

# **VERBAL DOCUMENT ENGAGEMENT SYSTEM**

Project report submitted in partial fulfillment of the requirements for the award of the degree of

## **Master of Computer Applications**

Submitted by

**DEEPHAA P**

**(2238M0152)**

Under the Supervision and Guidance of

**DR.C. SATHYA B.Tech(IT)., M.sc (IT)., M.Phil., Ph.D.**

**ASSISTANT PROFESSOR**



**DEPARTMENT OF COMPUTER SCIENCE**

**KGiSL INSTITUTE OF INFORMATION MANAGEMENT**

Saravanampatti, Coimbatore – 641 035.

**MARCH 2024**

## **DECLARATION**

I hereby declare that this project work titled “**VERBAL DOCUMENT ENGAGEMENT SYSTEM**”, submitted to the Department of Computer Science, KGiSL Institute of Information Management, Saravanampatti, Coimbatore is a record of original work done by me under the supervision and guidance of **Dr. C. Sathya B.Tech(IT)., M.sc(IT)., M.Phil., Ph.D.** and that this project work has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or similar title to any candidate of any University.

**PLACE:**

**DATE:**

---

**(DEEPHAA P)**

## **CERTIFICATE**

This is to certify that the project work titled **VERBAL DOCUMENT ENGAGEMENT SYSTEM** submitted to Bharathiar University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a record of original work done by **DEEPHAA P, 2238M0152** under my supervision and guidance and that this project work has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or similar title to any candidate of any University.

**SIGNATURE OF THE GUIDE**

**SIGNATURE OF THE HOD**

(College Seal)

Submitted for the Viva-Voce Examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

I would like to take this opportunity to express my deep sense of gratitude to all who helped me directly or indirectly during this project work.

I express my sincere thanks to **Dr. Ashok Bakthavathsalam, B.E, M.S., Ph.D., Managing Director**, for giving me an opportunity to do this course of study and to undertake this project work.

I take this opportunity to convey my sincere thanks to **Dr. P. Krishna Priya MCA., M.Phil., Ph.D., Director**, KGiSL Institute of Information Management for her moral support throughout the course.

I have great pleasure in acknowledging my thanks to **Mr. Alwin Pinakas James M.Sc., M.Phil., (Ph.D), Head, Department of Computer Science**, KGiSL Institute of Information Management for his encouragement and help throughout the course.

I would like to thank my supervisor, **Dr.C.Sathya B.Tech(IT), M.Sc(IT), M.Phil., Ph.D. Assistant Professor**, Department of Computer Science, KGiSL Institute of Information Management for being a great mentor and the best adviser I could ever have. Her advise, encouragement and critics are source of innovative ideas, inspiration and causes behind the successful completion of this dissertation.

Last but not the least, I express my thanks to my parents and friends who have kindly provided necessary support for successful completion of the project.

-DEEPHAA P

# CONTENTS

<b>ABSTRACT</b>	<b>i</b>
<b>LIST OF FIGURES</b>	<b>ii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>iii</b>
<b>CHAPTER</b>	
<b>I</b>	<b>INTRODUCTION</b>
	<b>01</b>
1.1	General introduction
	<b>01</b>
1.2	Importance of the study
	<b>01</b>
1.3	Problem statement
	<b>02</b>
1.4	Aim & Objectives
	<b>03</b>
1.5	Scope and Limitation of the study
	<b>03</b>
1.6	contribution of the study
	<b>04</b>
1.7	Outline of report
	<b>04</b>
<b>II</b>	<b>LITERATURE REVIEW</b>
	<b>05</b>
<b>III</b>	<b>METHODOLOGY</b>
	<b>07</b>
3.1	Existing system
	<b>07</b>
3.2	Proposed system
	<b>07</b>
3.2.1	Modules of the Proposed system
	<b>08</b>
3.2.2	Programming environment
	<b>11</b>
<b>IV</b>	<b>WORKING OF SYSTEM</b>
	<b>14</b>
4.1	System Architecture
	<b>14</b>
4.2	Algorithms
	<b>16</b>
4.2.1	OCR Algorithm
	<b>16</b>
4.2.2	cosine similarity Algorithm
	<b>17</b>
4.2.3	Text splitting Algorithm
	<b>18</b>
<b>V</b>	<b>EXPERIMENTAL ANALYSIS</b>
	<b>19</b>
5.1	System Configuration
	<b>19</b>
5.2	Sample Code
	<b>20</b>
5.3	Performance Analysis
	<b>23</b>

	5.4	Performance Measures	24
	5.5	Results and Discussions	25
<b>VI</b>		<b>CONCLUSION &amp; RECOMMENDATIONS</b>	<b>27</b>
	6.1	Conclusion	27
	6.2	Recommendations for Future Work	27
<b>REFERENCES</b>			
<b>APPENDICES</b>			

## **ABSTRACT**

The verbal document engagement system uses Artificial Intelligence and Natural Language Processing to make document interaction more user-friendly and efficient. Its goal is to allow users to converse with their documents in natural language , as if they were conversing with humans. The existing closed domain Question Answering system refers to a type of system that can answer questions within a specific domain, such as medicine or law with high accuracy. While closed domain QA systems have many advantages including improved accuracy and efficiency, they also have several disadvantages, including limited scope,maintenance and updating and lack of context.

The proposed system is ideal for swiftly extracting information or responding to inquiries from huge PDF files such as manuals , essays, legal contracts, books or research papers. It analyzes the content to construct a semantic index of each paragraph, and when you ask a question, the AI will respond by relevant paragraphs. The queries will be answered based on the chat history. It can enhance learning experience by effortlessly comprehending textbooks, handouts and presentations. It can efficiently analyze documents for work purposes.

.

## LIST OF FIGURES

S No.	Figure Description	Page No.
3.1	File Loader Module	---
3.2	Embeddings	---
4.1	System Architecture	---



## **LIST OF ABBREVIATIONS**

ML	Machine Learning
AI	Artificial Intelligence
EM	Exact Match
QA	Question Answering
LLM	Large Language Model
NLP	Natural Language Processing
SAS	Semantic Answer Similarity

# CHAPTER I

## INTRODUCTION

### 1.1 GENERAL INTRODUCTION

Conversational Document leverages the power of Artificial Intelligence (AI) and Natural Language Processing (NLP) to enhance the user experience and efficiency when interacting with documents. Its primary objective is to enable users to engage with their documents using natural language, as if they were having a conversation with a human.

The main purpose of Conversational Document is to facilitate quick and effective information extraction and response to inquiries from large PDF files. It is particularly useful for dealing with extensive documents such as manuals, essays, legal contracts, books, or research papers. To achieve its functionality, Conversational Document performs a comprehensive analysis of the document's content. It constructs a semantic index for each paragraph, capturing the meaning and context of the information. When a user poses a question, the AI system retrieves and analyzes relevant paragraphs from the document's semantic index. Based on this analysis, the AI generates a response that directly addresses the user's query.

By employing AI and NLP techniques, Conversational Document streamlines the process of interacting with documents, making it more user-friendly and efficient. It eliminates the need for manual searching and browsing through lengthy documents, enabling users to access the information they need swiftly and accurately.

### 1.2 IMPORTANCE OF STUDY

The importance of a verbal engagement system lies in its ability to efficiently retrieve relevant information from vast collections of documents. Here are some key reasons for its importance:

**Information Retrieval:** In today's digital age, organizations and individuals have access to vast amounts of information stored in documents such as reports, articles, emails, and research papers. A document query system enables users to quickly and accurately retrieve the specific information they need from this vast sea of documents.

**Knowledge Discovery:** Document querying facilitates knowledge discovery by enabling users to explore and analyze information within documents. By retrieving relevant documents based on user queries, document query systems help users discover new insights, trends, and relationships within the data.

**Research and Innovation:** Researchers and innovators rely on access to relevant literature and research findings to advance their work. Document query systems provide researchers with the ability to efficiently search and access relevant research papers, articles, and documents, thereby accelerating the pace of discovery and innovation.

**Efficiency and Productivity:** Document querying improves efficiency and productivity by reducing the time and effort required to find information. Instead of manually searching through large volumes of documents, users can use document query systems to quickly locate the information they need, saving time and increasing productivity.

### 1.3 PROBLEM STATEMENT

Professionals and students often encounter significant challenges when dealing with a large volume of documents, such as legal contracts, research papers, technical manuals, and textbooks. The manual process of searching for specific information or answers within these documents can be extremely time-consuming, inefficient, and prone to errors.

For professionals, the need to extract relevant information from numerous documents to address inquiries or make informed decisions can be overwhelming. The time and effort required to sift through lengthy papers not only leads to frustration but also causes delays and increases the likelihood of making mistakes. This ultimately impacts productivity and hampers their ability to provide accurate and timely responses. Similarly, students face similar difficulties when trying to find solutions in textbooks or study materials. The task of locating specific information within lengthy chapters can be daunting and time-consuming. It often necessitates extensive reading, which can hinder comprehension and impede the ability to locate the required answers efficiently.

Moreover, teachers invest a significant amount of time and effort into developing question papers for class assessments. Creating well-structured questions that cover essential topics and align with the curriculum can be a time-consuming and repetitive task. Overall, the need to efficiently extract information or answers from a vast number of documents poses a common challenge for professionals, students, and teachers alike. This process is frequently laborious,

time-consuming, and error-prone, resulting in decreased productivity, increased frustration, and delays in obtaining the necessary information.

## **1.4 AIM & OBJECTIVES**

To enable efficient information extraction from large documents such as manuals, essays, legal contracts, books, and research papers. By analyzing the content and constructing a semantic index of each paragraph, the model can quickly retrieve pertinent information in response to user inquiries.

To Enhance educational experiences by simplifying the understanding of presentations, handouts, and textbooks. By extracting key information and providing explanations, it aims to make learning materials more accessible and comprehensible for students.

To Support multiple file formats and languages the AI model aims to support a wide range of file formats, including PDF, Word, and PPTX. It also aims to provide language support for multiple languages, allowing users to interact with the system in their preferred language.

To Analyze business documents providing insights and information while ensuring data privacy. By utilizing secure cloud storage, it aims to protect sensitive business data and maintain confidentiality. And then extract insights from literary works and historical documents. This objective aims to enable users to gain a deeper understanding of cultural and historical texts.

## **1.5 SCOPE AND LIMITATION OF THE STUDY**

The scope of the project encompasses the development of a robust and scalable system capable of efficiently retrieving relevant information from large collections of documents. Key features include semantic indexing of documents using advanced natural language processing (NLP) techniques, query understanding mechanisms to infer user intent and context, and dynamic ranking algorithms for prioritizing search results. The system aims to empower users across diverse domains with a powerful tool for information discovery, decision-making, and knowledge extraction. However, limitations may arise due to challenges such as handling unstructured or poorly formatted documents, language barriers, and the need for continuous improvement in accuracy and relevance of search results. Additionally, scalability issues may arise when dealing with extremely large document collections, and the system's effectiveness may vary depending on

the quality and quantity of available data. Therefore, ongoing evaluation, optimization, and refinement are essential to address these limitations and enhance the system's performance over time.

## **1.6 CONTRIBUTION OF THE STUDY**

The contribution of this project lies in its provision of an innovative and efficient solution for accessing and extracting valuable information from extensive document repositories. By leveraging advanced natural language processing (NLP) techniques, dynamic ranking algorithms, and semantic indexing, the system enhances information retrieval accuracy and relevance, thereby facilitating informed decision-making, knowledge discovery, and research advancement across various domains. Its user-centric design and customizable features empower individuals and organizations to streamline their workflow, improve productivity, and gain insights from diverse sources of information. Overall, this project's contribution lies in its ability to unlock the full potential of document querying, enabling users to harness the wealth of knowledge available in digital documents effectively.

## **1.7 OUTLINE OF REPORT**

The current chapter provides an overview of the significance and challenges associated with conventional document query systems, laying out the aims and objectives of the project. Chapter 2 delves into the literature review, examining traditional document querying methods and introducing relevant studies on advanced techniques such as natural language processing and machine learning for document retrieval. Chapter 3 outlines the methodology, detailing the proposed framework for the document query system, the setup of the environment for training, and the training process itself. The findings and discussions are presented in Chapter 4, while Chapter 5 concludes the outcomes of the document query system developed and offers recommendations for enhancing the methodology and future research directions. Lastly, Chapter 6 delves into a detailed discussion of the obtained results, their implications, the study's limitations, and provides conclusions and recommendations for further research and implementation.

## CHAPTER II

### LITERATURE REVIEW

Several deep learning models have been proposed for question answering. However, due to their single-pass nature, they have no way to recover from local maxima corresponding to incorrect answers. For example, Caiming Xiong, Victor Zhong, Richard Socher developed Dynamic Coattention Networks For Question Answering for question answering. The DCN first fuses co-dependent representations of the question and the document in order to focus on relevant parts of both. Then a dynamic pointing decoder iterates over potential answer spans. This iterative procedure enables the model to recover from initial local maxima corresponding to incorrect answers. On the Stanford question answering dataset, a single DCN model improves the previous state of the art from 71.0% F1 to 75.9%, while a DCN ensemble obtains 80.4% F1.

[1]”QUESTION ANSWERING CHATBOT USING DEEP LEARNING WITH NLP” Devanshi Singh, K. Rebecca Suraksha, *et al.*, 2021. This paper discusses developing a closed domain, factoid question answering chatbot that uses deep learning and NLP techniques. The system consists of three main components: a question processing module, an answer processing module, and a knowledge base.

[2]”QUESTION ANSWERING SYSTEM ON EDUCATION ACTS USING NLP TECHNIQUES” Sweta P. Lende and Dr.M.M. Raghuwanshi, 2020. This paper proposes a closed domain Question Answering (QA) system that uses Natural Language Processing (NLP) techniques to provide accurate answers to queries related to education acts. The authors conducted a literature survey of existing QA systems and found that closed domain QA systems are more accurate than open domain QA systems.

[3]”A BRIEF SURVEY OF WORD EMBEDDING AND ITS RECENT DEVELOPMENT” Qilu Jiao, and Shunyao Zhang, 2021. This paper provides an overview of the recent developments in learning effective representations of text words. It begins by introducing the concept of word embedding and its importance in natural language processing and other machine learning tasks.

[4]”AUTOMATED QUESTION GENERATOR SYSTEM USING NLP LIBRARIES” Priti Gumaste, Shreya Joshi, *et al* 2020. The paper discusses various NLP libraries, such as Spacy and NLTK, to perform essential text processing tasks like tokenization, stemming, lemmatization, and punctuation handling.

[5]” AUTOMATIC GENERATION OF QUESTION PAPER USING NLP TECHNIQUE” Shivangi Daware, Rachana Ankar, *et al* 2019. The paper aims to revolutionize the process of creating question papers by introducing organization and streamlining. By utilizing modern evolutionary algorithms and randomization algorithms, the system effectively manages multiple constraints and generates question papers from a comprehensive question bank database

[6]”A DEEP LEARNING MODEL BASED ON BERT AND SENTENCE TRANSFORMER FOR SEMANTIC KEYPHRASE EXTRACTION ON BIG SOCIAL DATA” R. Devika , Subramaniasamy Vairavasundaram, *et al* 2021. This paper discusses a novel approach for keyphrase extraction from Twitter content using a deep learning model called Semkey-BERT. The authors note that keyphrase extraction is an important task for analyzing large amounts of text data, and that social media platforms like Twitter present unique challenges due to the brevity and informal nature of user-generated content.

[7]”CONTEXTUALIZED EMBEDDINGS BASED TRANSFORMER MODELING FOR SENTENCE SIMILARITY” Md Tahmid Rahman Laskar, Jimmy Huang, *et al* 2020. This paper discusses the limitations of traditional word embeddings, which do not capture the context-dependent meaning of words. Then it introduces contextualized embeddings, which are able to capture the meaning of words in context by taking into account the surrounding words and their relationships.

[8]”DOC FORMER: END-TO-END TRANSFORMER FOR DOCUMENT UNDERSTANDING” Srikar Appalaraju, Bhavan Jasani, *et al* 2021. The paper discusses DocFormer, an end-to-end transformer-based model for visual document understanding (VDU). DocFormer combines text, vision, and spatial features to understand documents and performs well on various VDU tasks, including document classification, information extraction, and question answering.

## **CHAPTER III**

### **METHODOLOGY**

#### **3.1 EXISTING SYSTEM**

- Users would manually open PDF documents in PDF reader software and use the built-in search functionality to find specific words or phrases within the document
- creating document indexes or catalogs that provided metadata and keywords for PDF documents.
- some tools allowed users to extract text from PDF documents, converting them into plain text or other formats that could be searched using text
- Physical document collections were constrained by physical storage space, while digital document repositories were often inaccessible or fragmented across different systems or locations.
- In the absence of automated document query systems, users often had to rely on manual indexing methods to organize and categorize documents. Information retrieval typically involved browsing through physical or digital document collections using manual indexing systems such as file cabinets, folders, or indexes

#### **3.2 PROPOSED SYSTEM**

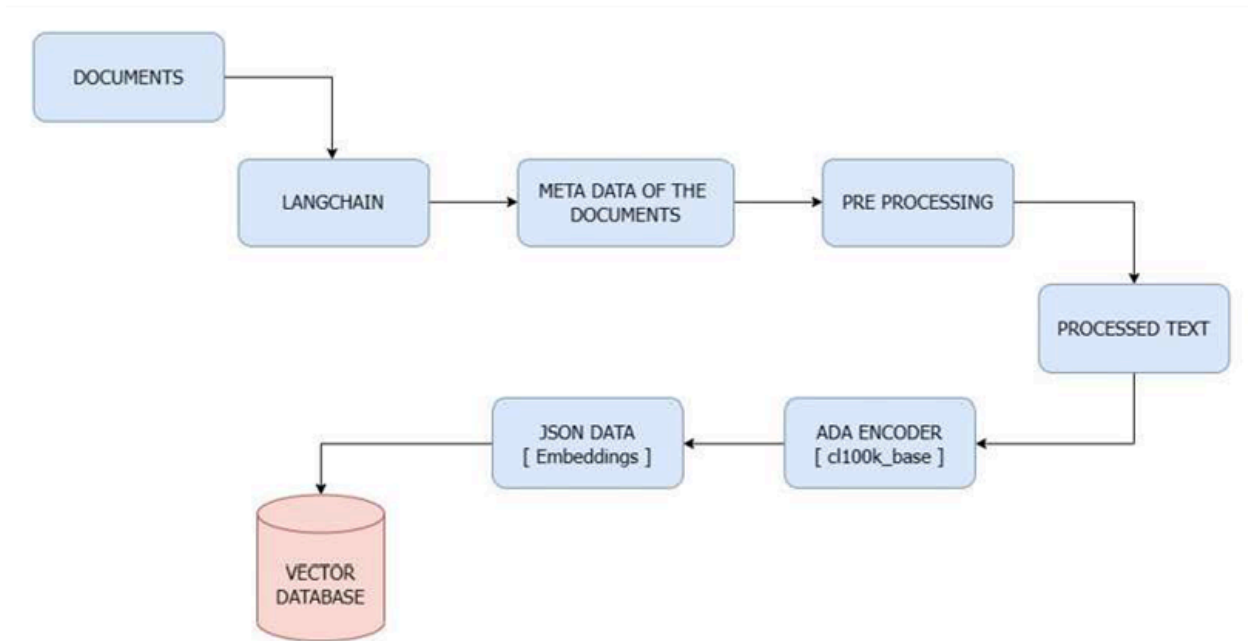
- Verbal document engagement system solve several problems related to accessing, searching, and extracting information from PDF documents.
- enable efficient information retrieval, content analysis, document organization, cross-document analysis, data extraction, compliance management, and enhanced productivity and collaboration
- users to perform comprehensive analysis across multiple PDF documents simultaneously
- Users can engage in natural language conversations with the system to clarify, summarize, or extract information from documents



## 3.2.1 MODULES OF THE PROPOSED SYSTEM

### 3.2.1.1 FILE LOADER

The File Loader module in the LangChain framework extracts and preprocesses text, images, and tables from various file formats. It uses the ADA text encoder to generate embeddings and stores them in vector database, facilitating efficient information retrieval from input files.



**Figure 3.1: File Loader Module**

### 3.2.1.2 TEXT EXTRACTION

The process of text extraction plays a vital role in the LangChain framework's file loader module. It is responsible for extracting not only the textual content but also images and tables from various file formats such as PDF, Word, and PPTX. This step is crucial because it allows the system to gain access to the raw content of the files, enabling further analysis and processing. By extracting the text, LangChain ensures that the framework can effectively work with the textual data present in the input files.

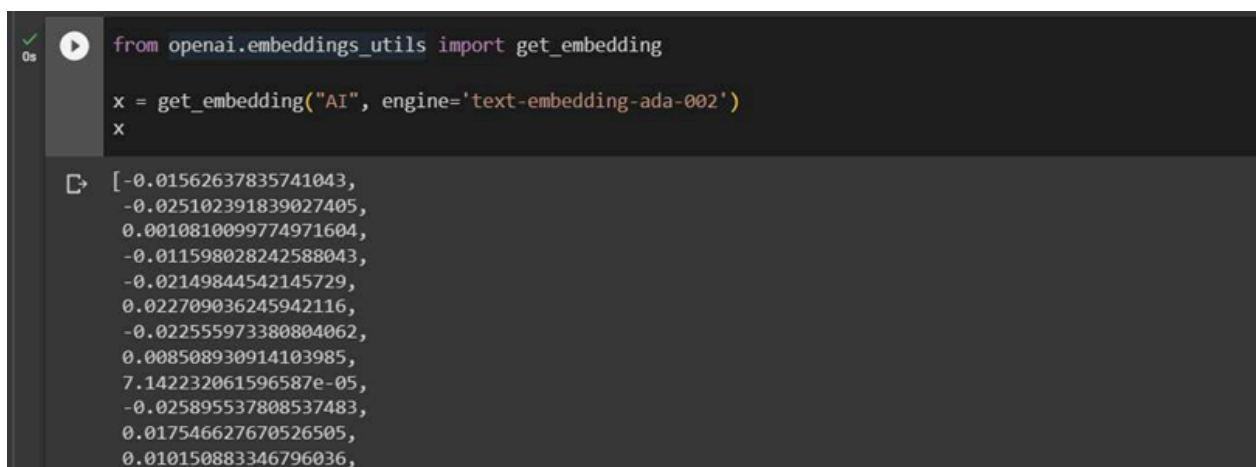
During the text extraction phase, the file loader module employs sophisticated techniques to identify and extract the relevant textual information accurately. It handles the complexities of

different file formats, parsing the content and separating it into meaningful units such as paragraphs, headings, or sections. This process ensures that the extracted text retains its structural integrity, allowing for a more comprehensive analysis and understanding.

### 3.2.1.3 EMBEDDING

Once the text has been extracted and processed, it undergoes the embedding step in the LangChain framework. The ADA text encoder, a specialized language model designed specifically for encoding textual data, takes the processed text as input and generates embeddings. These embeddings are numerical representations of the semantic meaning and contextual information captured within the text.

The ADA text encoder utilizes advanced natural language processing techniques to convert the textual data into a numerical format that can be easily understood and analyzed by machine learning algorithms. By transforming the text into embeddings, LangChain enables efficient and effective manipulation of the textual content, allowing for tasks such as semantic search, question answering, and summarization.



```
from openai.embeddings_utils import get_embedding

x = get_embedding("AI", engine='text-embedding-ada-002')
x
```

```
[ -0.01562637835741043,
  -0.025102391839027405,
   0.0010810099774971604,
  -0.011598028242588043,
  -0.02149844542145729,
   0.022709036245942116,
  -0.022555973380804062,
   0.008508930914103985,
   7.142232061596587e-05,
  -0.025895537808537483,
   0.017546627670526505,
   0.010150883346796036,
```

Figure 3.2: Embeddings

#### **3.2.1.4 QUERY MODULE**

The query module in the Conversational Document system utilizes the ADA encoder to generate embeddings for user queries. A semantic search is performed to find closely matched embeddings and retrieve related documents. The standalone question and retrieved documents are then passed through a fine-tuned LLM, enabling the chatbot to generate accurate and relevant responses by considering the query history and contextual information from the documents.

##### **3.2.1.4.1 Creating Standalone Question**

In the Conversational Document system, the chatbot interface allows users to make queries about the uploaded documents. Each new query is related to the previous queries and forms a standalone question, considering the query history. This approach ensures that the chatbot understands the context and can provide more relevant responses.

The standalone question is formulated based on the user's input and is passed through the ADA encoder. The ADA encoder utilizes the CLOCK base, which is a language model specifically designed for encoding textual data. By processing the standalone question through the ADA encoder, embeddings are generated that capture the semantic meaning and context of the question.

The ADA encoder plays a crucial role in understanding the user's query and representing it in a numerical format that can be further processed. The generation of embeddings enables efficient matching and retrieval of relevant information from the vector database, facilitating accurate responses to user queries.

##### **3.2.1.4.2 Semantic Search**

Once the embeddings for the standalone question are generated, the system performs a semantic search to find closely matched embeddings in the vector database. The goal of semantic search is to identify embeddings that are similar to the standalone question's embeddings, indicating related documents that may contain relevant information.

By contrasting the embeddings of the standalone question with the embeddings stored in the vector database, the system identifies the documents that have similar semantic meaning or

context. This semantic search process allows for efficient retrieval of closely related documents, narrowing down the scope of information to be considered for generating a response.

The semantic search capability enhances the accuracy and relevance of the system's responses by retrieving documents that are highly likely to contain the information sought by the user. It improves the overall user experience by providing targeted and precise information.

#### **3.2.1.4.3 Fine - Tuned Model**

After retrieving the closely matched embeddings and corresponding related documents, the system further processes the standalone question and the retrieved documents using a fine-tuned LLM. In this case, the text Davinci-003 model, a member of the GPT-3 model family, is utilized.

The LLM has undergone fine-tuning, which involves training the model on specific prompts to provide more accurate replies. By passing the standalone question and the retrieved documents through the fine-tuned LLM, the system leverages the model's advanced language understanding capabilities to generate a response that is relevant to the query.

### **3.2.2 PROGRAMMING ENVIRONMENT**

#### **3.2.2.1 LANGCHAIN**

LangChain is an advanced framework that aims to simplify the development of applications utilizing large language models (LLMs). By offering modular abstractions and a collection of implementations, LangChain provides developers with the necessary components to effectively work with language models. The framework ensures that these components can be seamlessly integrated, even if the developer chooses not to use the entire LangChain framework.

One of the notable features of LangChain is its provision of use-case specific chains. These chains are pre-configured arrangements of components tailored to specific application scenarios, allowing developers to quickly start working on their desired use cases. These chains are highly customizable, enabling developers to fine-tune them to best suit their specific requirements. This higher-level interface simplifies the initial setup process, enabling developers to get up and running swiftly. LangChain sets itself apart by advocating for applications that go

beyond simply making API calls to a language model. It promotes applications that are data-aware and agentic, meaning they connect language models with other data sources and enable interaction with the environment. By designing the framework around these principles, LangChain empowers developers to create more powerful and differentiated applications.

The framework encompasses a wide range of use cases, including autonomous agents, agent simulations, personal assistants, question answering systems, chatbots, and tabular data querying. LangChain provides best practices and pre-built implementations for these common use cases, saving developers time and effort in building these functionalities from scratch. Overall, It is a robust and versatile tool for developers seeking to leverage the capabilities of language models in their applications. Its modular design, use-case specific chains, and emphasis on data-awareness and agency make it an ideal choice for creating intelligent and interactive applications. By providing a comprehensive framework with built-in implementations, LangChain streamlines the development process and empowers developers to harness the full potential of large language models.

### **3.2.2.2 FINE TUNING**

Fine-tuning a GPT-3 model involves training the pre-trained language model on a specific task or domain to enhance its performance in that particular area. This process allows for more optimal utilization of the models available through the API, offering superior quality results compared to prompt design. Fine-tuning also enables training on a larger number of examples than can fit in a prompt, leading to token savings through shorter prompts and reduced latency in requests.

The process of fine-tuning can be summarized into three main steps: preparing and uploading training data, training a new fine-tuned model, and utilizing the fine-tuned model. Moreover, it is possible to continue fine-tuning an already fine-tuned model by incorporating additional data, eliminating the need to start from scratch.

To fine-tune a Davinci model, it is necessary to prepare and upload training data in JSON format. Each line in the document represents a prompt-completion pair that corresponds to a

specific training example. Once the training data is ready and uploaded, the fine-tuning process can be initiated using the OpenAI CLI.

Fine-tuning surpasses the limitations of few-shot learning by training on a significantly larger number of examples, resulting in improved performance across a wide range of tasks. Once a model has undergone fine-tuning, it no longer requires explicit examples in the prompt. This not only reduces costs but also enables faster and more efficient requests with lower latency.

When designing prompts and completions for fine-tuning, it is essential to adopt a different approach compared to prompts used with base models. In fine-tuning, each training example typically consists of a single input along with its associated output, eliminating the need for detailed instructions or multiple examples within the same prompt. This streamlined approach simplifies the fine-tuning process and enhances its effectiveness.

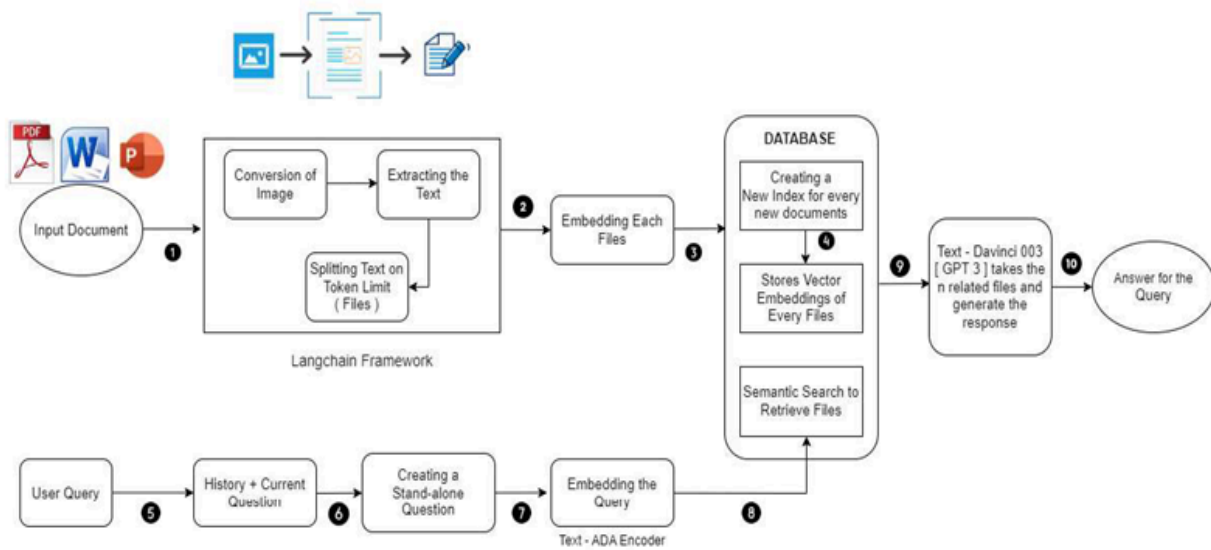
### **3.2.2.3 STREAMLIT**

Streamlit is an open-source Python library that simplifies the creation of interactive web applications for data science, machine learning, and other data-centric tasks. With Streamlit, developers can quickly turn their Python scripts into interactive web applications that can be accessed via a web browser. It offers an intuitive API, enabling developers to create UI components such as sliders, buttons, and plots directly within their Python scripts. Streamlit integrates seamlessly with popular data science libraries like Pandas, Matplotlib, and Plotly, allowing developers to leverage their existing knowledge and tools for data visualization and analysis. Additionally, Streamlit provides customization options for styling and theming, along with built-in deployment capabilities to platforms such as Streamlit Sharing, Heroku, AWS, or Google Cloud Platform. Its vibrant community contributes to its ecosystem, offering third-party libraries, components, and templates to extend its functionality. Overall, Streamlit is a powerful tool for rapidly prototyping and deploying interactive web applications for data science and machine learning tasks.

## CHAPTER IV

### WORKING OF SYSTEM

#### 4.1 SYSTEM ARCHITECTURE



**Figure 4.1: System Architecture**

In this architecture, the File Loader Module, comprising steps 1, 2, and 3, is responsible for handling the initial loading and processing of data. In step 1, the module reads and extracts data from the input files, which could include documents, images, or any other relevant data sources. Step 2 involves preprocessing the extracted data, which may include tasks such as text cleaning, normalization, and feature extraction. Step 3 further prepares the data for indexing or storage, possibly by converting it into a suitable format for efficient retrieval and processing.

Following the File Loader Module, step 4 involves creating a new index in the database to store the processed data or vectors. This index serves as a structured storage mechanism that enables fast and efficient retrieval of relevant information during query processing.

Subsequently, steps 5 through 8 constitute the Query Module, where user interactions with the document take place, and responses are generated. In step 5, the user initiates a query, typically by providing input such as keywords, phrases, or specific criteria. Step 6 involves processing the user query, which may include analyzing and interpreting the query to understand the user's intent and retrieve relevant information.

Once the query is processed, step 7 entails searching the index created in step 4 to retrieve documents or vectors that match the user's query. This step may involve complex algorithms for similarity matching, ranking, and relevance scoring to identify the most relevant results.

Finally, in step 8, the retrieved documents or vectors are presented to the user as a response to their query.

Overall, this architecture facilitates the efficient handling of data loading, indexing, query processing, and response generation, enabling users to interact with the document repository effectively and obtain relevant information in a timely manner.



## 4.2 ALGORITHMS

### 4.2.1 OCR ALGORITHM

---

```
for each idx, file in enumerate(files) do

    Open the file using pdfplumber and assign it to the variable pdf

    for each i in range(len(pdf.pages)) do

        Get the current page from pdf.pages[i] and assign it to the var page

        for each image in page do

            Save the image as a file

            OCR and append the extracted text to txt
        end for
    end for

    Extract tables from the page

    for each table in tables do

        Convert the table to a DataFrame

        Convert to JSON records and append them to txt
    end for

    Append the contents of txt to doc []

end for
```

---

The Algorithm 4.2.1 iterates over a list of files, opens each file using pdfplumber, and extracts images and text from each page. The extracted text is appended to a list called *txt*. Tables are extracted from the pages and converted to JSON records, which are also appended to *txt*. Finally, the contents of *txt* are appended to a list called *doc*.

#### 4.2.2 COSINE SIMILARITY ALGORITHM

---

Start with an empty list *Cos* []

**for** each *i* in Vectors **do**

**for** each *j* in range(*i*, len(Vectors)) **do**

        Calculate Dot Product of *i* and *j* assign to *dot*

        Calculate the magnitude of *i* and assign it to *m1*

        Calculate the magnitude of *j* and assign it to *m2*

        Calculate the cosine similarity as the ratio of *dot* and (*m1* \* *m2*)

        Append to *Cos*

**end for**

**end for**

---

The Algorithm 4.2.2 calculates the cosine similarity between all pairs of vectors in a given list. It iterates through each pair, calculates the dot product, and divides it by the product of the magnitudes of the vectors. The resulting cosine similarity values are appended to a list.

### 4.2.3 TEXT SPLITTING ALGORITHM

---

Start with an empty list *Res* [ ]

Split the text into lines using 'Slash n' as the delimiter

**for** each *line* in SplittedText **do**

Remove any trailing newline characters from the line Append the line to *Res*

**end for**

---

The Algorithm 4.2.3 initializes an empty list called *Res*. It splits a given text into lines using the newline character as a delimiter. For each line, any trailing newline characters are removed, and the line is added to the *Res* list.

## **CHAPTER V**

### **EXPERIMENTAL ANALYSIS**

#### **5.1 SYSTEM CONFIGURATION**

Designing a system for a verbal document engagement system requires a computer system with certain configurations. Here are the recommended specifications:

##### **CPU:**

A multi-core processor with a clock speed of at least 2 GHz is recommended for fast data processing.

##### **GPU:**

A powerful graphics processing unit (GPU) with at least 4GB of memory is recommended for training deep neural networks.

##### **RAM:**

At least 16GB of RAM is recommended to handle large datasets and to prevent memory-related issues during training.

##### **Storage:**

A solid-state drive (SSD) with at least 256GB of storage is recommended to store the training data and model parameters.

##### **Operating System:**

A 64-bit operating system such as Windows 10 or Ubuntu is recommended to utilize the full potential of the hardware.

## Software:

The system should have Python installed along with necessary libraries for natural language processing (NLP), machine learning (ML), and web application development. Common libraries include NLTK, TensorFlow, PyTorch, Scikit-learn, Streamlit, and others.

## Additional Hardware:

Depending on the size of the dataset and the complexity of the model, additional hardware such as external hard drives or cloud-based computing resources may be necessary.

## 5.2 SAMPLE CODE

```
1. import streamlit as st
2. from dotenv import load_dotenv
3. import pickle
4. from PyPDF2 import PdfReader
5. from streamlit_extras.add_vertical_space import add_vertical_space
6. from langchain.text_splitter import RecursiveCharacterTextSplitter
7. from langchain.embeddings.openai import OpenAIEmbeddings
8. from langchain.vectorstores import FAISS
9. from langchain.llms import OpenAI
10. from langchain.chains.question_answering import load_qa_chain
11. from langchain.callbacks import get_openai_callback
12. import os
13.
14. # Sidebar contents
15. with st.sidebar:
16.     st.title('LLM Chat App')
17.     st.markdown("""
18.     ## About
19.     This app is an LLM-powered chatbot built using:
20.     - [Streamlit](https://streamlit.io/)
21.     - [LangChain](https://python.langchain.com/)
22.     - [OpenAI](https://platform.openai.com/docs/models) LLM model
```

```

23.
24.     "")
25.     add_vertical_space(5)
26.
27.
28. load_dotenv()
29.
30. def main():
31.     st.header("Chat with PDF ")
32.
33.
34.     # upload a PDF file
35.     pdf = st.file_uploader("Upload your PDF", type='pdf')
36.
37.     # st.write(pdf)
38.     if pdf is not None:
39.         pdf_reader = PdfReader(pdf)
40.
41.         text = ""
42.         for page in pdf_reader.pages:
43.             text += page.extract_text()
44.
45.         text_splitter = RecursiveCharacterTextSplitter(
46.             chunk_size=1000,
47.             chunk_overlap=200,
48.             length_function=len
49.         )
50.         chunks = text_splitter.split_text(text=text)
51.
52.         # # embeddings
53.         store_name = pdf.name[:-4]
54.         st.write(f'{store_name}')
55.         # st.write(chunks)
56.

```

```

57.     if os.path.exists(f'{store_name}.pkl'):
58.         with open(f'{store_name}.pkl', "rb") as f:
59.             VectorStore = pickle.load(f)
60.             # st.write('Embeddings Loaded from the Disk')s
61.     else:
62.         embeddings = OpenAIEmbeddings()
63.         VectorStore = FAISS.from_texts(chunks, embedding=embeddings)
64.         with open(f'{store_name}.pkl', "wb") as f:
65.             pickle.dump(VectorStore, f)
66.
67.     # embeddings = OpenAIEmbeddings()
68.     # VectorStore = FAISS.from_texts(chunks, embedding=embeddings)
69.
70.     # Accept user questions/query
71.     query = st.text_input("Ask questions about your PDF file:")
72.     # st.write(query)
73.
74.     if query:
75.         docs = VectorStore.similarity_search(query=query, k=3)
76.
77.         llm = OpenAI()
78.         chain = load_qa_chain(llm=llm, chain_type="stuff")
79.         with get_openai_callback() as cb:
80.             response = chain.run(input_documents=docs, question=query)
81.             print(cb)
82.             st.write(response)
83.
84. if __name__ == '__main__':
85.     main()

```

## **5.3 PERFORMANCE ANALYSIS**

Performance analysis of a document querying application involves evaluating the effectiveness and efficiency of the system in retrieving relevant documents. Here are the steps for performance analysis:

### **Query Generation**

Create a set of test queries representing typical user queries that the document querying application is expected to handle. These queries should cover a range of topics, complexities, and query types to evaluate the system's responsiveness and accuracy.

### **Evaluation Metrics Selection**

Choose appropriate evaluation metrics to measure the performance of the document querying application. Common metrics include precision, recall, F1-score, mean average precision (MAP), and mean reciprocal rank (MRR).

### **Scalability Testing**

Assess the scalability of the document querying system by evaluating its performance under varying document collection sizes, query loads, and concurrent user requests. Measure how the system's response time and resource utilization scale with increasing workload.

### **Optimization and Iteration**

Identify performance bottlenecks and areas for improvement based on the evaluation results. Optimize the system's algorithms, data structures, and configurations to enhance its efficiency and effectiveness. Iterate on the performance analysis process to continuously refine and improve the document querying application.



## **5.4 PERFORMANCE MEASURES**

Evaluation metrics for verbal document engagement systems play a crucial role in assessing their performance and effectiveness. These metrics enable the objective measurement of how well a QA system is able to provide accurate and relevant answers to given questions.

### **5.4.1 Exact Match**

The Exact Match (EM) metric is a binary evaluation measure used in Question Answering (QA) systems. It assesses the accuracy of the model's predicted answer by comparing it with the true answers. The EM metric determines if the characters of the model's prediction exactly match the characters of any of the true answers. If there is an exact match, the EM score is 1, indicating a correct prediction. However, even a single character difference results in a score of 0. The EM metric is strict, providing a clear distinction between correct and incorrect predictions. It is commonly used in QA evaluation to measure the system's ability to produce precise and exact answers.

### **5.4.2 F1 Score**

F1 score is a common metric for classification problems, and widely used in QA. It is appropriate when we care equally about precision and recall. In this case, it's computed over the individual words in the prediction against those in the True Answer. The number of shared words between the prediction and the truth is the basis of the F1 score: precision is the ratio of the number of shared words to the total number of words in the prediction, and recall is the ratio of the number of shared words to the total number of words in the ground truth.

### **5.4.3 Semantic Answer Similarity**

Semantic Answer Similarity (SAS) is an evaluation metric used in Question Answering (QA) systems to measure the degree of semantic agreement between the predicted answer and the reference answers. SAS takes into account the meaning and semantic equivalence of the answers, rather than relying solely on character-level or token-level matches. It allows for partial matches and considers the overall semantic similarity, providing a more nuanced evaluation of the system's ability to generate answers that align with the meaning and intent of the reference answers.

## **5.5 RESULTS AND DISCUSSIONS**

A Closed Domain QA system is designed to operate within a specific domain or topic and can provide accurate answers to questions within that limited scope. On the other hand, the proposed system using the LangChain framework offers a more comprehensive and versatile approach to document interaction and information retrieval. Here's a comparison between the two:

### **5.5.1 Scope and Flexibility**

CDQA System: Limited to a specific domain or topic, often requiring manual annotation or training on a specific dataset.

Proposed System: Can handle a wide range of document types and formats, such as PDF, Word, and PPTX. It is not restricted to a single domain and can be applied to various use cases like chatbots, GQA, and summarization.

### **5.5.2 Data Extraction and Processing**

CDQA System: Relies on pre-defined rules and templates specific to the domain to extract relevant information.

Proposed System: Utilizes the LangChain framework to extract text, graphics, and tables from files. The content is then preprocessed and encoded into embeddings, capturing semantic meaning in a numerical format.

### **5.5.3 Long-Term Memory and Retrieval**

CDQA System: Does not typically have a dedicated long-term memory for storing processed text and embeddings.

Proposed System: Utilizes the vector database to store embeddings, enabling efficient retrieval and serving as a reliable long-term memory for the system.

#### **5.5.4 Query Processing and Context**

CDQA System: Processes individual queries without considering the context or query history.

Proposed System: Considers the query history and forms standalone questions related to previous queries. The standalone question is encoded using the ADA encoder and subjected to a semantic search to find closely matched embeddings, retrieving relevant documents.

#### **5.5.5 Response Generation**

CDQA System: Relies on predefined prompts and templates for generating responses.

Proposed System: Utilizes a fine-tuned Language Model (LLM) to generate responses based on the standalone question and the retrieved documents. The LLM takes into account the context provided by the documents, resulting in more accurate and informative replies.

Overall, the proposed system using the LangChain framework offers several advantages over a CDQA system. It provides a more flexible and adaptable approach to document interaction, supports various document types, incorporates long-term memory for efficient retrieval, considers query history, and utilizes a fine-tuned LLM for generating context-aware responses. These features make the proposed system more versatile, efficient, and capable of handling complex information retrieval tasks across different domains.

## **CHAPTER VI**

### **CONCLUSION AND RECOMMENDATIONS**

#### **6.1 CONCLUSION**

In conclusion, the verbal document engagement system represents a major breakthrough in the realm of document interaction. By harnessing the capabilities of Artificial Intelligence and Natural Language Processing, this innovative technology empowers users to engage with their documents in a conversational manner, mimicking human-like interactions. The potential impact of a verbal document engagement system on document management and information retrieval cannot be overstated. The ability to converse with documents in natural language brings forth a new era of user-friendly and efficient document interaction. Professionals and students grappling with large volumes of complex documents, such as legal contracts, research papers, and textbooks, will benefit immensely from the ease and speed with which they can extract information and find answers to inquiries. Moreover, teachers will find value in the time-saving features of Conversational Document when creating question papers for assessments.

#### **6.2 RECOMMENDATIONS FOR FUTURE WORK**

In future development, verbal document engagement system aims to include essential features to enhance user experience. The integration of a chat history function, which enables users to preserve and revisit their earlier chats with documents, is a key feature. Users can make use of this feature to look back on previous interactions, check progress, and keep a record of their effort. The addition of new tabs for various documents within the interface, allowing users to work on many projects at once, is another useful improvement. Having the ability to multitask boosts productivity and simplifies document management. It will also focus on providing the summary of the document which will help the user to know the centric of the document at ease and make them to bring an overview about context.

## REFERENCES

- [1] Devanshi Singh, K Rebecca Suraksha, and S Jaya Nirmala, “Question Answering Chatbot using Deep Learning with NLP”, In IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, pages 1–6, 2021.
- [2] Sweta P Lende and MM Raghuwanshi, “Question Answering System On Education Acts Using NLP Techniques”, In IEEE, World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), Coimbatore, India, pages 1–6, 2016.
- [3] Qilu Jiao and Shunyao Zhang, “A Brief Survey of Word Embedding and its Recent Development”, In IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, volume 5, pages 1697–1701, 2021.
- [4] Priti Gumaste, Shreya Joshi, Srushtee Khadpekar, and Shubhangi Mali, “Automated Question Generator System using NLP Libraries”, International Research Journal of Engineering and Technology (IRJET),,Maharashtra, India, Volume: 07, Issue: 06, 2020.
- [5] Miss Shivangi Daware, Chincholi SVIT, Miss Rachana Ankar, Mr Kishor Shedge, and Miss Priyanka Phulmogare, “Automatic Generation of Question Paper using NLP Techniques”, In National Level Conference On "Advanced Computing and Data Processing(ACDP 2K19) Nashik, India,2019.
- [6] R Devika, Subramaniaswamy Vairavasundaram, C Sakthi Jay Mahenthara, Vijayakumar Varadarajan, and Ketan Kotecha, “A Deep Learning Model Based on Bert and Sentence Transformer for Semantic Keyphrase Extraction on Big Social Data”, Maharashtra, India, IEEE Access, Volume : 9, Pages: 165252–165261, 2021.
- [7] Md Tahmid Rahman Laskar, Xiangji Huang, and Enamul Hoque, “Contextualized Embeddings based Transformer Encoder for Sentence Similarity Modeling in Answer Selection Task”, In Proceedings of the Twelfth Language Resources and Evaluation Conference, Marseille, France, pages 5505–5514, 2020.

- [8] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha, “Docformer: End-To-End Transformer for Document Understanding”, In Proceedings of the IEEE/CVF international conference on computer vision, pages 993–1003, 2021.
- [9] Miriam Fernandez, Vanessa Lopez, Marta Sabou, Victoria Uren, David Vallet, Enrico Motta, and Pablo Castells, “Semantic Search Meets the Web”. In IEEE international conference on semantic computing, ,CA, USA, pages 253–260, 2008.

## OUTPUT SCREENS

