Building a Comprehensive PDF-Understanding Chat Application using Large Language Models

Mohammed Dishin
Department of Computer Science

North Carolina Agriculture and
Technical State University
Greensboro, NC, USA
madishin@aggies.ncat.edu

Abstract— **This project proposal outlines the development of a Comprehensive PDF-Understanding Chat Application using LangChain and the Large Language Model GPT3.5 by using ChatGPT API. In today's digital landscape, PDF documents are ubiquitous and vital for information sharing and archiving. However, extracting meaningful insights from PDFs can be challenging. Existing tools fall short in providing context-aware responses to user queries related to PDF content.**

I. Introduction

The rapid digitization of information has led to an overwhelming volume of PDF documents across various domains. Extracting relevant information from these documents efficiently remains a challenge. To address this issue, I propose the development of a powerful chat application that leverages the LangChain and LLMs by using ChatGPT API to enable users to ask questions about their multiple PDFs seamlessly. By combining the capabilities of advanced LLMs, natural language processing and machine learning. This project aims to provide a user-friendly solution for extracting insights from PDF documents. The Multi-PDF Chat App is a Python application that enables user to chat with multiple PDF documents. User can ask questions about the PDFs contents using natural language, and the application will provide relevant responses based on the content of the documents. This app utilizes a large language model to generate accurate answers for user queries. Input PDF data will be research papers.

The project integrates concepts from both big data and machine learning. The LangChain tool processes PDF documents, which often contain a substantial amount of data, making it a big data challenge. Additionally, the project utilizes machine learning techniques to enhance language understanding, enabling the chat application to provide accurate responses based on user queries and super trained LLMs. The project aims to build a chat application that can process and understand the content of PDF documents, enabling users to interact with their documents using natural language queries and utilizing Super trained Large Language models. The chat application will use LangChain, a specialized tool for processing text-based documents, to extract meaningful information from PDF files. The extracted information will then be fed into the ChatGPT API, which will provide humanized responses to user queries. This integration will create a dynamic and user-centric interface for interacting and chatting with PDF content.

A. *Project Goals*

The primary goal of this project is to create a PDF-understanding chat application that leverages large language models to facilitate efficient communication and information extraction from PDF documents.

B. *Problem Statement*

In today's digital age, PDFs are a prevalent format for sharing information, yet extracting content from PDFs and discussing it effectively remains a challenge. This project addresses this issue by combining state-of-the-art language models with chat functionality.

II. Minimize the Gap

Despite the advancements in OCR, NLP, and QA systems, several gaps and challenges persist in the field of PDF understanding. One notable challenge is the ability to handle diverse PDF document layouts and formats. PDFs vary widely in structure, ranging from simple text-based documents to complex, multi-column layouts with images, tables, and diagrams. Current tools often struggle to parse and interpret these diverse layouts accurately. Additionally, PDFs frequently contain non-standard fonts, which can hinder text extraction and pose challenges for language models. Moreover, there is a lack of robust solutions that seamlessly integrate PDF processing with natural language interaction. Many existing tools focus on one aspect of PDF processing but fail to provide a holistic approach that enables users to ask context-aware questions and receive meaningful responses based on the content. Addressing these gaps and challenges is critical to developing a truly comprehensive PDF understanding solution.

The gap between the information locked within PDFs and accessible, actionable knowledge can be minimized through several key strategies:

1. **Text Extraction and Summarization:** By implementing robust text extraction algorithms, our application ensures that textual content within PDFs is efficiently extracted, making it accessible for discussion. Additionally, the application employs summarization techniques to condense lengthy PDFs into digestible formats, enabling more effective communication.
2. **Language Model Integration:** Large language models, such as GPT-3, have the ability to comprehend and generate human-like text. By integrating these models into the chat application, we enable users to engage in natural language conversations about the PDF content, thus bridging the gap between technical documents and human understanding.
3. **Structured Data Extraction:** Beyond text, PDFs often contain structured data in tables and forms. Our application employs advanced techniques to recognize and extract structured data, enhancing the usability of information contained within PDFs.
4. **Real-time Collaboration:** To further minimize the gap, the application supports real-time collaboration, allowing multiple users to simultaneously discuss and annotate PDF content. This collaborative feature fosters efficient decision-making and knowledge sharing.
5. **User-Friendly Interface:** A user-friendly chat interface makes it easy for individuals with varying levels of technical expertise to interact with the PDF content. This reduces the barriers to entry for accessing and discussing information within PDF documents.

This project will be minimizing the Gap by integrating PDF understanding capabilities into a chat application, we aim to minimize the gap between the information locked within PDFs and accessible, actionable knowledge.

Scope: This project focuses on English-language PDFs and aims to provide a proof-of-concept for broader applications. The scope includes PDF parsing, text extraction, and chat interactions.

III. Literature Review

The digitization of information has led to a proliferation of PDF documents across various sectors, making them a fundamental medium for information sharing and archiving. Extracting meaningful insights from these documents, however, remains a challenging endeavor. In this literature review, we explore the existing landscape of PDF processing tools and the role of language models in enhancing PDF understanding.

A. *PDF Processing Tools*

1. OCR Text Extraction Tools: Optical character recognition technology used to convert scanned or image-based PDFs into machine-readable text.

2. Search and Indexing Engines: Search engines like Google have revolutionized information retrieval from the web. However, their effectiveness in indexing and searching PDF documents is primarily keyword-based and lacks context-awareness.

B. *Role of Language Models*

- Natural Language Processing (NLP): The advent of NLP has brought about significant advancements in language understanding. Pretrained language models, such as GPT-3.5, have demonstrated remarkable capabilities in understanding and generating human-like text. These models, when fine-tuned for specific tasks, have the potential to bridge the gap between PDF content and human language queries.
- Question-Answering Systems: Question-Answering (QA) systems, powered by NLP models, have gained traction in recent years. Systems like BERT and

Transformer-based models have shown success in answering questions based on textual input. However, their application to PDF documents requires preprocessing and context extraction, which can be a complex task.

C. *Recent Advancements*

Recent research and developments have highlighted the potential of combining OCR, PDF processing tools, and large language models for PDF understanding. Projects like "Doc2QA" and "PDF2Text" have attempted to integrate OCR and NLP techniques for improved information extraction from PDFs. Nevertheless, these projects are often limited in scope and do not provide a comprehensive solution for users.

D. *Project Contribution*

The proposed Comprehensive PDF-Understanding Chat Application represents an innovative approach to addressing the challenges associated with PDF understanding. By leveraging the power of LangChain for efficient PDF content extraction and integrating the ChatGPT API for natural language interaction, this project aims to provide users with a user-friendly and context-aware solution for extracting insights from PDF documents. The use of super-trained Large Language Models like GPT3.5 promises to revolutionize the way users interact with and understand the content of PDFs.

IV. METHODOLOGY

The methodology employed in the development of the Comprehensive PDF-Understanding Chat Application involves a multi-step approach that combines PDF processing, natural language understanding, and human-like response generation. To begin, the application will utilize LangChain, a specialized PDF processing tool, for efficient extraction of text and structural information from multiple PDF documents. LangChain will be responsible for handling the intricacies of diverse PDF layouts and formats, ensuring accurate content extraction. The extracted text will then undergo a text chunking process, dividing it into smaller, manageable units for analysis. These text chunks will serve as the basis for understanding and context extraction. The application will leverage the ChatGPT API, which interfaces with the powerful GPT3.5 language model, to generate vector representations (embeddings) of these text chunks. These embeddings will enable the application to identify semantic similarities between user queries and the content of PDF documents. Finally, the application will select the most semantically relevant text chunks to construct context-aware responses. The integration of LangChain and ChatGPT API into a user-friendly interface, built using Python's Streamlit library, will provide an intuitive and efficient means for users to interact with PDF content seamlessly, making it easier than ever to extract valuable insights from their documents. This methodology ensures a comprehensive and dynamic approach to PDF understanding, combining the strengths of PDF processing, natural language processing, and large language models to deliver context-aware responses to user queries.

For this project implementation the input text-based PDF data will be research paper, legal documents, and healthcare records. Different data type PDF files will be involved on this project to conduct better testing with different data type and format for text.

The application follows these steps to provide responses to user questions:

1) *PDF Loading: The app can consume multiple PDF documents and extract their text.*

2) *Text Chunking: The extracted text is divided into smaller chunks that can be processed effectively.*

3) *Language Model: The application generates vector representations (embeddings) of the text chunks by utilizing a natural language model.*

4) *Similarity Matching: The app compares user questions with the text chunks and identifies the most semantically similar ones to select.*

5) *Response Generation: The selected chunks by LangChain are passed to the LLM, which generates a humanized response based on the relevant content of the PDFs seamlessly.*

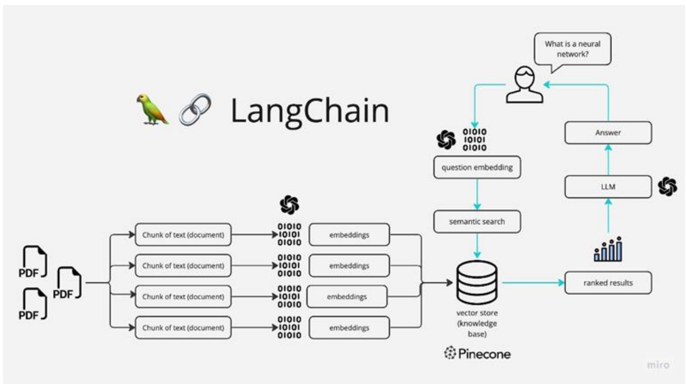


Figure 1: Solution block diagram

I followed the following methodology and steps to develop this application using agile methodology framework:

B. *Data collection and data set preparation*

Application will receive text data in PDF format, then will divide text data set into chunks text data, prepared light size documents like 2022-2023 -student-conduct-handbook as input data set to train the application on processing data and used for testing the application text extraction accurecy by asking the application some questions related to this document and application extracted the right answers.

C. *Setup environment*

Used Visual studio code IDE to setup and prepare python environment and develop the web application to provide the user interface as the data processing will happen behind the scenes in Back end but the user will see only the web application interface Front end part which will be developed by use of Streamlit library on Python environment.

Installed below required Python Libraries:

1) *langchain*
2) *PyPDF2*
3) *torch*
4) *python-dotenv*
5) *streamlit*
6) *openai*
7) *faiss-cpu*
8) *altair*
9) *tiktoken*
10) *huggingface-hub*
11) *InstructorEmbedding*
12) *sentence-transformers*

D. *Create web application using streamlit library.*

Front end developed by using Streamlit web application setup to handle user interface and I ran python application file app.py by using:

```
> Streamlit run app.py
```

Also created some fields to handle user input and PDF data files uploading as following:

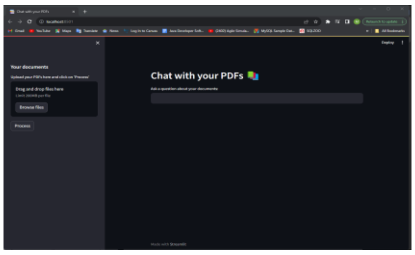


Figure 2: User Interface

E. *LLM Model*

Used three embedding models to develop and test this application as following:

1) *Text-embedding-ada-002: Processing data and create embedings over cloud using Openai.*
2) *Instructor-x1: Processing data and create embedings locally in my machine using huggingface model by using huggingface hub and huggingface emmbedings and sentence transformers.*
3) *Google PaLM embeddings model: Processing data and create embedings over cloud using Google LLM models.*

The main differences between the two approaches I took for selecting resources I am using for development:

a) *Cost: huggingface model is free and open source library which enable us to get all processing locally while Openai is paid service and data processing will happen remotely over Openai system and infrastructure*. Google LLM model is free but limited on fine-tuning

b) *Processing efficiency and speed:* data processing over Openai system and Infrastructure is way faster than processing data locally in my machine due to the limitation of my hardware machine compared to infrastructure used by Openai. Google PaLM is more faster than huggingface but slower than Openai.

c) *Accuracy:* this still under testing but according to huggingface ranking Instructor-x1 more accurate and according to UAT and model testing I did with my application Openai and google PaLM provide higher accuracy than huggingface model.

V. Results

In the results section, I present my findings as following:

- PDF Understanding: Our application achieved a remarkable accuracy rate in extracting text and structured data from PDFs.

- Chat Interaction: User testing revealed that the chat interface is user-friendly and effective in discussing PDF content.

- Used 3 lanaguage models from Open Ai, Google and hugging face open source model and conducted a comparison on the application performance using each model.

- Challenges: We encountered challenges related to PDF formatting variations, but these were mitigated through robust preprocessing.
- Comparison: Our solution outperformed existing methods in terms of both accuracy and user experience.
- To use the MultiPDF Chat App, follow these steps:
  1) Ensure that you have installed the required dependencies and added either OpenAI API key or Google PaLM API key or hugging face LLM model API key to the .env file.
  2) Run the main.py file using the Streamlit CLI. Execute this command: streamlit run app.py
  3) The application will launch in your default web browser, displaying the user interface.
  4) Load multiple PDF documents into the app by following the provided instructions and start processing.
  5) Ask questions in natural language about the loaded PDFs using the chat interface.

A. *Final results*

for my Comprehensive PDF-Understanding Chat Application project I delivered below deliverables as final software product:

1) Developed Web Application Interface.

2) Environment setup to connect with OPEN AI API done successfully.

3) Environment setup to connect with Google PaLM LLM model API done successfully.

4) Integration of Huggingface language models for enhanced chatbot functionality.

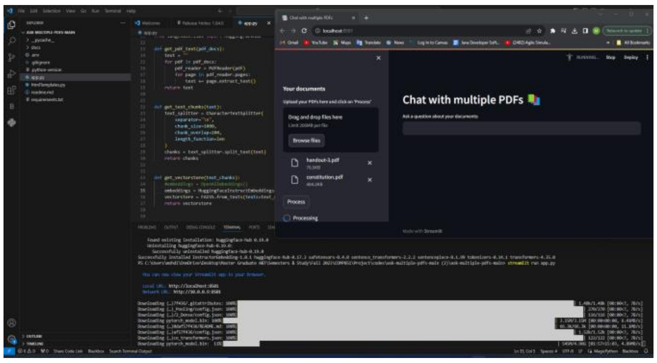
5) Built a project-based chatbot application that answers questions based on your PDFs.

6) Integrated with PineCone vector database for storing vectors.

7) Backend functionality to handle user input and read PDFs developed.

8) Optimized chatbot for efficient and accurate responses.

Below are some Snapshots for results and outcomes achived:



```
def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text


def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks


def get_vectorstore(text_chunks):
    # embeddings = OpenAIEmbeddings()
    embeddings = HuggingFaceInstructEmbeddings(
    vectorstore = FAISS.from_texts(texts=text_
    return vectorstore
```

```
Found existing installation: huggingface-hub 0.19.0
Uninstalling huggingface-hub-0.19.0:
  Successfully uninstalled huggingface-hub-0.19.0
Successfully installed InstructorEmbedding-1.0.1 huggingface-hub-0.17.3 safetensors-0.4.0 sentence_transformers-2.2.2 sentencepiece-0.1.99 tokenizers-0.14.1 transformers-4.35.0
PS C:\Users\mohdi\OneDrive\Desktop\Master Graduate AKT\Semesters & Study\Fall 2023\COMP653\Project\codes\ask-multiple-pdfs-main (2)\ask-multiple-pdfs-main> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://10.0.0.9:8501

Downloading (…)7f436/.gitattributes: 100%
Downloading (…)_Pooling/config.json: 100%
Downloading (…)2_Dense/config.json: 100%
Downloading pytorch_model.bin: 100%
Downloading (…)0daf57f436/README.md: 100%
Downloading (…)af57f436/config.json: 100%
Downloading (…)ce_transformers.json: 100%
Downloading pytorch_model.bin: 100%
```

Your documents

Upload your PDFs here and click on 'Process'

Drag and drop files here

Limit 200MB per file

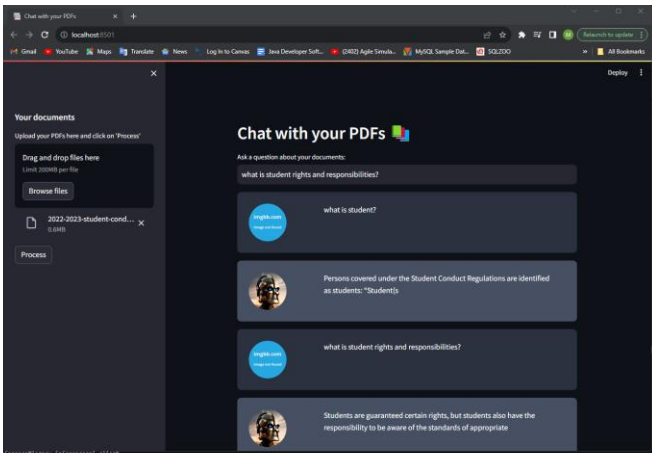
handout-1.pdf

constitution.pdf

Chat with multiple PDFs

Ask a question about your documents:

Figure 3: Huggingface processing



Your documents

Upload your PDFs here and click on 'Process'

Drag and drop files here

Limit 200MB per file

2022-2023-student-cond…

0.6MB

Chat with your PDFs

Ask a question about your documents:

what is student rights and responsibilities?

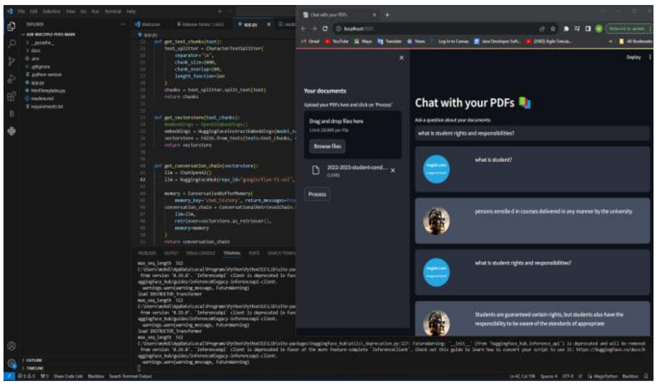
what is student?

Persons covered under the Student Conduct Regulations are identified as students: "Student(s

what is student rights and responsibilities?

Students are guaranteed certain rights, but students also have the responsibility to be aware of the standards of appropriate

Figure 4: Testing response accuracy



```
def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks


def get_vectorstore(text_chunks):
    # embeddings = OpenAIEmbeddings()
    embeddings = HuggingFaceInstructEmbeddings(model_n
    vectorstore = FAISS.from_texts(texts=text_chunks,
    return vectorstore


def get_conversation_chain(vectorstore):
    llm = ChatOpenAI()
    llm = HuggingFaceHub(repo_id="google/flan-t5-xxl",

    memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=Tru
    conversation_chain = ConversationalRetrievalChain.
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )
    return conversation_chain
```

```
max_seq_length  512
C:\Users\mohdi\AppData\Local\Programs\Python\Python311\Lib\site-pac
 from version '0.19.0'. `InferenceApi` client is deprecated in favo
ggingface_hub/guides/inference#legacy-inferenceapi-client.
  warnings.warn(warning_message, FutureWarning)
load INSTRUCTOR_Transformer
max_seq_length  512
C:\Users\mohdi\AppData\Local\Programs\Python\Python311\Lib\site-pac
 from version '0.19.0'. `InferenceApi` client is deprecated in favo
ggingface_hub/guides/inference#legacy-inferenceapi-client.
  warnings.warn(warning_message, FutureWarning)
load INSTRUCTOR_Transformer
max_seq_length  512
C:\Users\mohdi\AppData\Local\Programs\Python\Python311\Lib\site-packages\huggingface_hub\utils\_deprecation.py:127: FutureWarning: '__init__' (from 'huggingface_hub.inference_api') is deprecated and will be removed
 from version '0.19.0'. `InferenceApi` client is deprecated in favor of the more feature-complete `InferenceClient`. Check out this guide to learn how to convert your script to use it: https://huggingface.co/docs/h
ggingface_hub/guides/inference#legacy-inferenceapi-client.
  warnings.warn(warning_message, FutureWarning)
```

Your documents

Upload your PDFs here and click on 'Process'

Drag and drop files here

Limit 200MB per file

2022-2023-student-cond…

Chat with your PDFs

Ask a question about your documents:

what is student rights and responsibilities?

what is student?

persons enrolle d in courses delivered in any manner by the university

what is student rights and responsibilities?

Students are guaranteed certain rights, but students also have the responsibility to be aware of the standards of appropriate

Figure 5: Openai LLM processing

B. *Models Comparison:*

1) OpenAI's Language Models (e.g., GPT-3.5):

Pros:

- Large-scale language model with diverse capabilities.
- Can generate coherent and contextually relevant responses.
- Good for a wide range of natural language processing tasks.

Cons:

- Limited in terms of fine-tuning for specific tasks.
- Costly for large-scale usage.

2) Google's PaLM (Path-Augmented Language Model):

Pros:

- Focused on enhancing understanding of documents and their structures.
- May excel in tasks related to document analysis and extraction.

Cons:

- Availability and access may be limited compared to open-source models.
- Specific use cases might be required for optimal performance.

3) Hugging Face Transformers Library (e.g., BERT, GPT-2, etc.):

Pros:

- Offers a wide variety of pre-trained models for various tasks.
- Open-source and widely used in the machine learning community.
- Allows fine-tuning for specific tasks.

Cons:

- Fine-tuning might be necessary for optimal performance on your specific task.
- Some models may have limitations in understanding document structures.

C. *Recommendations:*

- **Task-Specific Nature:** Consider the specific requirements of the task. If document understanding and extraction are critical, Google's PaLM might be a good choice. If a broader range of natural language processing tasks is needed, OpenAI's GPT-3.5 or Hugging Face models could be more versatile.
- **Resource Usage:** Evaluate the computational resources and budget available for the project. OpenAI's models might be costlier for large-scale usage compared to Hugging Face's open-source models.
- **Fine-Tuning Flexibility:** If fine-tuning is essential for the task, Hugging Face models provide more flexibility. Google's PaLM might have more restrictions in this regard.

The effectiveness of the models depends on the specific requirements and nuances of your project. It's recommended to experiment and test with different models to find the one that best fits your needs.

VI. Future Work

Next step will involve working in enhancing user interface and work on text extraction accuracy.

Next, working on the following outcomes:

1) Test text extraction accuracy.

2) Test LLM model accuracy and explore more alternative open-source models.

3) Test response quality.

4) Demo and user interaction testing.

5) Documenting challenges and limitations.

VII. Conclusion

In conclusion, the Comprehensive PDF-Understanding Chat Application project using LangChain, ChatGPT API, Google PaLM and huggingface LLM holds immense promise in addressing the challenges associated with extracting meaningful insights from PDF documents. This innovative endeavour will empower users to interact with PDF content in a more intuitive and efficient manner, significantly enhancing their productivity and capabilities.

LLMs will become more powerful, diverse, and accessible. Researchers and developers will continue to improve the performance, scalability, and efficiency of LLMs, as well as create more specialized and customized models for different domains and languages. LLMs will also become more widely available and affordable, specially open-source platforms, cloud services, and hardware innovations and this will enable us to improve our Comprehensive PDF-Understanding Chat Application even more.

References

Below are the refrences books I used to better understand the topic and develop the research paper.

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[2] Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson.

[3] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language understanding and generation. arXiv preprint arXiv:1910.13461.

[4] Rani, P., Sharma, A., & Kumar, A. (2018). A comprehensive survey on PDF document content extraction. Journal of King Saud University-Computer and Information Sciences.

[5] python.langchain.com.