Report on Pdf Answering AI (ArIES)-IITR

Table of Content:

1.Objective
2.Choice of Model
3.Implementation steps
    3.1.Setup your Environment
    3.2.Create and Write the Streamlit Script
3.3.ExtractionRun the Streamlit App

4.Model Training
5.Learning from This Project
6.Conclusion
7.Future Work and Improvements

Objective:

The objec)ve of this project is to create a web-based ques)on answering system that can extract text from a PDF document and answer ques)ons based on the extracted text. The system will use Flask for the web applica)on framework and the Hugging Face Transformers library for the ques)on-answering model.

Choice of Model:

For this project, we chose the ques)on-answering pipeline from the Hugging Face Transformers library. This library provides state-of-the art models for natural language processing tasks. The specific model used in this pipeline is based on BERT (Bidirec)onal Encoder Representa)ons from Transformers), a powerful transformer-based model pre-trained on a large corpus of text.

Reasons for Choosing BERT:

Accuracy: BERT has been shown to perform excep)onally well on ques)on-answering tasks.
Pre-trained: Using a pre-trained model saves significant)me and computa)onal resources.
Flexibility: BERT can be fine-tuned for specific tasks with rela)vely liMle data.

Support: The Hugging Face library provides a user-friendly API for implemen)ng the BERT model.

Implementation steps

1.)Setup your Environment:

1.1.Navigate to Your Project Directory: Open your terminal and navigate to the directory where you want to create your project.
1.2.Create a Virtual Environment: Create a virtual environment named venv using python3 -m venv.
1.3.Ac)vate the Virtual Environment: Ac)vate the virtual environment.
1.4.Install Required Packages: Install the necessary Python packages using pip.

2.) Create and Write the Streamlit Script:

2.1.Create a New Python Script: Create a new Python file named streamlit_app.py in your project directory.
2.2.Write the Code: Copy the provided code into streamlit_app.py.

3.) Extraction Run the Streamlit App:

3.1.Run the Streamlit Command: In your terminal, make sure your virtual environment is ac)vated and run the Streamlit app using the following command: (streamlit run streamlit_app.py)
3.2.Open the Streamlit App in a Browser: A[er running the command, Streamlit will start a local server and provide you with a URL (e.g., hMp://localhost:8501). Open this URL in your web browser to access the app.

3.3.Upload a PDF File: Use the file uploader in the Streamlit app to upload a PDF file
3.4.Ask a Ques)on: A[er the PDF text is extracted and displayed, input a ques)on in the text box provided. The app will use the ques)on answering model to find the answer based on the extracted text from the PDF..

PDF Text Extraction:

1. PyMuPDF Integra)on:
1.1 Use PyMuPDF to open and read the PDF file.
1.2 Extract text from each page of the PDF and concatenate it into a single string.

2.Func)on Defini)on:
2.1Define extract_text_from_pdf(pdf_path) func)on to perform the text extrac)on.

Question Answering Pipeline:

1. Hugging Face Integra)on:
1.1 Ini)alize the ques)on-answering pipeline using the Transformers library.
1.2 Define answer_ques)on (ques)on, context) func)on to get answers from the model.

2. AJAX Handling:
2.1 Use AJAX to send the ques)on and context to the Flask backend. 2.2 Return the answer as a JSON response.

Model Training:

In this project, we u)lize a pre-trained BERT model for the ques)on answering task. Fine-tuning a BERT model involves the following steps:

1. Data Collec)on:
1.1 Gather a dataset with context-ques)on-answer pairs, such as the SQuAD dataset.

2. Data Pre-processing:
2.1 Tokenize the context and ques)ons.
2.2 Prepare the input data in a format compa)ble with BERT.

3. Training:
3.1 Fine-tune the pre-trained BERT model on the dataset using the Hugging Face Transformers library.
3.2 Adjust hyperparameters such as learning rate, batch size, and number of epochs for op)mal performance.

4.Evalua)on:
4.1 Evaluate the model on a valida)on set to monitor performance.
4.2 Use metrics such as Exact Match (EM) and F1 score to assess accuracy.

5.Deployment:
5.1 Save the fine-tuned model and integrate it into the Flask applica)on.

Learning from This Project:

This project provided several key learning

experiences: 1.Understanding NLP Models:

1.1Gained insights into how transformer-based models like BERT work and their applica)on in NLP tasks.

2.Web Development:
2.1Learned how to create a web applica)on using Flask.
2.2Understood the process of handling file uploads and processing user input.

3.PDF Text Extrac)on:
3.1Explored methods to extract text from PDF documents using PyMuPDF.

4.Model Integra)on:
4.1Successfully integrated a machine learning model into a web applica)on.
4.2Dealt with challenges related to model inference and response handling in a web environment.

Conclusion:

Overall, the project demonstrates the prac)cal applica)on of transformer models in real-world scenarios by crea)ng a system that can extract text from PDF files and provide accurate answers to user ques)ons. This not only highlights the capabili)es of advanced NLP models but also opens up numerous possibili)es for future development and applica)on in various domains. The success of this project lies in its ability to combine sophis)cated machine learning

techniques with accessible web technologies, making advanced NLP tools available to a broader audience

Future Work and Improvements:

There are several areas for future improvement:

1.User Interface:
1.1Enhance the user interface to make it more intui)ve and user friendly.

2.Performance Op)miza)on:
2.1Op)mize the text extrac)on and model inference processes for faster response)mes.

3.Scalability:
3.1Explore ways to scale the applica)on to handle mul)ple users and large documents efficiently.

4.Advanced Features:
4.1Implement addi)onal features such as keyword highligh)ng, summariza)on, and mul)-document support.

5.Fine-tuning:
5.1Fine-tune the BERT model on a more specific domain dataset to improve accuracy for specialized applica)ons.

This project demonstrates the poten)al of combining modern NLP techniques with web development to create interac)ve and intelligent applica)ons. By addressing the areas for improvement, the system can be made even more robust and versa)le for various use cases.

References:

Books:
"Natural Language Processing with Transformers" by Lewis Tunstall, Leandro von Werra, and Thomas Wolf.

Ar)cles:
"Extrac)ng Text from PDFs Using Python" – Real Python.

Websites:
Hugging-Face-Transformers-Documenta)on:
hMps://huggingface.co/transformers/
PyMuPDF Documenta)on:

hMps://pymupdf.readthedocs.io/en/latest/

Thank You