Report on Pdf Answering AI (ArIES)-IITR

By
Shaksham Verma
21112098

Table of Content:

Objective:

The objective of this project is to create a web-based question answering system that can extract text from a PDF document and answer questions based on the extracted text. The system will use Flask for the web application framework and the Hugging Face Transformers library for the question-answering model.

Choice of Model:

For this project, we chose the question-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 (Bidirectional Encoder Representations 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 exceptionally well on question-answering tasks.
Pre-trained: Using a pre-trained model saves significant time and computational resources.
Flexibility: BERT can be fine-tuned for specific tasks with relatively little data.
Support: The Hugging Face library provides a user-friendly API for implementing 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. Activate the Virtual Environment: Activate 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 activated and run the Streamlit app using the following command: (streamlit run streamlit_app.py)
3.2. Open the Streamlit App in a Browser: After running the command, Streamlit will start a local server and provide you with a URL (e.g., http://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 Question: After the PDF text is extracted and displayed, input a question in the text box provided. The app will use the question-answering model to find the answer based on the extracted text from the PDF..

PDF Text Extraction:

1. PyMuPDF Integration:
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. Function Definition:
2.1 Define extract_text_from_pdf(pdf_path) function to perform the text extraction.

Question Answering Pipeline:

1. Hugging Face Integration:
1.1 Initialize the question-answering pipeline using the Transformers library.
1.2 Define answer_question (question, context) function to get answers from the model.

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

Model Training:

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

2. Data Pre-processing:
2.1 Tokenize the context and questions.
2.2 Prepare the input data in a format compatible 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 optimal performance.

4. Evaluation:
4.1 Evaluate the model on a validation 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 application.

Learning from This Project:

This project provided several key learning experiences:

1. Understanding NLP Models:

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

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

3. PDF Text Extraction:
3.1 Explored methods to extract text from PDF documents using PyMuPDF.

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

Conclusion:

Overall, the project demonstrates the practical application of transformer models in real-world scenarios by creating a system that can extract text from PDF files and provide accurate answers to user questions. This not only highlights the capabilities of advanced NLP models but also opens up numerous possibilities for future development and application in various domains. The success of this project lies in its ability to combine sophisticated 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:
1.1 Enhance the user interface to make it more intuitive and user-friendly.

2. Performance Optimization:
2.1 Optimize the text extraction and model inference processes for faster response times.

3. Scalability:
3.1 Explore ways to scale the application to handle multiple users and large documents efficiently.

4. Advanced Features:
4.1 Implement additional features such as keyword highlighting, summarization, and multi-document support.

5. Fine-tuning:
5.1 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:
https://huggingface.co/transformers/
PyMuPDF Documentation: https://pymupdf.readthedocs.io/en/latest/

Thank You