PDF Answering AI Report (ArIES) – IITR

By – Rishi Chourasia 21112089

Purpose:

The aim of this project is to build a web-based question-answering system capable of extracting text from PDF files and answering questions based on the extracted content. This system will employ Flask for the web framework and the Hugging Face Transformers library for the question-answering model.

Model Selection:

We opted for the question-answering pipeline from the Hugging Face Transformers library for this project. This library offers state-of-the-art models for a range of natural language processing tasks. The chosen model is based on BERT (Bidirectional Encoder Representations from Transformers), a powerful transformer model pre-trained on an extensive text corpus.

Implementation Process:

1. Environment Setup:

- Navigate to Your Project Directory: Open your terminal and go to the project directory.

- Create a Virtual Environment: Use `python3 -m venv` to create a virtual environment named venv.

- Activate the Virtual Environment: Activate the virtual environment.

- Install Necessary Packages: Use pip to install the required Python packages.

2. Streamlit Script Creation and Development:

- Create a New Python Script: Create a new file named `streamlit\_app.py` in your project directory.

- Write the Script: Input the provided code into `streamlit\_app.py`.

3. Streamlit Application Execution:

- Run the Streamlit Command: Ensure your virtual environment is activated and run the Streamlit app with: `streamlit run streamlit\_app.py`.

- Open the Streamlit App in a Browser: Streamlit will start a local server and provide a URL (e.g., http://localhost:8501). Open this URL in your browser to access the app.

- Upload a PDF File: Use the file uploader in the Streamlit app to upload a PDF file.

- Submit a Question: Once the PDF text is extracted and displayed, enter a question in the provided text box. The app will use the question-answering model to find an answer based on the extracted text from the PDF.

PDF Text Extraction:

1. PyMuPDF Integration:

- Open and read the PDF file using PyMuPDF.

- Extract text from each page of the PDF and combine it into a single string.

2. Function Definition:

- Define `extract\_text\_from\_pdf(pdf\_path)` function to handle the text extraction.

Question Answering Pipeline:

1. Hugging Face Integration:

- Initialize the question-answering pipeline using the Transformers library.

- Define `answer\_question(question, context)` function to generate answers from the model.

2. AJAX Handling:

- Use AJAX to send the question and context to the Flask backend.

- Return the answer as a JSON response.

Model Training:

For this project, we utilize a pre-trained BERT model for the question-answering task. Fine-tuning a BERT model involves the following steps:

1. Data Collection:

- Gather a dataset with context-question-answer pairs, such as the SQuAD dataset.

2. Data Pre-processing:

- Tokenize the context and questions.

- Prepare the input data in a format compatible with BERT.

3. Training:

- Fine-tune the pre-trained BERT model on the dataset using the Hugging Face Transformers library.

- Adjust hyperparameters such as learning rate, batch size, and number of epochs for optimal performance.

4. Evaluation:

- Evaluate the model on a validation set to monitor performance.

- Use metrics such as Exact Match (EM) and F1 score to assess accuracy.

5. Deployment:

- Save the fine-tuned model and integrate it into the Flask application.

Key Learnings:

This project provided several key learning experiences:

1. Understanding NLP Models:

- Gained insights into transformer-based models like BERT and their application in NLP tasks.

2. Web Development:

- Learned how to create a web application using Flask.

- Understood the process of handling file uploads and processing user input.

3. PDF Text Extraction:

- Explored methods to extract text from PDF documents using PyMuPDF.

4. Model Integration:

- Successfully integrated a machine learning model into a web application.

- Dealt with challenges related to model inference and response handling in a web environment.

Final Thoughts:

The project effectively demonstrates the application of transformer models in real-world scenarios by creating a system that can extract text from PDF files and accurately answer user questions. This highlights the potential of advanced NLP models and suggests many opportunities for further development and application across various fields. The success of this project lies in its ability to merge sophisticated machine learning techniques with accessible web technologies, making advanced NLP tools available to a broader audience.

Future Enhancements:

Several areas for future improvement include:

1. User Interface:

- Enhance the user interface to make it more intuitive and user-friendly.

2. Performance Optimization:

- Optimize the text extraction and model inference processes for faster response times.

3. Scalability:

- Explore ways to scale the application to handle multiple users and large documents efficiently.

4. Advanced Features:

- Implement additional features such as keyword highlighting, summarization, and multi-document support.

5. Fine-tuning:

- Fine-tune the BERT model on a more specific domain dataset to improve accuracy for specialized applications.

This project demonstrates the potential of combining modern NLP techniques with web development to create interactive and intelligent applications. By addressing the areas for improvement, the system can be made even more robust and versatile for various use cases.

References:

Books:

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

Articles:

- "Extracting Text from PDFs Using Python" – Real Python.

Websites:

- Hugging Face Transformers Documentation: [Hugging Face](https://huggingface.co/transformers/)

- PyMuPDF Documentation: [PyMuPDF](https://pymupdf.readthedocs.io/en/latest/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Thank You

---

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ArIES - AI PDF Answering System Report - IITR

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Contents:

1. Objective

2. Model Selection

3. Implementation Steps

1. Setting Up the Environment

2. Creating and Writing the Streamlit Script

3. Running the Streamlit Application

4. Model Training Process

5. Key Learnings

6. Conclusion

7. Future Enhancements

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Objective:

The primary goal of this project is to develop a web-based question-answering system that can extract text from PDF documents and answer questions based on the extracted text. The system will be built using Flask for the web framework and the Hugging Face Transformers library for the question-answering model.

Model Selection:

For this project, the question-answering pipeline from the Hugging Face Transformers library was chosen. This library provides advanced models for natural language processing tasks. The specific model used in this pipeline is based on BERT (Bidirectional Encoder Representations from Transformers), a powerful transformer model pre-trained on a large corpus of text.

Advantages of BERT:

- High Accuracy: BERT performs exceptionally well on question-answering tasks.

- Pre-trained: Using a pre-trained model saves significant time and computational resources.

- Versatility: BERT can be fine-tuned for specific tasks with relatively little data.

- User-Friendly: The Hugging Face library offers a straightforward API for implementing the BERT model.

Implementation Steps:

1. Setting Up the Environment:

- Navigate to Project Directory: Open your terminal and navigate to your project directory.

- Create a Virtual Environment: Use `python3 -m venv` to create a virtual environment named venv.

- Activate the Virtual Environment: Activate the virtual environment.

- Install Required Packages: Use pip to install the necessary Python packages.

2. Creating and Writing the Streamlit Script:

- Create a New Python Script: Create a new file named `streamlit\_app.py` in your project directory.

- Write the Code: Enter the provided code into `streamlit\_app.py`.

3. Running the Streamlit Application:

- Run the Streamlit Command: Ensure your virtual environment is activated and run the Streamlit app using: `streamlit run streamlit\_app.py`.

- Open the Streamlit App in a Browser: Streamlit will start a local server and