

SUMMARIZATION AND QUERYING OF PDF USING GENERATIVE ARTIFICIAL INTELLIGENCE

**Dissertation Submitted
in Partial Fulfillment of the Requirements for the
Degree of**

MASTER OF SCIENCE

in

MATHEMATICS

by

**Mihika Jain
(2K22/MSCMAT/24)**

**Somya Rao
(2K22/MSCMAT/44)**

**Under the Supervision of
Prof. Anjana Gupta, Delhi Technological University**



Department of Applied Mathematics

**DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultpur, Main Bawana Road, Delhi – 110042, India**

May, 2024



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultapur, Main Bawana Road, Delhi – 110042, India

ACKNOWLEDGEMENT

We are deeply grateful to our supervisor, Prof. Anjana Gupta, for her time and efforts she provided throughout the year, which provided us with the opportunity to conduct extensive research and learn many new skills. We would like to extend our sincere thanks to all the researchers whose work is cited in the references section, as their contributions were crucial to the completion of this dissertation. As a result of this collaboration, we are both extremely grateful for each other's hard work and determination towards this dissertation. We extend our gratitude to Delhi Technological University for giving us this opportunity.

Thanking You

Mihika Jain

Somya Rao



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultapur, Main Bawana Road, Delhi – 110042, India

CANDIDATE'S DECLARATION

We, Mihika Jain (2K22/MSCMAT/24) and Somya Rao (2K22/MSCMAT/44), hereby certify that the work which is being presented in the Dissertation entitled “Summarization and Querying of PDF using Generative Artificial Intelligence” in partial fulfilment of the requirements for the award of the Degree of Master of Science, submitted in the Department of Applied Mathematics, Delhi Technological University is an authentic record of my own work carried out during the period from August,2023 to May,2024 under the supervision of Prof. Anjana Gupta.

The matter presented in the dissertation has not been submitted by us for the award of any other degree of this or any other Institute.

Candidate's Signature

This is to certify that the student has incorporated all the corrections suggested by the examiners in the Dissertation and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor

Signature of External Examiner



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi – 110042, India

CERTIFICATE BY THE SUPERVISOR

Certified that Mihika Jain (2K22/MSCMAT/24) and Somya Rao (2K22/MSCMAT/44) have carried out their search work presented in the Dissertation titled “Summarization and Querying of PDF using Generative Artificial Intelligence” for the award of the Degree of Master of Science, submitted in the Department of Applied Mathematics, Delhi Technological University, Delhi under my supervision. The dissertation embodies results of original work, and studies are carried out by the students themselves and the contents of the dissertation do not form the basis for the award of any other degree to the candidates or to anybody else from this or any other University/Institution.

Prof. Anjana Gupta

Professor

Dept. of Applied Mathematics

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

Date:

Place:

Summarization and Querying of PDF using Generative Artificial Intelligence

Mihika Jain (2K22/MSCMAT/24)

Somya Rao (2K22/MSCMAT/44)

ABSTRACT

Generative AI is a branch of artificial intelligence dedicated to creating new content, including text, images, audio, and video, by learning from existing data. Although still in its nascent stages, this technology is quickly reshaping the way we create and engage with content, holding the potential to revolutionize multiple industries. Its versatility is evident in its diverse applications, ranging from writing and music to data production. As we look to the future, Generative AI is poised to bring even more intriguing advancements, impacting various sectors and necessitating a thorough understanding of its ethical considerations and risk management.

Nowadays, people face the problem of information overload in lengthy and unstructured PDFs, making it time-consuming and challenging to quickly extract and comprehend key details and information from the document. Then technique of summarization and querying of PDF can be useful. So that is when this paper comes into the picture which allows to provide the summary of PDF documents of varying lengths and retrieve desired information from the document in a concise and coherent form.

The aim of this study is to summarize and query PDF documents using Generative AI and LangChain framework. And further assess the efficacy and quality of generated summaries and the relevance of query outputs. To summarize PDFs, the Stuff Documentation Chain method and the Map Reduce method are employed depending upon the length of the PDF. The Stuff Documentation Chain method is used to summarize small PDFs (preferably consisting of 1 to 2 pages). Hence, we used this technique to summarize a single-page PDF. Then to summarize a 4-page PDF, we applied the MapReduce method, which is suitable for handling large documents that exceed the token limit of a single prompt. Further, we performed Querying of PDF using LangChain framework to unsheathe the desired answers from the PDF. LangChain facilitated the extraction of key information and generation of concise summaries. Also, while querying, it enabled efficient processing of user queries and retrieval of relevant information from PDFs. After applying all these techniques on the PDFs, we retrieved summaries of the respective PDFs and answers addressing to the various queries. Next, we evaluated the relevancy of the query outcomes as well as the effectiveness and calibre of the summaries that were produced. The summarization techniques efficiently condensed the content of PDFs while retaining crucial details, resulting in a well-structured summary which provided a comprehensive overview of

the document's content. The responses to the queries had a high level of pertinence and directly addressed the questions posed.

Keywords: Generative Artificial Intelligence, LangChain, Stuff Document Chain method, Map-Reduce method, Querying of PDF

TABLE OF CONTENTS

Acknowledgement	ii
Candidate's Declaration	iii
Certificate by the Supervisor	iv
Abstract	v
List of Figures	viii
Chapter 1 INTRODUCTION	1-9
1.1 About Generative Artificial Intelligence	1
1.1.1 Application and Use Cases of Generative AI In various fields	2
1.1.2 Advantages of Generative AI	4
1.1.3 Disadvantages of Generative AI	5
1.1.4 Working of Generative AI	6
1.1.5 Large Language Models (LLMs)	7
1.1.6 LangChain	7
1.2 Related Work	8
Chapter 2 RESEARCH METHODOLOGY AND SOLUTION APPROACH	10-18
2.1 Stuff Document Chain Method	11
2.2 Map-Reduce Method	13
2.3 Querying of PDF	16
Chapter 3 DISCUSSION AND RESULTS	19-20
Chapter 4 CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT	21-23
4.1 Conclusion	21
4.2 Future Scope	22
4.3 Limitations	23
4.4 Social Impact	23
REFERENCES	24
PLAGIARISM REPORT	25

LIST OF FIGURES

Figure 2.1: Figure explaining Stuff Document Chain and Map-Reduce methods

Figure 2.2: Part of code for Stuff Document Chain method

Figure 2.3: Part of code for Stuff Document Chain method

Figure 2.4: Part of code for Stuff Document Chain method

Figure 2.5: Part of code for Stuff Document Chain method

Figure 2.6: Part of code for Map-Reduce method

Figure 2.7: Part of code for Map-Reduce method

Figure 2.8: Part of code for Map-Reduce method

Figure 2.9: Part of code for Map-Reduce method

Figure 2.10: Part of code for Querying of PDF

Figure 2.11: Part of code for Querying of PDF

Figure 2.12: Part of code for Querying of PDF

Figure 2.13: Part of code for Querying of PDF

Figure 2.14: Part of code for Querying of PDF

Figure 3.1: Output for Stuff Document Chain method

Figure 3.2: Output for Map-Reduce method

Figure 3.3: Output for Querying of PDF

Figure 3.4: Output for Querying of PDF

CHAPTER 1

INTRODUCTION

1.1 About Generative Artificial Intelligence

Over the past decade, Artificial Intelligence has consistently been a significant technological topic, but the advent of generative AI has propelled it into global prominence, spurring an unprecedented wave of innovation and adoption. Generative AI has redefined the limits of what machines can achieve by enabling them not only to learn but also to create.

Generative AI is a branch of artificial intelligence dedicated to creating new content, including text, images, audio, and video, by learning from existing data. Using deep learning methods, particularly neural networks, these models can produce outputs that replicate or enhance the patterns found in the training data. This capability allows generative AI to generate realistic and coherent content, making it highly valuable for various applications such as creative arts, data enrichment, and content creation. Notable examples of generative AI models are Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and transformer-based models like GPT-3 and DALL-E.

By understanding the fundamental patterns in the input data, generative AI can generate new outputs that closely mimic the original data. This ability is powered by neural networks, enabling the creation of diverse content such as images, videos, text, and audio. Among the most commonly utilized generative AI models are generative Adversarial Networks (GANs) and transformer-based models like Generative Pre-Trained (GPT) language models. GANs are particularly effective at producing visual and multimedia artifacts from image and text inputs, whereas transformer-based models specialize in generating textual content by utilizing information from the internet.

Typically, Generative AI operates through three phases:

1. Training: Developing a foundation model that serves as the basis for various generative AI applications.

2. Tuning: Customizing the foundation model to suit specific applications.
3. Generation, Evaluation, and Re-tuning: Assessing the application's output and continually enhancing its quality and accuracy.

Generative AI offers numerous benefits, including the ability to create realistic and high-quality content, automate creative tasks, and generate diverse data for training other AI models. Its applications span multiple fields, such as text generation for automated writing and summarization, image synthesis for design and entertainment, and music composition. In the medical field, generative AI assists in drug discovery by simulating molecular structures and predicting their interactions, and enhances medical imaging by generating high-resolution images for better diagnosis. Additionally, generative AI can improve data augmentation, increasing the robustness of machine learning models, and support innovative developments in fields like 3D modeling. This versatility makes generative AI a powerful tool across numerous industries.

However, generative AI also poses challenges. The technology is intricate and requires significant computational resources and vast training data. Additionally, ethical concerns emerge regarding potential misuse, such as generating deepfakes or disseminating misinformation.

The impact of generative AI on the realm of artificial intelligence is significant and widespread, with vast potential yet to be fully explored.

1.1.1 Application and use cases of Generative AI in various fields

Content generation: images, videos, music

Generative AI models, like GANs (Generative Adversarial Networks), have the capability to produce original content, spanning images, videos, and music. Notably, tools such as DALL-E, Stable Diffusion, and Midjourney demonstrate the widespread application of generative AI in visual content creation.

The drive to enhance both the quality and variety of generated content is gaining momentum. Researchers are actively investigating methods to generate 3D scenes from static 2D images utilizing AI technology.

Automated custom software engineering

Generative AI finds application in software engineering, facilitating the automated development of tailored software. An exemplar is GPT-3, which can generate code from natural language descriptions. Ongoing research concentrates on enhancing the precision and effectiveness of code generation. For instance, endeavours are underway to refine AI models for a deeper comprehension of code context, thus enabling the generation of more precise and efficient code.

Entities like GitHub harness AI to aid developers in code composition, reflecting the industry's commitment to leveraging AI for software engineering advancements.

Text-to-speech solutions

Generative AI models are capable of converting text into speech, producing human-like audio. These models are valuable for applications such as voice assistants, text-to-speech services, and more. Current research aims to enhance the naturalness and expressiveness of the generated speech, making it sound more lifelike and engaging.

Healthcare solutions

Generative AI models have applications in healthcare, such as generating new drug molecules and predicting progression of diseases. These models analyze existing medical data to produce fresh insights, accelerating research and treatment processes. Current research aims to enhance the precision and pertinence of these medical insights. For example, researchers are working on models that accurately forecast disease advancement and produce more potent drug compounds. Companies like DeepMind leverage AI to support healthcare research. DeepMind's AlphaFold, for example, utilizes AI to forecast the 3D configuration of proteins, a critical factor in understanding diseases and creating new drugs.

Media and entertainment

The media and entertainment industry can leverage generative AI in various ways, as both focus on creating unique content. Generative AI can assist in producing and editing visual content, generating short highlight videos of sporting events, and simplifying the use of content management systems.

Financial services

Generative AI in financial services has applications in fraud detection, risk assessment, and portfolio optimization. It creates synthetic data for model training, enhancing privacy. Utilizing deep learning, it predicts market trends and customer behavior, supporting algorithmic trading and personalized financial recommendations. This technology optimizes investment strategies and enhances customer experiences.

Business

Generative AI in business can be applied to content generation for marketing, optimizing product design through generative design, and delivering personalized customer experiences. It boosts product innovation, automates creative processes, and

customizes offerings to individual preferences, thereby enhancing competitive advantage and customer satisfaction.

1.1.2 Advantages of Generative AI

Generative artificial intelligence offers important advantages in diverse fields by generating new data and content similar to existing examples. One of the primary strengths of generative AI lies in its capability to significantly streamline and enhance creative processes. By creating content based on specific criteria, generative AI reduces the time and effort required for completing creative projects, which is particularly beneficial in high-demand industries such as marketing and advertising.

Generative AI enables businesses to explore extensive design spaces effectively. Generative design algorithms can produce numerous iterations for fields like product design and architecture based on input parameters and constraints. This not only speeds up the design process but also uncovers innovative solutions that might not be identified through traditional methods.

Another critical feature of generative AI is its capability to personalize experiences and recommendations. By leveraging large datasets of user preferences and behaviors, generative models can provide personalized recommendations for products, services, and content. This degree of customization boosts customer satisfaction and involvement, resulting in higher loyalty and retention rates.

Generative AI also aids in improving decision-making processes. In areas like finance and healthcare, generative models can simulate various scenarios and generate predictive insights based on historical data. This allows businesses to anticipate potential outcomes, mitigate risks, and seize opportunities more effectively.

Moreover, generative AI fosters innovation by exploring new concepts and ideas. Businesses can discover unconventional solutions and drive breakthrough innovations by generating diverse alternatives and thoroughly exploring the design space. This capability is particularly valuable in research and development, where innovation is essential for staying competitive and addressing complex challenges.

In summary, generative AI enhances the efficiency of creative tasks, improves personalization, supports better decision-making, and encourages innovation. As businesses continue to adopt generative AI technologies, they will unlock new opportunities for growth, differentiation, and value creation across various industries.

1.1.3 Disadvantages of Generative AI

Generative AI technologies have introduced significant ethical considerations in various domains, including copyright issues, misuse, data privacy concerns, data provenance, and the amplification of existing biases. Copyright Issues arise because generative AI can create content that closely resembles copyrighted material, creating uncertainties around ownership and legality. Determining the boundaries of fair use is challenging when AI-generated content blurs the lines between original and reproduced works.

Misuse of generative AI raises concerns about deceptive practices, such as deepfakes and fabricated information. These technologies can be exploited for malicious purposes, including spreading disinformation, impersonation, and defamation. Ensuring safeguards against such misuse is crucial to maintaining trust and information integrity.

Data privacy concerns are significant because generative AI models often rely on large datasets for training. If these datasets include sensitive or personal data that is not adequately shielded or anonymized, privacy breaches and security risks can occur. Stricter regulations and robust data protection measures are essential to address these issues.

The amplification of existing biases is a critical ethical issue, as generative AI models can perpetuate and even magnify biases present in the training data. If the datasets used are biased or unrepresentative, the generated content may reinforce stereotypes and discrimination. Addressing this challenge requires ensuring diverse and unbiased training data and continuously evaluating and mitigating biases in algorithms.

Workforce roles and morale are also impacted by generative AI. With AI systems capable of automating tasks and generating content previously performed by humans, there is a potential risk of job displacement and uncertainty about future employment opportunities. This can lead to decreased job security and lower morale among employees. Organizations must prioritize transparent communication, provide opportunities for reskilling, and foster a supportive work environment that values human creativity and collaboration to address these concerns.

Hence, Human-AI collaboration should be emphasized, seeking ways to augment human work with AI technology and identifying new roles where human expertise can complement generative AI systems. By involving employees in decision-making and offering growth opportunities, organizations can alleviate workforce anxieties and promote the ethical integration of generative AI technology.

1.1.4 Working of Generative AI

Generative AI encompasses artificial intelligence models and techniques designed to produce new data, including text, images, audio, or video, by learning patterns and relationships from training data. These models employ deep learning algorithms, particularly neural networks, to grasp the underlying structure and features of the training data, enabling them to generate new, creative, and coherent content that resembles or extends the original data.

The general workflow of how generative AI operates can be outlined in the following steps:

1. **Data Collection:** A substantial dataset of pertinent training examples is gathered. For instance, generating realistic images requires a dataset comprising millions of images, while text generation necessitates a large corpus of textual data.
2. **Data Preprocessing:** The collected data undergoes preprocessing and formatting to be efficiently utilized by the neural network model. This process can involve tokenization (breaking text into smaller units), normalization, or conversion to a suitable representation.
3. **Model Architecture:** The neural network architecture is chosen or designed based on the generative task. Commonly used architectures in generative AI include Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Transformers (such as GPT and DALL-E), and Diffusion Models (such as Stable Diffusion).
4. **Model Training:** The selected neural network model is trained using specialized training algorithms and objectives. During this phase, the neural network learns to capture the underlying patterns, distributions, and relationships within the data.
5. **Latent Space Encoding:** Many generative models, including VAEs and GANs, encode the training data into a lower-dimensional latent space representation. This latent space captures the essential characteristics and features of the data in a compressed and disentangled form.
6. **Sampling and Model Generation:** Post-training, the model can generate new data by sampling and generating. This involves the model producing new content based on its learned patterns (such as text generation with GPT models or image generation with diffusion models) either by sampling from the latent space distribution (as with VAEs and GANs) or by providing a seed or prompt to the model.
7. **Output Generation:** The generated output, whether text, images, audio, or video, is then post-processed or decoded into the desired format for

consumption or further analysis. Depending on the application, the output may be formatted for display, printing, or storage. Additionally, the output may undergo further processing to create derivative data, such as annotations or metadata. Finally, the output might be combined with other inputs to create more complex data.

This comprehensive approach ensures that generative AI can effectively produce high-quality, coherent, and creative content across various domains.

1.1.5 Large Language Models (LLMs)

A Large Language Model (LLM) is a type of artificial intelligence (AI) program designed for recognizing and generating text, among other tasks. These models are named for their scale, being trained on extensive datasets. LLMs utilize machine learning principles and primarily leverage transformer models, a specific kind of neural network. As the text generation component of generative AI, LLMs can both produce and interpret text inputs.

Using deep learning techniques and transformer architecture, Large Language Models (LLMs) analyze text data to predict the next word in a sequence. This involves converting tokens into numerical embeddings and considering the context provided by preceding words. By training on vast text corpora, LLMs develop an understanding of grammar and semantics, enabling them to generate coherent text for various applications.

To improve accuracy, LLMs undergo refinement techniques like fine-tuning and reinforcement learning with human feedback. These methods help reduce biases and errors, ensuring the generated text is reliable and usable. LLMs are versatile and can be trained for diverse tasks.

A key application of LLMs is their role in generative AI. When given prompts or questions, these models can produce text-based responses. For instance, the publicly available LLM ChatGPT can generate essays, poems, and other forms of text in response to user inputs.

1.1.6 LangChain

LangChain is a framework designed to help developers build applications using language models more efficiently. It acts as a toolkit, facilitating the creation of applications that comprehend and generate human language, such as chatbots, content creators, and language-based search engines. LangChain simplifies the integration and utilization of advanced language models in a practical and effective way. It supports a wide range of use cases for LLMs and natural language processing (NLP), including chatbots, intelligent search, question-answering, summarization services, and virtual agents capable of robotic process automation.

At the core of LangChain is a development environment that simplifies the programming of LLM applications through abstraction—simplifying code by representing complex processes as named components that encapsulate all their constituent steps. LangChain allows developers to tailor language models to specific business needs by defining the required actions to attain the desired outcomes.

Chains: Chains are sequences of automated actions that process a user's query to generate a model's output. They integrate different AI components to provide context-sensitive responses. Chains can perform tasks like connecting to data sources, generating content, translating languages, and answering queries.

Links: Links are individual actions within a chain. They decompose complex tasks into smaller, manageable steps, such as formatting user input, sending queries to an LLM, retrieving data, and translating languages. Links can be reordered to create different AI workflows, offering flexibility in processing.

1.2 Related Work

Ooi et al. (2023) [2] stated that generative AI utilizes machine learning, neural networks, and other techniques to generate new content (e.g. text, images, music) by analysing patterns and information from the training data. Diverse perspectives are presented in this paper concerning the application of generative artificial intelligence in areas such as marketing, healthcare, human resources, learning, banking services, retailing, workplace, and manufacturing. Additionally, the report stressed upon the immense potential that generative AI has, but also pointed out that there are still limitations, such as privacy and ethical dimensions, as well as data ownership concerns. Therefore, they concluded that discussing the opportunities and challenges involved in integrating generative AI in different areas will spur future research.

Ramdurai et al. (2023) [4] mentioned that Generative models, such as generative adversarial networks (GANs) and variational autoencoders (VAEs), form the foundation of generative AI. Research on Generative AI and its impact on the industry is discussed in this article. The authors provided detailed information about the algorithms behind generative AI. The GAN is composed of two components: a generator network and a discriminator network, which generate and evaluate content in competition. VAEs create new samples using an encoder- decoder architecture. Using models such as Generative Adversarial Networks (GANs), generative AI algorithms can learn to generate content that closely simulates real-world data, from images to text to music.

Lakshmi S et al. (2023) [1] mentioned that to initiate the assessment, we utilize "LangChain," an advanced natural language processing framework designed for summarization tasks. In order to summarize PDF content, LangChain uses key insights and context to create a condensed version. The researchers used benchmarking metrics to compare the summarized content and user understanding of PDF content. A score is calculated to measure the similarity between two sources of information.

Topsakal and Akinci (2023) [5] stated that LangChain, an opensource library, stands out due to its proficiency in integrating with diverse data sources and applications, making it an influential tool in the AI community. The authors also showed how to summarize a PDF using Stuff Document Chain and MapReduce techniques, and how to query a PDF using LangChain functionality.

Pesaru et al. (2023) [3] demonstrated the potential of using LangChain and LLM Model to build chatbots that can interact with users in a natural and informative way. To create a chatbot that can answer PDF file questions, they use LangChain and LLM Model. Using a dataset of PDF files, the chatbot is trained, and it generates text responses based on the LLM Model. With the help of Google Search, the chatbot is also able to access and process information from the real world, which allows it to provide more comprehensive and informative answers. Pinecone was used to store PDF file vectors. They have used vector store Pinecone to store embeddings and PDFs in text for future retrieval of related documents.

CHAPTER 2

RESEARCH METHODOLOGY AND SOLUTION APPROACH

If you have a collection of PDFs and aim to summarize and query their content, leveraging LLMs (Large Language Models) can be highly effective due to their advanced capabilities in comprehending and synthesizing text. Here will demonstrate the process of summarizing and querying PDFs using Generative AI and LangChain.

One significant consideration in the summarization process is how to input your documents into the context window of the LLM. Two prevalent strategies for this include:

Stuff Document Chain Method

Map-reduce Method

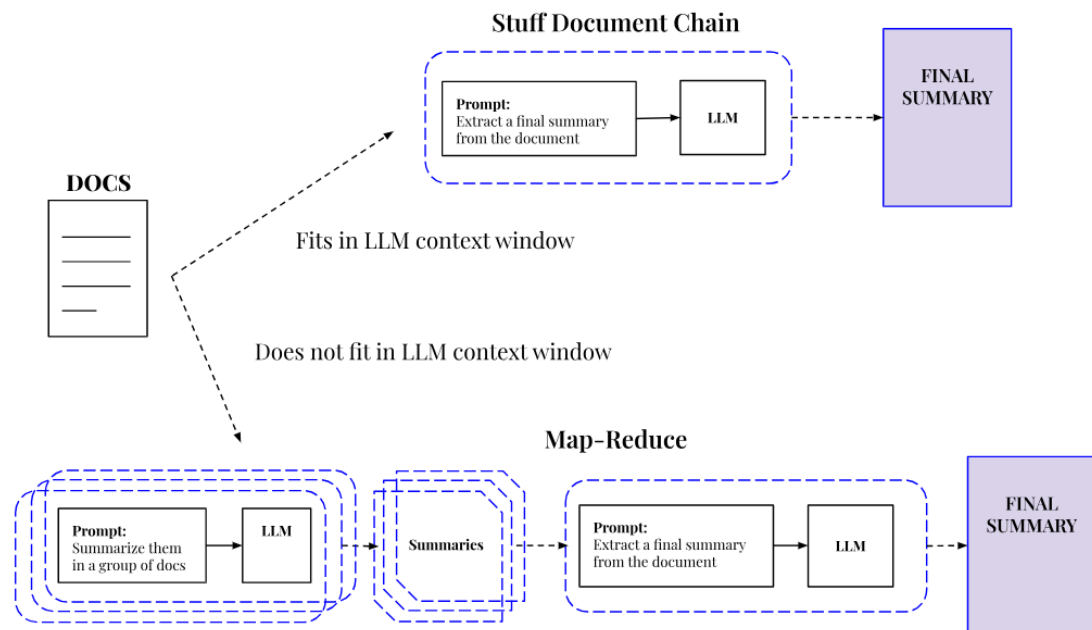


Figure 2.1: Figure explaining Stuff Document Chain and Map-Reduce methods

2.1 Stuff Document Chain Method

The "Stuff Document Chain" is a default chain offered by LangChain, tailored for summarization tasks. The Stuff Document Chain method is a summarization method to condense the contents of documents like PDFs into a summary. The objective of this approach is to extract essential information from the document and combine it into a cohesive summary. This approach is suitable for scenarios where documents are compact. However, if the tokens within the documents surpass the LLM's capacity, the stuff chain would not be effective. For handling larger volumes of data, alternative methods need to be employed.

```
✓ [1] |pip install openai
    Show hidden output

✓ [2] |import openai

✓ [3] |pip install langchain
      |pip install streamlit
      |pip install tiktoken
      |pip install unstructured
      |pip install pdf2image
      |pip install pdfminer
      |pip install PyPDF2
      |pip install faiss-cpu
    Show hidden output

✓ [4] |pip install langchain_community
    Show hidden output

✓ [5] |import os
      |open_api_key="sk-proj-gzJGp4U3Xf56w10cpUqRT3B1bkF3Rm6m0F91Bg1LAGrJxzUw"
      |os.environ["OPENAI_API_KEY"]=open_api_key
```

Figure 2.2: Part of code for Stuff Document Chain method

Firstly, we install all the required libraries and further import the necessary libraries from them. Then we set up the environment in which we use the API key which we generated. The API key will be used in the LLM models when they will be initialized.

```

from langchain.chat_models import ChatOpenAI
from langchain.schema import(
    AIMessage,
    HumanMessage,
    SystemMessage
)

from langchain.chains import LLMChain
from langchain import PromptTemplate

from PyPDF2 import PdfReader

# provide the path of pdf file/files.
pdfreader = PdfReader('/content/resume.pdf')

from typing_extensions import Concatenate
# read text from pdf
text = ''
for i, page in enumerate(pdfreader.pages):
    content = page.extract_text()
    if content:
        text += content

```

Figure 2.3: Part of code for Stuff Document Chain method

The PDF path is further provided and we try to read the PDF with the help of PdfReader. Now to read each and every page, we imported two more libraries, from typing extensions we imported Concatenate. Then we enumerate inside the pages of the PDF and extract this text and put it inside the variable “content”. Then putting all the content inside the variable text. At last, we can see what the raw text looks like.

```

text

'Male, 22 MAYANK JAIN \n
+91 9310152072 | mayank.jain45@gmail.com
\n
https://www.linkedin.com/in/m_ayank_-jain-0b994521a/ \n \nCareer Objective \nPassionate about data analytics with a degree in mathematics . Se
eking opportunities where I can utilize my analytical, \nmathematical and technical skills to solve real life problems related to analysing a bi
g volume of datasets to draw insights that can \nhelp with business decisions . \nAcademic Background \nQualification Year Institute Name %
/ CGPA \nMSC. Applied Mathematics 2024 Delhi Technological University 6/10 \nBachelor of Science [Mathematics (H)] 2022 Lakshmbai Colleg
e \n(University of Delhi) 8.378 /10 \nXII 2019 N. K. Bagrodia...'

from langchain.docstore.document import Document

docs = [Document(page_content=text)]
docs

Show hidden output

llm = ChatOpenAI(temperature=0, model_name='gpt-3.5-turbo')

from langchain import PromptTemplate
from langchain.chat_models import ChatOpenAI
from langchain.chains.summarize import load_summarize_chain
from langchain.docstore.document import Document

```

Figure 2.4: Part of code for Stuff Document Chain method

Then we convert the entire text in the form of a document. This is our main aim by using the stuff documentation chain method, that is, the entire text is pushed inside a document and then given to the LLM model and get the required response of the task,

here it is text summarization. Moving on, we initialize the ChatOpenAI. We will import and load some important libraries.

```

template = '''Write a short and concise summary of the following pdf.
pdf: {text}
'''
prompt = PromptTemplate(
    input_variables=['text'],
    template=template
)

chain = load_summarize_chain(
    llm,
    chain_type='stuff',
    prompt=prompt,
    verbose=False
)
output_summary = chain.run(docs)

output_summary

'Mayank Jain is a 22-year-old male with a passion for data analytics and a degree in mathematics. He is seeking opportunities to utilize his analytical, mathematical, and technical skills to solve real-life problems related to analyzing large datasets. Mayank has experience in programming languages such as Python and SQL, as well as data analysis tools like Pandas and Machine Learning. He has completed various academic projects in data science, including predicting housing prices in Mexico and customer segmentation in the US. Mayank has also held positions of responsibility in various organizations and has completed internships at Viral Fission, where he managed a team of campus ambassadors.'

```

Figure 2.5: Part of code for Stuff Document Chain method

Then we go ahead and write this template according to our requirement. We use `load_summarize_chain` with the following parameters. We set `chain_type` to `stuff`. Finally, we will use this `chain.run` and pass the document that we have created. Hence, we get the output summary of the PDF.

2.2 Map-Reduce Method

Map Reduce is a technique utilized for handling documents when the retrieved data cannot be accommodated within a single prompt. This method involves dividing the chunks of data, along with the query, and forwarding them to a language model. Subsequently, it retrieves responses for each chunk and then employs another language model call to consolidate all individual responses into a final answer. Map Reduce is versatile, capable of processing any number of documents and handling multiple questions concurrently. However, it necessitates more calls and treats each document as independent entities. Essentially, MapReduce simplifies complex data processing tasks by segregating them into two distinct phases: the Map phase and the Reduce phase.

In the Map Phase, the PDF undergoes segmentation into smaller sections, with each section independently summarized by the map function. Subsequently, in the Reduce Phase, the individual summaries generated from the map phase are amalgamated and further condensed by the reduce function to generate a cohesive final summary. This

method adeptly manages large documents by decomposing the task into smaller, more manageable components and subsequently consolidating the outcomes.

```

17s [1] !pip install openai
    Show hidden output

1s import openai

2m [3] !pip install langchain
    !pip install streamlit
    !pip install tiktoken
    !pip install unstructured
    !pip install pdf2image
    !pip install pdfminer
    !pip install PyPDF2
    !pip install faiss-cpu
    Show hidden output

16s [4] !pip install langchain_community
    Show hidden output

0s [5] import os
    open_api_key="sk-proj-gzJGp4U3Xf56w10cpUqRT3B1bkFJRm6m0F91Bg1lAGrJxzUw"
    os.environ["OPENAI_API_KEY"]=open_api_key
  
```

Figure 2.6: Part of code for Map-Reduce method

Firstly, we install all the required libraries and further import the necessary libraries from them. Then we set up an environment in which we used the API key which we generated.

```

0s from langchain.chat_models import ChatOpenAI
    from langchain.schema import (
        AIMessage,
        HumanMessage,
        SystemMessage
    )

0s [7] from langchain.chains import LLMChain
    from langchain import PromptTemplate

1s [8] from PyPDF2 import PdfReader

0s [9] from langchain import PromptTemplate
    from langchain.chat_models import ChatOpenAI
    from langchain.chains.summarize import load_summarize_chain
    from langchain.text_splitter import RecursiveCharacterTextSplitter

0s [10] # provide the path of pdf file/files.
    pdfreader = PdfReader('/content/flowchart.pdf')
    from typing_extensions import Concatenate
    # read text from pdf
    text = ''
    for i, page in enumerate(pdfreader.pages):
        content = page.extract_text()
        if content:
            text += content
  
```

Figure 2.7: Part of code for Map-Reduce method

The PDF path is further provided and we try to read the PDF with the help of PDFReader. Now to read each and every page, we imported two more libraries, from typing_extensions we imported Concatenate. Then we enumerate inside the pages of the PDF and extract this text and put it inside the variable “content”. Then putting all the content inside the variable text.

```

12 llm = ChatOpenAI(temperature=0, model_name='gpt-3.5-turbo')
    Show hidden output

13 [12] llm.get_num_tokens(text)
    4132

14 [13] ## Splitting the text
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=10000, chunk_overlap=20)
    chunks = text_splitter.create_documents([text])

15 [14] len(chunks)
    1

16 [15] chain = load_summarize_chain(
    llm,
    chain_type='map_reduce',
    verbose=False
    )
    Summary = chain.run(chunks)

```

Figure 2.8: Part of code for Map-Reduce method

Moving on, we initialize the ChatOpenAI and then we check the number of tokens present. We take the entire text and generate into chunks. We then split the text using RecursiveCharacterTextSplitter with chunk size 10000. Then we go ahead and write this template according to our requirement. We use load_summarize_chain with the following parameters. We set chain_type to map reduce. Finally, we will use this chain.run and pass the document that we have created.

```

17 [16] Summary
    'Computational Thinking Week 1 focuses on flow charts, which are graphical representations of algorithms used by programmers for program planning. Flow charts use symbols to indicate the flow of information and processing, aiding in communication, program design, debugging, and documentation. However, they can be challenging for large and complex programs. The concept of data sanity is also discussed, including data types and subtypes like characters, integers, booleans, dates, marks, and names. Records, which are data types with multiple fields, are also explained with examples like marks cards, word lists, and shopping bills.'
```

Figure 2.9: Part of code for Map-Reduce method

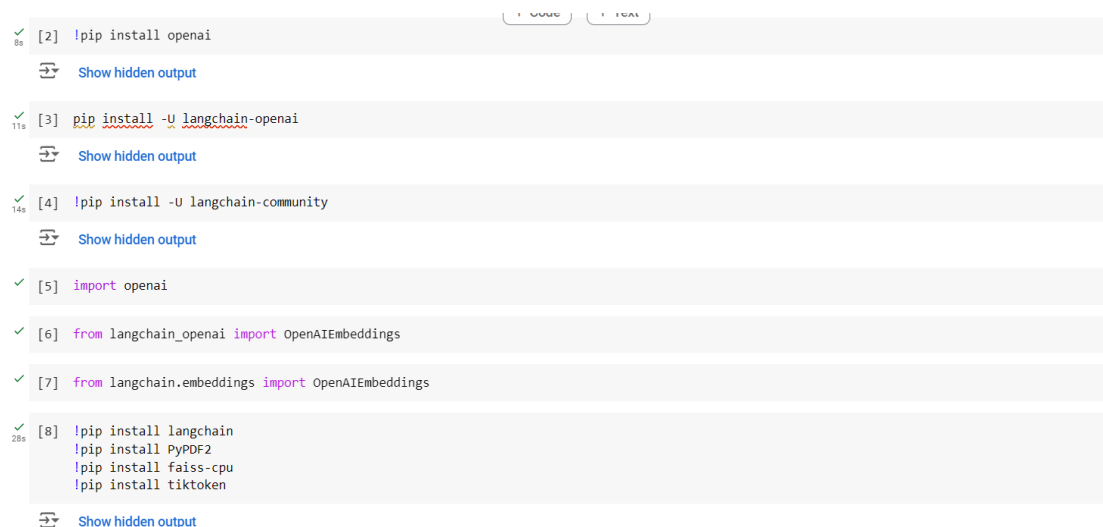
Hence, we get the final output that is the summary of PDF.

2.3 Querying of PDF

Querying a PDF with LangChain entails importing the PDF into the framework, where it undergoes division into manageable segments. Each segment is then analysed by a large language model (LLM) to extract pertinent information in response to user queries. LangChain subsequently consolidates and refines these extracted responses to furnish precise and contextually relevant answers to the queries. This methodology guarantees the effective management and querying of extensive documents.

For PDF question and answering tasks, we can leverage libraries such as Embeddings and FAISS. Text embeddings transform raw text into vectors within a multi-dimensional space. Within this space, texts with similar semantics are positioned closer together, facilitating efficient similarity assessments. Notably, OpenAI's models excel in generating comprehensive embeddings that encapsulate the nuances of the text.

FAISS is employed to process the text, establishing a knowledge base for the function. When a user poses a question, FAISS executes a similarity search on the knowledge base to retrieve pertinent documents based on the user's query. This methodology ensures the extraction of relevant information for effective question answering.



```
[2] !pip install openai
[3] pip install -U langchain-openai
[4] !pip install -U langchain-community
[5] import openai
[6] from langchain_openai import OpenAIEmbeddings
[7] from langchain.embeddings import OpenAIEmbeddings
[8] !pip install langchain
    !pip install PyPDF2
    !pip install faiss-cpu
    !pip install tiktoken
```

Figure 2.10: Part of code for Querying of PDF

Firstly, we install all the required libraries and further import the necessary libraries from them.


```

✓ [9] from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS

✓ [10] import os
open_api_key="sk-proj-gzJGp4U3xf56w10cpUqRT3B1bkfJRm6m0F9lBgl1AGrJxzUw"
os.environ["OPENAI_API_KEY"]=open_api_key

✓ [11] pdfreader = PdfReader('/content/genai.pdf')

✓ [12] from typing_extensions import Concatenate
raw_text = ''
for i, page in enumerate(pdfreader.pages):
    content = page.extract_text()
    if content:
        raw_text += content

✓ [13] raw_text
15 ↗
'Generative AI\nStefan Feuerriegel1, and Jochen Hartmann2 and Christian Janiesch3 and Patrick Zschech4\n1LMU Munich & Munich Center for Machine Learning, Geschwister-Scholl-Platz 1, 80539 Munich, \nfeuerriegel@lmu.de\n2Technical University of Munich, TUM School of Management, Arcisstr. 21, 80333 Munich, \njochen.hartmann@tum.de\n3TU Dortmund University, Otto-Hahn-Str. 12, 44319 Dortmund, \nchristian.janiesch@tu-dortmund.de\n4FAU Erlangen-Nürnberg, Lange Gasse 20, 90403 Nürnberg, \npatrick.zschech@fau.de\nTo whom correspondence should be addressed; e-mail: feuerriegel@lmu.de.\n\nThe term "generative AI" refers to computational techniques that are capable of generating seemingly new, meaningful content such as text, images, or audio from training data. The widespread diffusion of this technology with examples such as Dall-E 2, GPT-4, and Copilot is currently revolutionizing the way we work and communicate with each other. In this article, we provide a conceptualization of generative AI as ...

```

Figure 2.11: Part of code for Querying of PDF

Then we set up the environment in which we use the API key which we generated. The API key will be used in the LLM models when they will be initialized. The PDF path is further provided and we try to read the PDF with the help of PDFReader. Now to read each and every page, we imported two more libraries, from `typing_extensions` we imported `Concatenate`. Then we enumerate inside the pages of the PDF and extract this text and put it inside the variable “content”. Then putting all the content inside `raw_text`. At last, we can see what the raw text looks like.

```

✓ [14] text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 800,
    chunk_overlap = 200,
    length_function = len,
)
texts = text_splitter.split_text(raw_text)

✓ [15] len(texts)
0s ↗
141

✓ [16] embeddings = OpenAIEmbeddings(openai_api_key = os.environ["OPENAI_API_KEY"])
0s ↗
Show hidden output

✓ [17] document_search = FAISS.from_texts(texts, embeddings)

✓ [18] document_search
0s ↗
<langchain_community.vectorstores.faiss.FAISS at 0x7a1947918bb0>

✓ [19] from langchain.chains.question_answering import load_qa_chain
from langchain.llms import OpenAI

```

Figure 2.12: Part of code for Querying of PDF

We split the text using `Character Text Splitter` such that it should not increase the token size. We go ahead and download the embeddings from OpenAI. Then we use

from `texts` from FAISS to put the text into the embeddings to create the vector `document_search`. Then `load_qa_chain` is imported.

```

06 [20] chain = load_qa_chain(OpenAI(), chain_type="stuff")
    Show hidden output

26 [22] query = "What is Generative Artificial Intelligence?"
    docs = document_search.similarity_search(query)
    chain.run(input_documents=docs, question=query)

    '\nGenerative AI refers to computational techniques that can create new and meaningful content, such as text, images, or audio, based on training data. It is currently being used in various applications, including art, question-answering systems, and support for knowledge workers. It is expected to have a significant impact on the global economy and could potentially replace many jobs in the future.'
```

```

16 [35] query = "Tell about the limitations of Generative AI in brief"
    docs = document_search.similarity_search(query)
    chain.run(input_documents=docs, question=query)

    ' The limitations of Generative AI include incorrect outputs, lack of theory of mind, and reliance on probabilistic algorithms which may lead to errors in the output. These limitations persist at the model level and may have implications for system and application levels.'
```

Figure 2.13: Part of code for Querying of PDF

We take `load_qa_chain` and use `OpenAI` as an object and set `chain_type = "stuff"` so that whenever a question is asked, it is able to give the answer. Then we ask different queries from the PDF and got the respective outputs.

```

query = "Explain the conclusion in detail"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)

'\n\nThe conclusion of the given context is that there are several important research directions that need to be explored in order to improve the usage and reliability of generative AI systems. These include using generative AI to generate explanations for business analytics models to make them more understandable for non-experts, developing solutions for detecting and mitigating hallucination in generative AI, and finding ways for users to verify and reduce the risk of incorrect outcomes. Additionally, there is a need for generative AI to support decision analytics and data science projects by bridging the gap between modeling experts and domain users. This conclusion highlights the potential of generative AI in various fields and the importance of addressing its limitations for its widespread and effective use.'
```

```

query = "What is color of sky?"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)

'I don't know, as the context provided does not mention the color of the sky.'
```

Figure 2.14: Part of code for Querying of PDF

CHAPTER 3

DISCUSSIONS AND RESULTS

In this paper, we presented our approach for summarizing and querying PDFs using Generative AI within the LangChain framework. The implementation of LangChain allowed for efficient querying and made it easy to extract specific information from the PDFs.

We employed the Stuff Documentation Chain method to summarize a single-page PDF, specifically a resume in PDF format.

output_summary

'Mayank Jain is a 22-year-old male with a passion for data analytics and a degree in mathematics. He is seeking opportunities to utilize his analytical, mathematical, and technical skills to solve real-life problems related to analyzing large datasets. Mayank has experience in programming languages such as Python and SQL, as well as data analysis tools like Pandas and Machine Learning. He has completed various academic projects in data science, including predicting housing prices in Mexico and customer segmentation in the US. Mayank has also held positions of responsibility in various organizations and has completed internships at Viral Fission, where he managed a team of campus ambassadors.'

Figure 3.1: Output for Stuff Document Chain method.

Through this approach, the summary effectively encapsulated the primary points and crucial details from the PDF document. The resulting summary is characterized by clarity and coherence.

To summarize a 4-page PDF, we applied the MapReduce method, which is suitable for handling large documents that exceed the token limit of a single prompt.

Summary

'Computational Thinking Week 1 focuses on flow charts, which are graphical representations of algorithms used by programmers for program planning. Flow charts use symbols to indicate the flow of information and processing, aiding in communication, program design, debugging, and documentation. However, they can be challenging for large and complex programs. The concept of data sanity is also discussed, including data types and subtypes like characters, integers, booleans, dates, marks, and names. Records, which are data types with multiple fields, are also explained with examples like marks cards, word lists, and shopping bills.'

Figure 3.2: Output for Map-Reduce method.

This method efficiently condensed the content of PDFs while retaining crucial details. The resulting summary is well-structured, ensuring clarity and providing a comprehensive overview of the document's content.

We conducted PDF querying using LangChain on a research paper in PDF format, aiming to evaluate the effectiveness and accuracy of generative AI in extracting relevant information from PDFs. Through the querying process, we extracted relevant information from the documents. The outcomes of the PDF querying are illustrated below.

```
query = "what is Generative Artificial Intelligence?"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)

'Generative AI refers to computational techniques that can create new and meaningful content, such as text, images, or audio, based on training data. It is currently being used in various applications, including art, question-answering systems, and support for knowledge workers. It is expected to have a significant impact on the global economy and could potentially replace many jobs in the future.'
```

```
query = "Tell about the limitations of Generative AI in brief"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)

'The limitations of Generative AI include incorrect outputs, lack of theory of mind, and reliance on probabilistic algorithms which may lead to errors in the output. These limitations persist at the model level and may have implications for system and application levels.'
```

Figure 3.3: Output for querying of PDF

```
query = "Explain the conclusion in detail"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)

'\n\nThe conclusion of the given context is that there are several important research directions that need to be explored in order to improve the usage and reliability of generative AI systems. These include using generative AI to generate explanations for business analytics models to make them more understandable for non-experts, developing solutions for detecting and mitigating hallucination in generative AI, and finding ways for users to verify and reduce the risk of incorrect outcomes. Additionally, there is a need for generative AI to support decision analytics and data science projects by bridging the gap between modeling experts and domain users. This conclusion highlights the potential of generative AI in various fields and the importance of addressing its limitations for its widespread and effective use.'
```

```
query = "What is color of sky?"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)

'I don't know, as the context provided does not mention the color of the sky.'
```

Figure 3.3: Output for querying of PDF

A coherent and logical progression was sustained across the generated responses, ensuring ample information to comprehensively address the queries. The answers exhibited a high degree of relevance, directly addressing the questions posed. We experimented with various query strategies, culminating in a question unrelated to the PDF content, resulting in the expected failure of the model to provide an answer.

CHAPTER 4

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

4.1 Conclusion

Generative AI is like a creative powerhouse, reshaping how we think about artificial intelligence. It's not just about processing data anymore; it's about generating entirely new content based on what it's learned. This technology has completely revolutionized the AI landscape, unleashing a wave of innovation and creativity that knows no bounds. From transforming how we create digital content to enhancing data analysis, the applications of generative AI are vast and promising. But as with any powerful tool, there are challenges to overcome. Ethical concerns, like misuse, and the importance of having solid training data, remind us of the responsibility we have in developing and deploying generative AI technologies responsibly.

In our research, we focused on how generative AI can be applied to summarizing and querying PDF documents. Using LangChain, a framework designed to harness generative AI models, we put two methods to the test: the Stuff Documentation Chain method and the MapReduce method. The results were exciting. Both methods proved to be effective in condensing complex PDFs into clear, concise summaries. They managed to capture the essence of the documents, making them valuable tools for understanding and analysis. In the querying experiments, LangChain once again proved its capabilities. It effortlessly extracted relevant information from PDFs in response to specific queries. The accuracy and consistency of the outputs showcased the potential of generative AI in automating information extraction.

These findings underscore the importance of leveraging generative AI in tasks like summarization and querying. They have broad implications across various domains, including education, research, and information retrieval systems. Generative AI isn't just a tool, it's a game-changer that's shaping the future of how we interact with digital content.

4.2 Future Scope

In the near future, numerous exciting opportunities will arise to enhance the capabilities of generative AI in summarizing and querying PDFs. One crucial focus area involves refining the accuracy and contextual understanding of AI models. This may entail improving training data quality and developing advanced algorithms to grasp language nuances and document contexts better, thereby enabling the AI to deliver more precise and relevant outputs.

Another promising direction is the integration of these summarization and querying tools with other data sources, such as databases or live feeds. This integration would provide users with a deeper understanding of their queries and facilitate the development of a more robust information retrieval system.

The combination of these capabilities within automated workflows could significantly improve processes across various industries, ranging from legal and healthcare to education and finance. This enhancement would lead to increased productivity and more informed decision-making as large document volumes are analyzed more efficiently and expediently.

Additionally, introducing real-time querying and summary functionality presents an exciting opportunity. This feature would offer instant insights and information retrieval from live document streams, enhancing the tool's flexibility and responsiveness.

Generative AI holds immense potential across diverse domains. Advancements in model architectures, training methods, and data augmentation are poised to render generative models more sophisticated and versatile in the future. These advancements will contribute significantly to applications such as content creation, virtual assistants, and personalized recommendation systems. Moreover, responsible and ethical deployment of generative AI technologies necessitates addressing challenges related to interpretability, fairness, and ethical considerations.

As the field of generative AI continues to progress, future research efforts may focus on refining existing methodologies, exploring novel approaches, and addressing inherent challenges to further enhance the capabilities of generative AI in processing and analyzing data. It is incumbent upon us to foster its development while upholding ethical principles. Through sustained innovation and interdisciplinary collaboration, generative AI is poised to revolutionize how we interact with and derive insights from data.

4.3 Limitations

Several limitations must be considered in the realm of Generative AI, particularly concerning the processes of summarizing and querying PDF documents using generative AI and LangChain. Firstly, the computational demands of generative AI models pose challenges for entities with limited resources. LangChain's efficacy heavily relies on the quality and diversity of its training data, and inadequate or biased data can lead to inaccurate and unreliable summaries and query responses. Moreover, LangChain may encounter token limitations common to many AI models, restricting input text length and output responses, making it challenging to handle lengthy documents or complex queries. Abrupt cessation of output due to incomplete processing or token limit issues is a possibility. Furthermore, LangChain's practicality and sustainability in real-world applications may be hindered by maintenance, update, and integration challenges.

Additionally, ethical concerns arise regarding the generation of content, including potential misuse such as deepfakes and fake news. To mitigate misuse, it is imperative to implement ethical guidelines, responsible development practices, and appropriate techniques.

Addressing these limitations is crucial for researchers and practitioners to effectively utilize LangChain for PDF summarization and querying, fostering future advancements and developments.

4.4 Social Impact

The generative AI generates a profound social impact by democratising the access to information as well as enhancing productivity in a variety of fields. In particular, generative AI can improve summarization and querying of PDF documents using LangChain, generating a significant social impact across a wide spectrum of fields. Researchers, educators, professionals, and individuals seeking information benefit greatly from automation of these tasks. As generative AI streamlines document analysis, it fosters knowledge discovery and dissemination, ultimately improving collaboration and fostering innovation.

There are however ethical concerns relating to its use, such as the potential for creating deepfakes and spreading misinformation, which require an approach that is responsible when it comes to its design and deployment.

In general, generative AI is having a far-reaching social impact and its applications are being applied in summarization and querying of PDF documents using LangChain. In addition to increasing productivity, it promotes digital inclusivity and fosters ethical practices, ultimately contributing to a more informed, accessible, and equitable society. By recognizing and addressing societal needs and ethical considerations, generative AI can be leveraged as powerful tool positively.

REFERENCES

- [1] Mahadevan, R., & Raman, R. C. (2023). Comparative Study and Framework for Automated Summarizer Evaluation: LangChain and Hybrid Algorithms. *arXiv preprint arXiv:2310.02759*.
- [2] Ooi, K. B., Tan, G. W. H., Al-Emran, M., Al-Sharafi, M. A., Capatina, A., Chakraborty, A., ... & Wong, L. W. (2023). The potential of generative artificial intelligence across disciplines: Perspectives and future directions. *Journal of Computer Information Systems*, 1-32.
- [3] Pesaru, A., Gill, T. S., & Tangella, A. R. (2023). AI assistant for document management Using Lang Chain and Pinecone. *International Research Journal of Modernization in Engineering Technology and Science*.
- [4] Ramdurai, B., & Adhithya, P. (2023). The impact, advancements and applications of generative AI. *International Journal of Computer Science and Engineering*, 10(6), 1-8.
- [5] Topsakal, O., & Akinici, T. C. (2023, July). Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences* (Vol. 1, No. 1, pp. 1050-1056).



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi – 110042, India

PLAGIARISM VERIFICATION

Title of the dissertation “Summarization and Querying of PDF using Generative Artificial Intelligence”.

Total Pages: 32

Name of Students: Mihika Jain (2K22/MSCMAT/24) and

Somya Rao (2K22/MSCMAT/44)

Name of Supervisor: Prof. Anjana Gupta

Department: Department of Applied Mathematics

This is to report that the above dissertation report was scanned for similarity detection. Process and outcome is given below:

Software Used: Turnitin , Similarity Index: 11% , Total Word Count: 7020 words

Date: 29 May, 2024

Candidate’s Signature

Signature of Supervisor

Similarity Report

PAPER NAME

Dissertation Final Mihika Somya.pdf

AUTHOR

Mihika somya

WORD COUNT

7020 Words

CHARACTER COUNT

40494 Characters

PAGE COUNT

32 Pages

FILE SIZE

1.1MB

SUBMISSION DATE

May 29, 2024 12:31 PM GMT+5:30

REPORT DATE

May 29, 2024 12:31 PM GMT+5:30**● 11% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 9% Internet database
- 2% Publications database
- Crossref database
- Crossref Posted Content database
- 7% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less than 10 words)

Similarity Report

● 11% Overall Similarity

Top sources found in the following databases:

- 9% Internet database
- 2% Publications database
- Crossref database
- Crossref Posted Content database
- 7% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Delhi Technological University on 2024-05-26 Submitted works	3%
2	v7labs.com Internet	2%
3	ibm.com Internet	<1%
4	dspace.dtu.ac.in:8080 Internet	<1%
5	listens.online Internet	<1%
6	Chemeketa Community College on 2024-01-28 Submitted works	<1%
7	as-proceeding.com Internet	<1%
8	export.arxiv.org Internet	<1%

Similarity Report

9	internationaljournalsrg.org Internet	<1%
10	University of Greenwich on 2021-03-28 Submitted works	<1%
11	Berlin School of Business and Innovation on 2024-01-30 Submitted works	<1%
12	Delhi Technological University on 2024-05-28 Submitted works	<1%
13	probius.bio Internet	<1%
14	Rushmore Business School on 2024-05-14 Submitted works	<1%
15	ibi.au.edu.tw Internet	<1%
16	University of Wales Institute, Cardiff on 2024-05-15 Submitted works	<1%
17	nith on 2024-05-25 Submitted works	<1%
18	businesstoday.in Internet	<1%
19	Singapore Polytechnic on 2024-02-02 Submitted works	<1%
20	University of Exeter on 2024-05-01 Submitted works	<1%

Similarity Report

21	technodocbox.com Internet	<1%
22	American University in Bulgaria on 2023-10-16 Submitted works	<1%
23	University Politehnica of Bucharest on 2023-09-28 Submitted works	<1%
24	theknowledgeacademy.com Internet	<1%



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultapur, Main Bawana Road, Delhi – 110042, India

CERTIFICATE OF DISSERTATION SUBMISSION FOR
EVALUATION

1. Name: Mihika Jain and Somya Rao
2. Roll No. : 2K22/MSCMAT/24 and 2K22/MSCMAT/44
3. Dissertation title: Summarization and Querying of PDF using Generative Artificial Intelligence
4. Degree for which the Dissertation is submitted: M.Sc. Mathematics
5. Faculty of the University to which the Dissertation is submitted: Prof. Anjana Gupta
6. Thesis Preparation Guide was referred to for preparing the Dissertation.
YES NO
7. Specifications regarding Dissertation format have been closely followed.
YES NO
8. The contents of the Dissertation have been organized based on the guidelines.
YES NO
9. The Dissertation has been prepared without resorting to plagiarism. YES NO
10. All sources used have been cited appropriately. YES NO
11. The Dissertation has not been submitted elsewhere for a degree. YES NO
12. Submitted 2 spiral bound copies plus one CD. YES NO

(Signature of Candidates)

Name(s): Mihika Jain and Somya Rao

Roll No: 2K22/MSCMAT/24 and 2K22/MSCMAT/44



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultapur, Main Bawana Road, Delhi – 110042, India

CERTIFICATE OF DISSERTATION SUBMISSION

1. Name: Mihika Jain and Somya Rao
2. Roll No. : 2K22/MSCMAT/24 and 2K22/MSCMAT/44
3. Dissertation title: Summarization and Querying of PDF using Generative Artificial Intelligence
4. Degree for which the Dissertation is submitted: M.Sc. Mathematics
5. Faculty of the University to which the Dissertation is submitted: Prof. Anjana Gupta
6. Thesis Preparation Guide was referred to for preparing the Dissertation.
YES NO
7. Specifications regarding Dissertation format have been closely followed.
YES NO
8. The contents of the Dissertation have been organized based on the guidelines.
YES NO
9. The Dissertation has been prepared without resorting to plagiarism. YES NO
10. All sources used have been cited appropriately. YES NO
11. The Dissertation has not been submitted elsewhere for a degree. YES NO
12. All the corrections has been incorporated. YES NO
13. Submitted 2 spiral bound copies plus one CD. YES NO

(Signature of Supervisor)

Name : Prof. Anjana Gupta

(Signature of Candidates)

Name : Mihika Jain

Somya Rao

Roll No. : 2K22/MSCMAT/24

2K22/MSCMAT/44