluation metrics such as ROUGE scores to measure the quality and accuracy of the summaries.

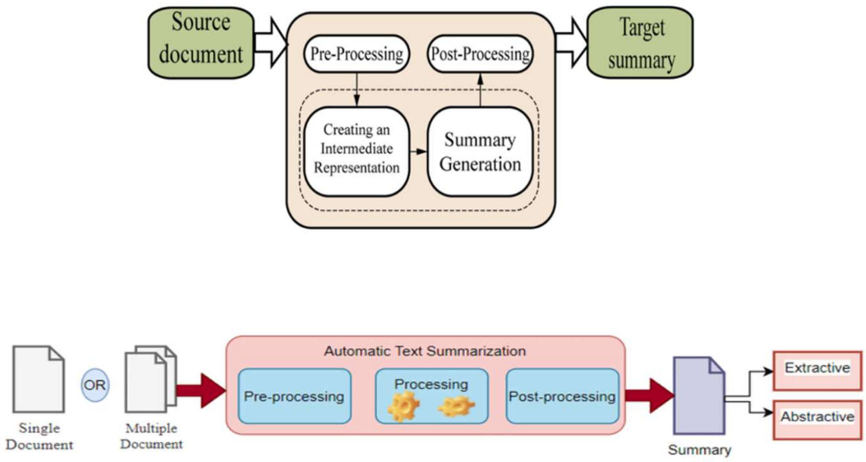


Figure 1: Basic architecture of summary generation process

2.2 Architecture

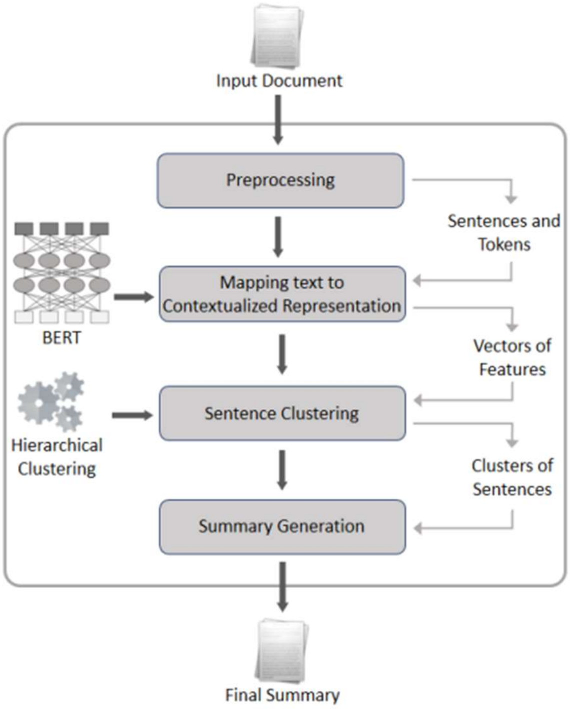


Figure 2: Detailed architecture

- Input Document:
  The process begins with the input document, which is a scientific research paper that the user wants to summarize. This document can be uploaded in various formats such as PDF, DOCX, or plain text. The input stage involves extracting the text content from these formats, ensuring that the full textual information, including titles, abstracts, body sections, and references, is accurately captured for subsequent processing.
- Preprocessing
  Preprocessing is a crucial step that prepares the raw text from the input document for analysis. This involves several tasks such as removing any non-textual elements (like figures and tables), cleaning the text by eliminating noise (such as special characters and irrelevant symbols), and normalizing the text by converting it to a consistent format (e.g., lowercasing, stemming, and lemmatization). Additionally, tokenization is performed to split the text into manageable units like sentences and words, which are essential for the next stages of processing.
- Mapping Text to Contextualized Representation

Once the text is preprocessed, the next step is mapping it to a contextualized representation using advanced natural language processing (NLP) models. This involves transforming the text into embeddings, which are numerical representations that capture the semantic meaning of the text. We will utilize transformer-based models such as BERT or GPT, which consider the context of each word in a sentence, providing a rich representation that encapsulates both syntactic and semantic information. These embeddings are essential for accurately understanding the content and context of the text.

- Sentence Clustering
  With the contextualized representations in hand, the next stage is sentence clustering. This involves grouping similar sentences together based on their semantic similarity. Clustering helps in identifying key themes and important points within the document by aggregating sentences that convey similar information. Various clustering algorithms, such as k-means or hierarchical clustering, can be employed to achieve this. The goal is to reduce redundancy and highlight the most significant sentences that will form the basis of the summary.
- Summary Generation
  The final stage is summary generation, where the most relevant and informative sentences from each cluster are selected to construct the summary. This involves selecting sentences that best represent the main ideas of the document while ensuring coherence and logical flow. The summarization algorithm aims to produce a concise and comprehensive summary that captures the essence of the original document, making it easier for users to quickly grasp the key findings and contributions of the research paper.

2.3 Tools and platform used:

We explored various summarization tools, technologies, and large language models (LLMs), discovering that models like OpenAI's GPT, BART, and many others are available on Hugging Face. Initially, we started with GPT due to its advanced capabilities. However, we encountered token limitations in the free version and realized that utilizing its full potential required a paid subscription. Consequently, we shifted our focus to the open-source models available on Hugging Face, which offer robust performance without the constraints of token limits, making them a more feasible option for our project.

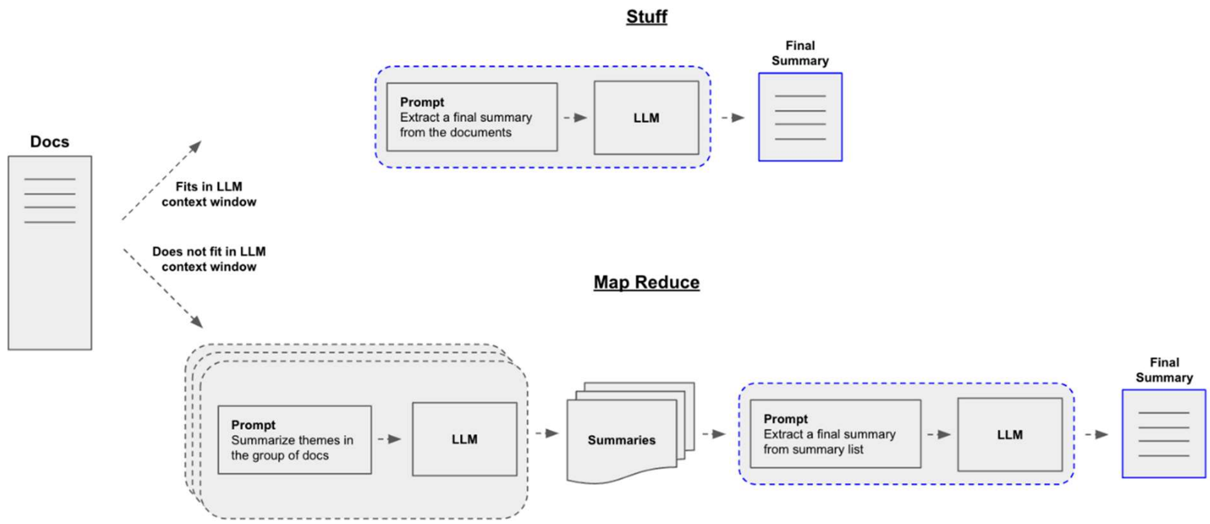


Figure 3: Summarization approach

2.3.1 Large Language Model (LLM):

A Large Language Model (LLM) is a type of artificial intelligence designed to understand and generate human language. These models are trained on vast amounts of text data, enabling them to recognize patterns, understand context, and generate coherent and contextually relevant responses. LLMs, such as GPT-3 by OpenAI, BERT by Google, and BART by Facebook, utilize deep learning techniques and transformer architectures to achieve high performance in various natural language processing tasks, including translation, summarization, question answering, and content generation. Their ability to process and generate text that closely mimics human writing has made them invaluable tools in a wide range of applications, from chatbots to automated content creation.

2.3.2 Huggingface:

Hugging Face is a leading company in the field of natural language processing (NLP) and artificial intelligence, known for its transformative impact on the development and accessibility of machine learning models. Initially gaining popularity with its open-source library for NLP, Hugging Face has grown into a comprehensive platform that offers a vast repository of pre-trained models, datasets, and tools. These resources facilitate tasks such as text classification, translation, summarization, and conversational AI. The platform's Model Hub provides easy access to state-of-the-art models like BERT, GPT, and BART, allowing researchers, developers, and businesses to leverage advanced AI capabilities without extensive computational resources. Hugging Face's commitment to fostering a collaborative community and democratizing AI technology has made it a cornerstone in the modern AI ecosystem.

2.3.3 Langchain:

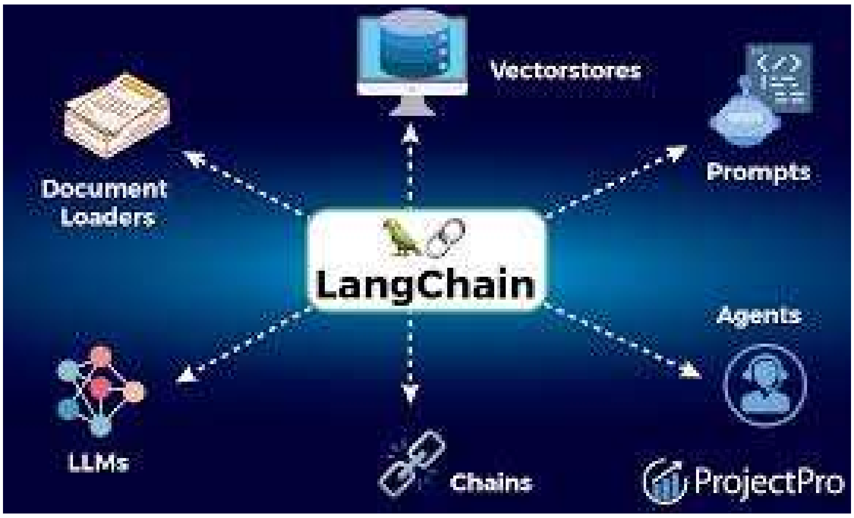


Figure 4: LangChain framework

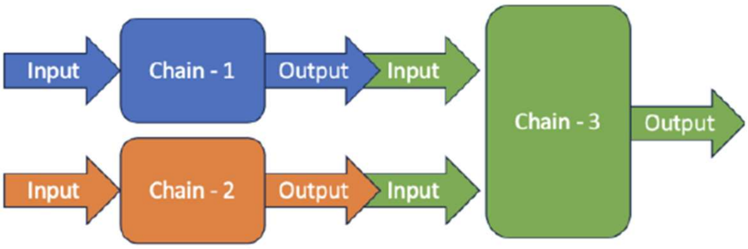


Figure 5: Example of a Sequential Chain that gets two inputs and outputs one result.

This implementation has significantly streamline our workflow and enhances the efficiency of processing and understanding research papers. The output of these two models now serves as a baseline reference for our work.

LangChain is a framework designed to simplify the development of applications that leverage large language models (LLMs). It provides a structured approach to integrate LLMs into various workflows and applications, making it easier to build, deploy, and manage language model-driven solutions. LangChain supports seamless integration with multiple LLMs, allowing developers to focus on creating innovative applications without worrying about the underlying complexities of language model management.

Advantages of LangChain:

- Ease of Integration: Simplifies the process of integrating various LLMs into applications, reducing the need for extensive technical expertise.
- Flexibility: Supports multiple LLMs, giving developers the freedom to choose and switch between different models as per their requirements.
- Modular Design: Offers a modular framework that allows developers to build applications in a structured manner, enhancing code maintainability and scalability.
- Enhanced Productivity: Streamlines development workflows, enabling faster prototyping and deployment of LLM-driven applications.
- Community and Support: Backed by a growing community and comprehensive documentation, providing ample resources and support for developers.
- Customization: Allows for easy customization and fine-tuning of models, helping developers create tailored solutions that meet specific needs.
- Integration with Existing Tools: Compatible with other popular tools and platforms in the AI ecosystem, facilitating seamless integration into existing workflows.

2.4 Data collection

we utilized the arXiv paper dataset available on Hugging Face. This dataset comprises a vast collection of academic papers from arXiv, covering a wide range of scientific domains. It includes comprehensive metadata, abstracts, and full-text summaries, making it an ideal resource for training and evaluating summarization models. By leveraging this rich dataset, we were able to fine-tune our models to accurately generate concise and informative summaries of research papers. The availability of this dataset on Hugging Face facilitated seamless access and integration into our workflow, enabling us to focus on enhancing the performance and accuracy of our summarization system.

Setting Up API Integrations:
To streamline data collection, sources with available APIs, such as arXiv and other academic databases, will be identified. API integration scripts will be developed and tested to efficiently fetch data. These scripts will be designed to handle large volumes of data and will be configured to run periodically, ensuring the dataset remains updated.

2.4.1 Dataset Details

The primary dataset used for this study was the arXiv dataset, which comprises a vast collection of research papers from various scientific domains. This dataset was chosen for its diversity and relevance to the academic community. The dataset was preprocessed to ensure compatibility with the BART-large-CNN model, including tokenization, normalization, and removal of non-essential elements.

Dataset Repo in huggingface	ccdv/arxiv-summarization
Column Names in Dataset	['article', 'abstract']
Total Number of Research Papers	203037
Number of Research Papers in Train Set	6436
Number of Research Papers in Validation Set	6440
Number of Research papers in Financial Domain	3450

Table 1:Table: Dataset details

2.5 Model Locating/Integration:

Initially, we used OpenAI's GPT model for summarizing our first dataset. However, due to the model's cost structure, which charges based on the number of tokens, we decided to discontinue its use. We then shifted our focus to exploring other available and free large language models to meet our project goals.

i. Falconsai
ii. Bart
iii. Flan-t5-base-summarization
iv. Bigbird-pegasus-large-arxiv

The LLM pipeline has been created for summarization using the loaded model and tokenizer. This pipeline simplifies the process of feeding text to the model and obtaining summaries.

2.6 Accuracy calculation and comparison

The project's effectiveness will be initially assessed through accuracy testing of the summarization model. This will involve evaluating the model's performance using established metrics such as ROUGE, BLEU, and METEOR to compare the generated summaries against human-annotated references. A high correlation with these reference summaries will indicate the model's ability to produce accurate and reliable summaries. Additionally, domain experts will review a sample of the summaries to ensure they are not only accurate but also relevant and informative.

The ROUGE score is a set of metrics used for evaluating automatic summarization and machine translation, focusing on the overlap of n-grams, word sequences, and word pairs between the generated summary and reference summaries. By using the ROUGE score, we can objectively assess the quality of the summaries produced by our models.

Formally, ROUGE-N measures the n-gram recall between a candidate summary and a set of reference summaries. It is calculated as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \quad (1)$$

Figure 6: Rouge calculation formula

Where n stands for the length of the n-gram, gramn, and Countmatch(gramn) is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

We have systematically test the current models and any new models we find on Hugging Face. This process involves running each model on a standardized dataset, calculating their ROUGE scores, and comparing the results. Our goal was to identify the model that delivers the highest accuracy and best performance, ensuring that our summarization tasks are handled with the utmost efficiency and precision.

Model accuracy comparison:

The accuracy of these models was measured using ROUGE metrics (ROUGE-1, ROUGE-2, and ROUGE-L). The results revealed that the BART model significantly outperformed the baseline, achieving the highest f-measure scores across all ROUGE metrics.

	model_name	rouge1_f1	rouge2_f1	rougeL_f1
0	Falconsai	0.151961	0.103448	0.132353
1	Bart	0.507299	0.432234	0.423358
2	flan-t5-base-summarization	0.131818	0.0684932	0.0954545
3	bigbird-pegasus-large-arxiv	0.31068	0.114786	0.182524

```
Best performing models:
ROUGE-1 best model: Bart with F1 score 0.5072992700729928
ROUGE-2 best model: Bart with F1 score 0.4322344322344322
ROUGE-L best model: Bart with F1 score 0.4233576642335766
```

Figure 7: Initial accuracy of selected models

3 Results

After evaluating the accuracy of various models, we selected the best performing model for further development due to its superior performance. We then extracted a subset from our comprehensive dataset, specifically focusing on the financial papers, to fine-tune the model. This subset, comprising 3,450 papers, was used to refine the BART model, ensuring that it was well-adapted to the domain-specific content of financial research. The results of this fine-tuning on both the training and test datasets are discussed below, demonstrating the model's effectiveness in summarizing financial documents.

3.1 Finetuning the model :

We fine-tuned the BART model on our financial research papers by configuring it to train over one epoch with a small batch size and employing techniques such as learning rate warmup and weight decay. The training process included regular evaluations and logging, with model checkpoints managed to retain the best-performing version. This approach ensured that the model was effectively adapted to our specific data while maintaining high accuracy.

```
# Defining training arguments
training_args = TrainingArguments(
    output_dir='./results',          # output directory
    num_train_epochs=1,              # number of training epochs
    per_device_train_batch_size=2,   # batch size for training
    per_device_eval_batch_size=2,    # batch size for evaluation
    warmup_steps=500,                # number of warmup steps for learning rate
    weight_decay=0.01,               # strength of weight decay
    logging_dir='./logs',            # directory for storing logs
    logging_steps=10,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    save_total_limit=2,
    load_best_model_at_end=True,
)
# Initializing Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_tokenized,
    eval_dataset=val_tokenized
)
# Training the model
trainer.train()
```

Figure 8: Training code snapshot on subset of dataset

3.2 Accuracy calculation after the finetuning of the selected model

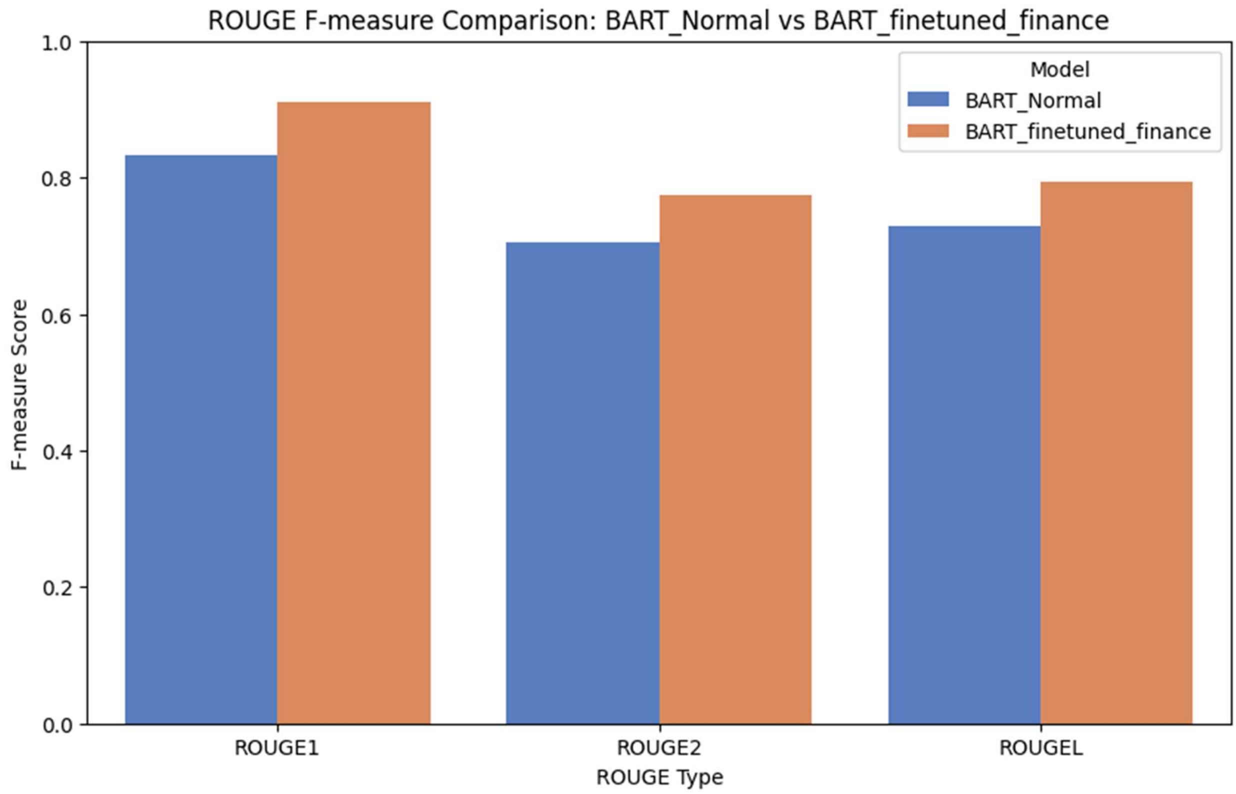


Figure 9: Accuracy of finetuned model on train dataset

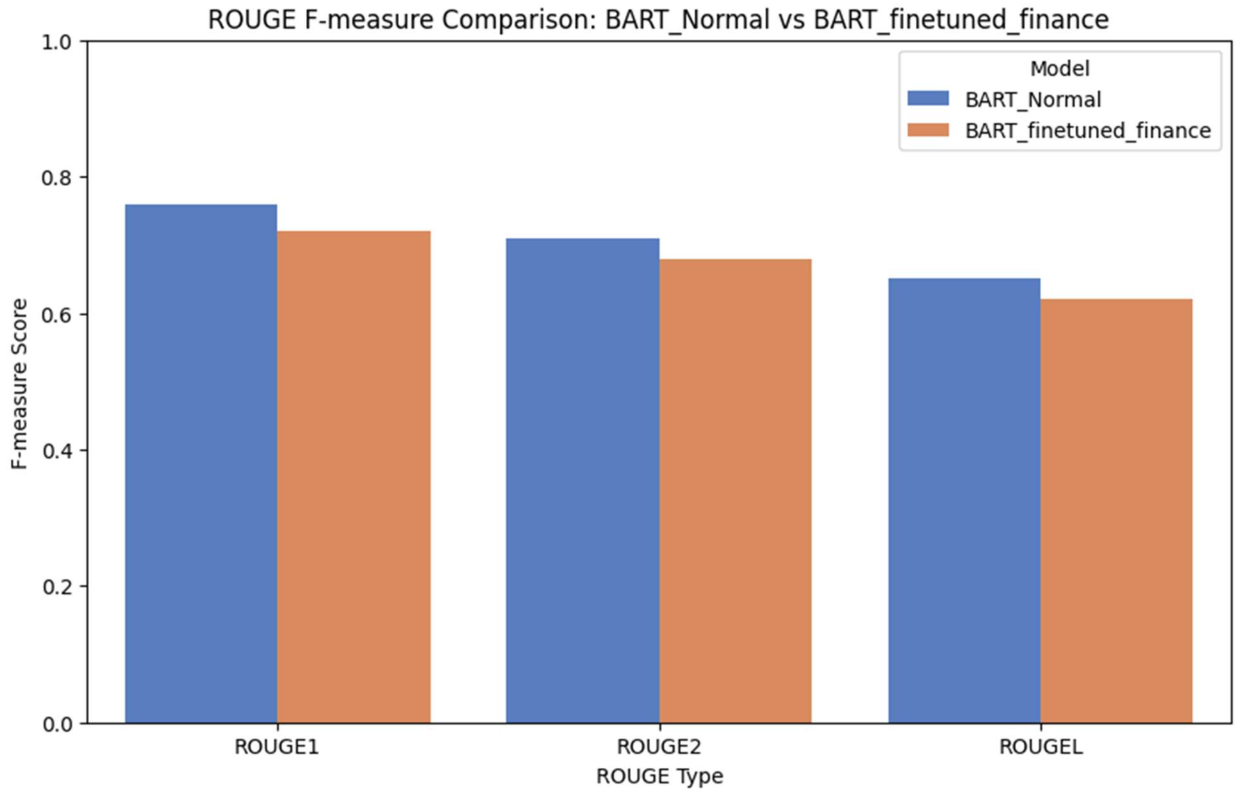


Figure 10: Accuracy of model on test set

Discussion:

The evaluation of ROUGE scores before and after fine-tuning confirms that the model has achieved satisfactory performance. The minor drop in test set scores suggests successful generalization, as opposed to overfitting. The bar charts visually reinforce that the fine-tuned model remains competitive, with scores that are both high and indicative of effective summarization capabilities. Overall, the fine-tuning process has positively impacted the model's performance, making it well-suited for our project goal.

3.3 User Interface Design

In the User Interface Design phase, we have created an intuitive and user-friendly platform that allows researchers, students, and other stakeholders to easily access and utilize the summarization tool. The interface has designed with a focus on usability, incorporating features such as search functionality, customizable summarization length, and the ability to download summaries in various formats. We have employed user-centered design principles, conducting usability tests and gathering feedback from potential users to refine the interface. The goal was to ensure that the platform not only meets the needs of its users but also integrates seamlessly into their existing workflows, thereby enhancing their productivity and engagement with scientific literature.

The Streamlit framework has been used to develop the user interface and integrate the finetuned model to generate the concise summary and display it in desired format.

User Input Handling: The application provides multiple options for inputting text:

- Direct text input (copy-paste or typing).
- Uploading .txt files.
- Uploading .pdf files.
- Fetching text from Wikipedia URLs.

3.4 Frontend Integration:

- Enhanced the frontend by adding a progress bar to indicate the status of the summarization process. This feature improves user experience by providing real-time feedback on the processing time.
- Improved error handling mechanisms to display user-friendly messages in case of issues

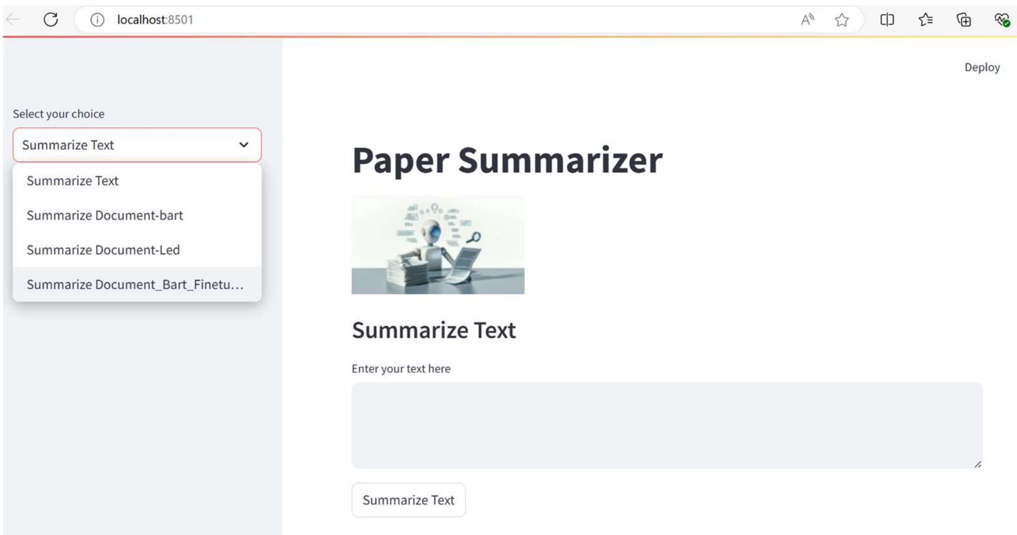


Figure 11: User interface with options on dropdown

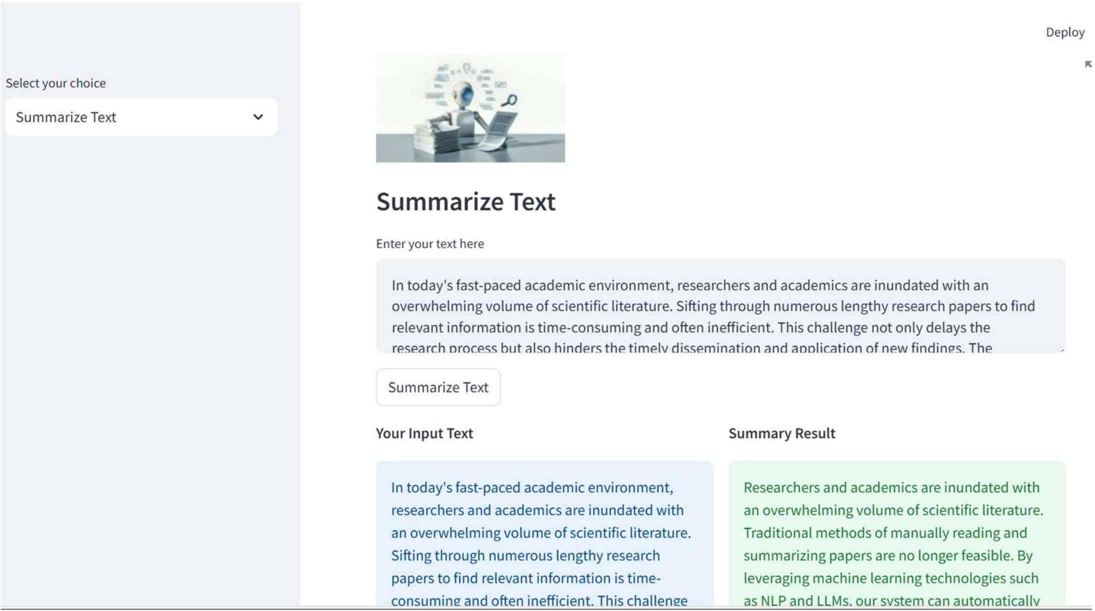


Figure 12: Sample summary generated from text input

Model Inference: For each input method, the application processes the text and uses the ‘LLM Pipeline’ function to generate a summary. This involves:

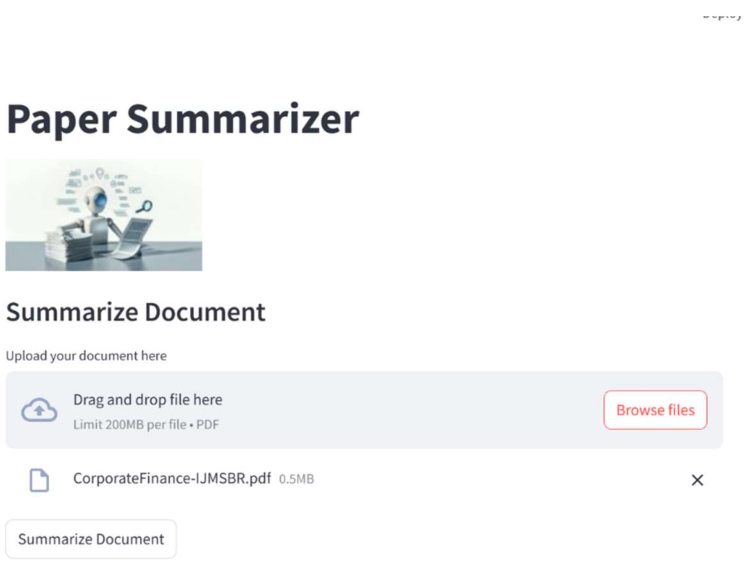
- Tokenizing the text.

- Feeding the tokens to the model.

- Generating a summary based on the model's output.

ii)Result Display: The summarized text is displayed alongside the original input, allowing users to compare and evaluate the quality of the summary. The application also provides options to view and delete uploaded files.

Figure 13: User interface for document upload option



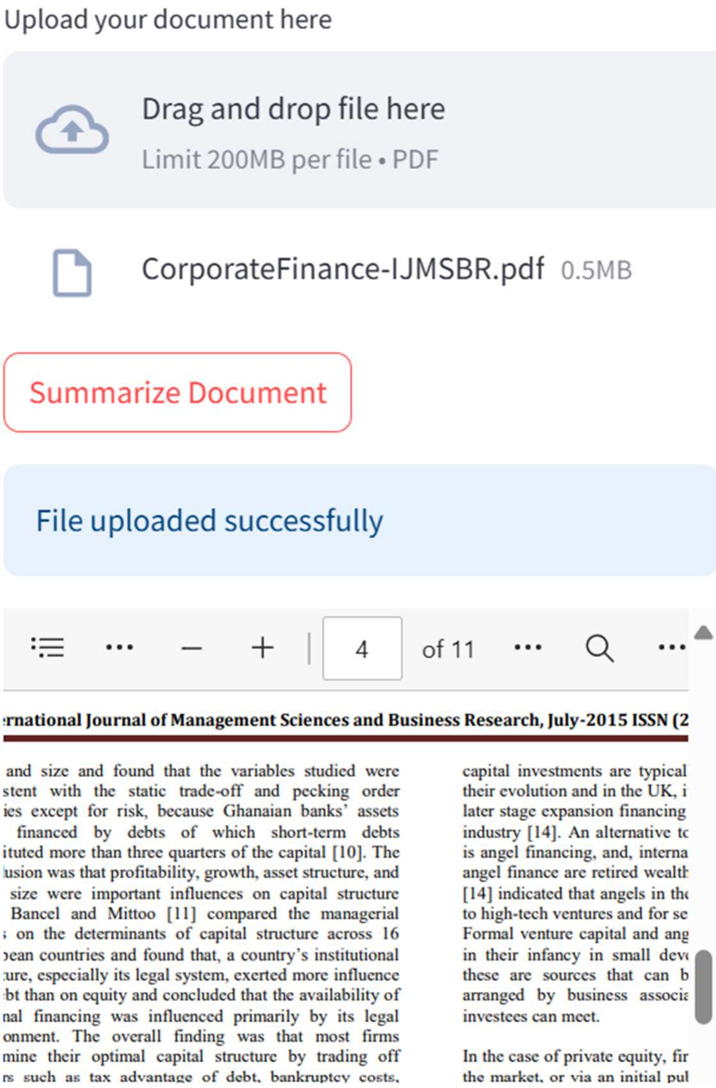


Figure 14: Displaying upload success message and uploaded document overview

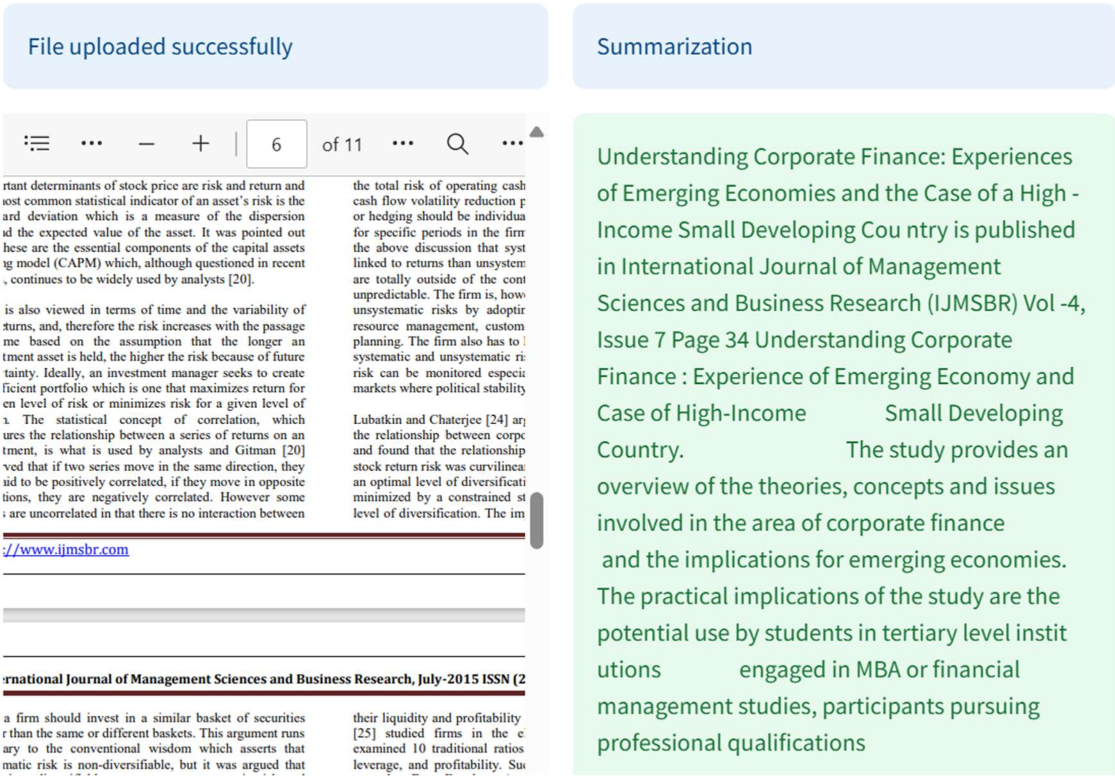


Figure 15: Displaying generated summary on side tab

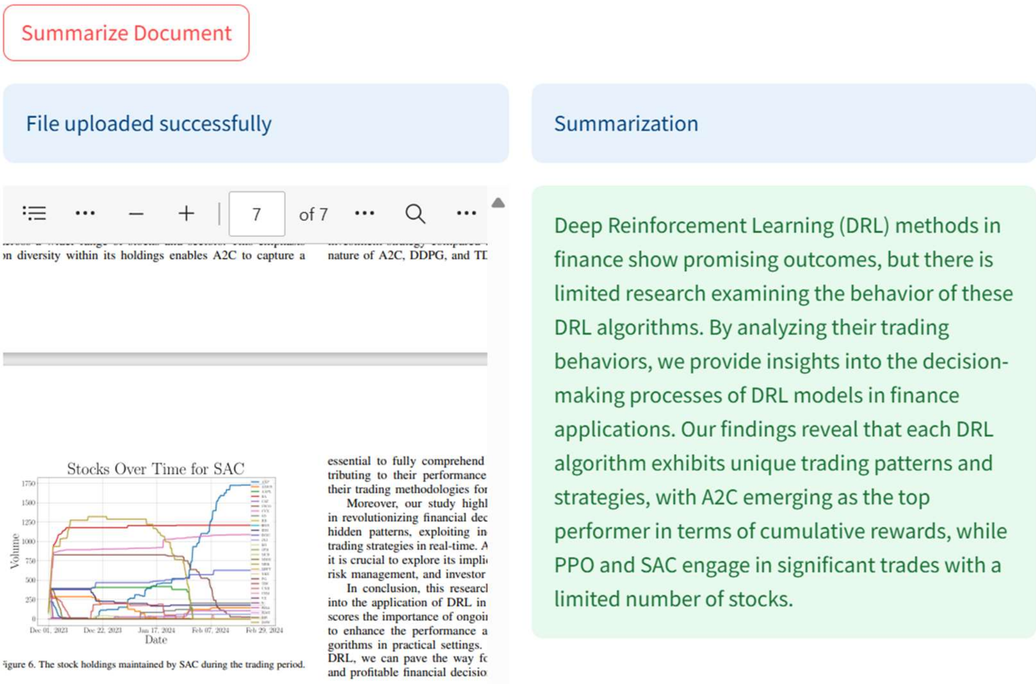


Figure 16: Summarization example of finance paper

3.5 Testing and refinement

In the Testing and Refinement phase, we have systematically evaluate the performance of the summarization model using a variety of metrics, such as ROUGE, BLEU, and human evaluations. This process has involved testing the model on a separate validation dataset that was not used during training to assess its generalizability and robustness across different scientific domains. We have identified the weaknesses or biases in the model's output and implement iterative improvements based on those findings. Techniques such as hyperparameter tuning, error analysis, and incorporating additional data has been employed to enhance model accuracy and reliability. Continuous feedback loops with domain experts will be added to ensure that the refinements align with the needs and expectations of the scientific community.

3.6 Usability Testing

Usability testing focus on the user interface and overall user experience. Potential users, including researchers and students, will be invited to interact with the summarization tool and provide feedback on its ease of use, functionality, and design. This testing will involve tasks such as inputting papers, retrieving summaries, and customizing summary lengths. Insights gained from these sessions has been used to refine and optimize the interface, ensuring it meets the needs and expectations of its target audience.

3.7 Feedback Collection

Analyzing the feedback will help identify areas for enhancement and ensure the tool evolves in line with user needs. During the project, feedback was collected from potential users and fellow colleagues. Their input has played a crucial role in improving the system, and this feedback process will continue in the future.

3.8 Impact Assessment

The final step in the evaluation process is an impact assessment to measure the tool's overall contribution to the research community. This involve tracking key performance indicators (KPIs) such as the reduction in time spent on literature review, the number of active users, and the frequency of use. Additionally, we will seek testimonials and case studies from users to understand how the tool has influenced their research productivity and collaboration. A positive impact on these metrics indicate the project's success in achieving its goals.

3.9 Challenges faced:

One of the major challenges we encountered was the slower-than-expected performance of the models in generating summaries. Despite utilizing GPU acceleration on Google Colab, the processing time required for the models to generate summaries was substantial. This slower performance affected the responsiveness of our application, diminishing the user experience during real-time summarization tasks. Additionally, the high memory requirements for fine-tuning and frequent session crashes posed significant issues, making the availability of computing resources a primary challenge.

The large size of the arXiv dataset further compounded these difficulties, as managing and processing such a voluminous dataset placed additional strain on our computational resources. These challenges underscored the necessity for more efficient resource allocation and optimization strategies to ensure faster summary generation and a smoother overall performance of the system.

4 Conclusions and Future Work

In summary, this project has developed a robust and user-friendly tool for summarizing scientific research papers using advanced machine learning techniques. By systematically collecting and annotating a comprehensive dataset, developing and refining a state-of-the-art summarization model, and designing an intuitive user interface, we have created a valuable resource for the academic community. Thorough evaluation through accuracy testing, usability testing, feedback collection, and impact assessment has ensured the tool's effectiveness and relevance.

Ultimately, this project has enhanced research efficiency, foster collaboration, and facilitate quicker dissemination of knowledge, significantly benefiting to the scholars and academicians.

Future work of this project includes:

- Train the model on rich resources with more epochs, using regularization techniques to handle overfitting and achieve high accuracy.
- Deployment to cloud platform for easy access.
- Expand Language Support: Add multi-language capabilities to accommodate a wider range of research papers.
- Enhance Summarization Quality: Continuously refine the model with diverse datasets and user feedback for improved accuracy.
- Real-Time Summarization: Develop features for automatic summarization of newly published papers and conference proceedings.
- Integrate Additional Databases: Broaden the system's reach by connecting with more academic databases.

References

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150.*

Data Pipeline architecture - https://www.montecarlodata.com/blog-data-pipeline-architecture-explained/

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Ekaterina Zolotareva, Tsegaye Misikir Tashu and Tomáš Horváth , https://ceur-ws.org/Vol-2718/paper28.pdf

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. Information Sciences Institute, University of Southern California.

Liu, P., & Saleh, M. (2021). Text Summarization Techniques: A Brief Survey. arXiv preprint arXiv:2101.00133.

Moradi, M., & Samwald, M. Clustering of Deep Contextualized Representations for Summarization of Biomedical Texts. *Institute for Artificial Intelligence and Decision Support, Center for Medical Statistics, Informatics, and Intelligent Systems, Medical University of Vienna, Vienna, Austria.*

Verleysen, N. (2024). *arxiv-summarizer* GitHub repository. GitHub. https://github.com/VerleysenNiels/arxiv-summarizer

Sneha Thange, Jayesh Dange, Vivek Karjule, Janhavi Sase, https://www.researchgate.net/publication/375120640_A_Survey_on_Text_Summarization_Techniques

Su, X. (2021). Using the T5 Model for Clinical Multi-Task Learning. GitHub Projects. .

Yadav, M., Gupta, S., Sahu, N. K., & Lone, H. R. (n.d.). Fine-tuning large language models for automated diagnostic screening summaries. Manuscript in preparation. arXiv.

Appendix

Responsibilities of team members:

Activity	**Person Responsible**
Data Collection	Ronak
Model Development/locating and Training	Bibek
User Interface Design	Jumana and Guneet
Testing and Refinement	Pratikkumar
Documentation and Training	All members

Table 2: Responsibilities division of the project

GitHub link of the project: https://github.com/sbibek51/Research-Paper-Summarizer

Video Presentation link: Team video presentation link