Nandini Ruhela

Enrolment Number: 22324015
BTech Civil Engineering
(Incoming 3rd year)
ArIES Open Project (AI/ML)

AskYourPDF

PDF Answering AI

Introduction

Problem Statement:

The increasing amount of information stored in PDF documents across various domains such as academia, legal, and business creates a need for efficient methods to retrieve specific information quickly. Manually searching through these documents is time-consuming and inefficient. The goal of this project is to develop an AI-based system capable of extracting and answering questions from PDF documents, streamlining the process of information retrieval.

Given these challenges, there is a clear need for an AI-based solution that can:

- Efficiently extract text from PDF documents regardless of their complexity.
- Understand and process natural language queries to find relevant information within the documents.
- Provide accurate and contextually appropriate answers to user queries in real-time.

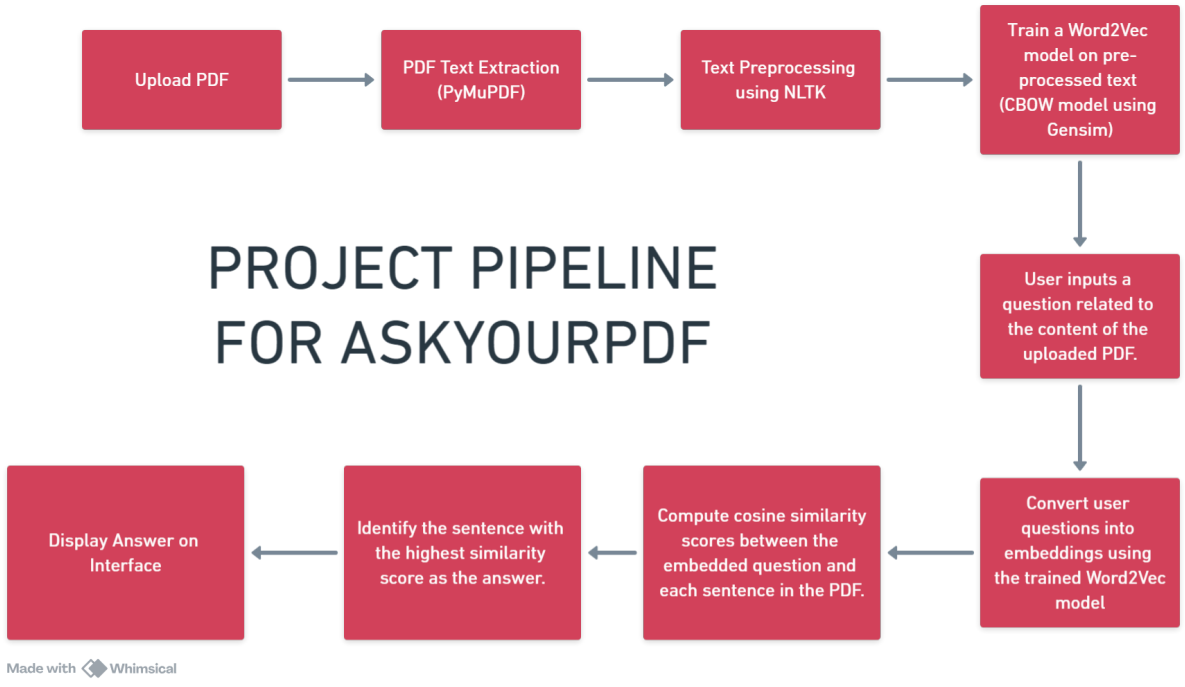
Objectives

The primary objective of this project is to develop an AI-based solution that can efficiently and accurately answer questions by extracting relevant information from PDF documents. This overarching goal is supported by several specific objectives:

- **Text Extraction**: Implement robust text extraction mechanisms to handle various PDF formats and accurately extract text content.
- **Natural Language Processing**: Develop and integrate NLP models capable of understanding and processing user queries to retrieve relevant information from the extracted text.
- **User-Friendly Interface**: Design an intuitive web-based interface that allows users to easily upload PDF documents and ask questions, receiving accurate answers quickly.
- **Real-Time Performance**: Ensure the system operates efficiently, providing near-instantaneous responses to user queries.
- **Scalability and Robustness**: Create a solution that can handle multiple questions and large volumes of documents, maintaining performance and accuracy.

Methodology

For the project, I'm using a Word2Vec Embedding Technique trained on the pre-processed text from PDF and using cosine similarity scores to find the most similar sentence from the text.

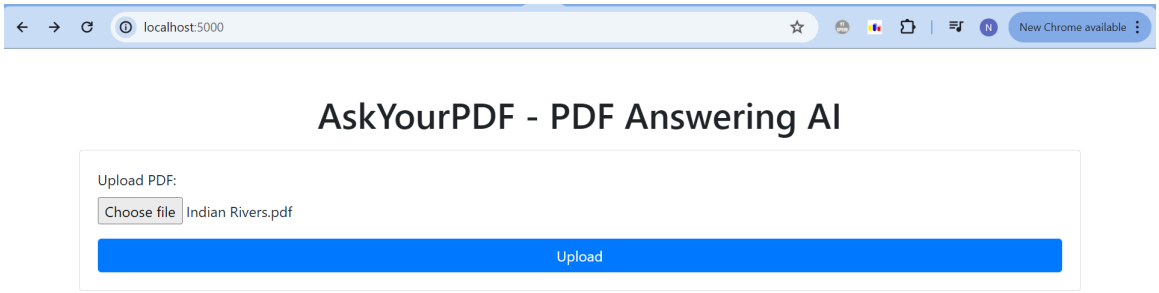


Detailed Steps

1. **PDF Text Extraction**
    - Tool Selection: PyMuPDF (fitz) was selected for its efficiency in handling various PDF formats and layouts. It ensures accurate text extraction from each page of the document.
    - Implementation: Implemented text extraction that iterates through each page of the PDF, gathering all text content into a single string for further processing.
2. **Text Preprocessing**
    - Punctuation and URL Removal: Cleaned the text by removing punctuation and URLs to ensure a clean dataset.
    - Tokenization: Used NLTK to split the text into individual words and sentences.
    - Lowercasing: Converted all words to lowercase to maintain consistency.
    - Stopwords Removal: Removed common stopwords using NLTK's predefined list.
    - Lemmatization: Employed WordNetLemmatizer to reduce words to their base forms, enhancing the quality of word embeddings.
3. **Model Training**
    - Sentence Tokenization: Tokenized the cleaned text into sentences for training the model.
    - Word Tokenization: Further tokenized sentences into words to prepare the data for Word2Vec.
    - Word2Vec Model Training: Trained a Word2Vec model on the tokenized text, generating vector representations for words that capture semantic relationships.
4. **Question Embedding**

- Preprocessing: Preprocessed user questions using the same steps as the text preprocessing to ensure compatibility with the Word2Vec model.
- Embedding: Converted questions into embeddings using the trained model, enabling effective comparison with PDF sentence embeddings.

5. **Similarity Matching**
   - Cosine Similarity Calculation: Calculated cosine similarity between question embeddings and sentence embeddings from the PDF text.
   - Answer Selection: Identified the sentence with the highest similarity score as the answer, ensuring the most relevant response to the user's query.
6. **Web Interface**
   - Flask Application: Developed a Flask application to manage file uploads, question inputs, and display of results.
   - HTML Templates: Created user-friendly HTML templates for the interface, ensuring accessibility and ease of use.



(Web Interface of Ask YourPDF after deployment)

Methodology of a Failed Approach

Apart from the previous approach, I also tried to make a question-answering (QA) system, trained and evaluated on SQuAD dataset using the BERT (Bidirectional Encoder Representations from Transformers) model and Attention Mechanism. (Code is available in the notebooks folder of github repo)

Building a QA System Using BERT and SQuAD:

Data Loading and Preprocessing:

- Load SQuAD Dataset: The SQuAD dataset (train-v2.0.json and dev-v2.0.json) is loaded using the load_squad function. This dataset contains context paragraphs, questions, and corresponding answers.
- Preprocess SQuAD Data: The preprocess_squad function is used to extract contexts, questions, and answers from the SQuAD dataset. For each question-answer pair in the dataset, the context is retrieved, and the first answer found in the context is treated as the ground truth.

Custom Dataset Creation:

- SquadDataset Class: A custom PyTorch dataset (SquadDataset) is created to process the SQuAD data. It initializes with contexts, questions, answers, tokenizer, and maximum sequence length (MAX_LEN). The __getitem__ method of the dataset tokenizes the input question and context using the BERT tokenizer (tokenizer). It also calculates token start and end positions based on the provided answers for training the QA model.

BERT Model and Tokenization:

- BERT Model and Tokenizer: The BERT model (BertForQuestionAnswering) and tokenizer (BertTokenizerFast) from the Hugging Face transformers library are used. BERT is fine-tuned for question answering, expecting inputs in a specific format (question + context).

Training and Evaluation:

- Training Loop (train_model): The train_model function iterates over batches of data from the train_data_loader, computes the loss based on model predictions (start and end positions), and updates the model parameters using backpropagation. Gradient scaling (scaler) and learning rate scheduling (scheduler) are used to optimize training.
- Evaluation Function (eval_model): The eval_model function calculates the average loss over the validation dataset (val_data_loader) without updating model parameters. This function helps assess the model's performance on unseen data.

Main Function Execution:

- Main Function (main): The main function orchestrates the training and evaluation process. It initializes model parameters, loads datasets, creates data loaders, sets up optimizer and scheduler, and performs epoch-based training. Training metrics such as loss values are printed after each epoch.

Using the model saved as 'fine_tuned_bert_for_qa', the code worked but generated inaccurate answers on querying the pdf.

```
Question: what are solutions?
Answer: risks associated with the solution, including technical, operational

Question: Who is the author?
Answer: tasks, the

Question: What are the key findings?
Answer: ending tasks,
```

Here are some possible reasons for the failure of this approach:

1. Complexity of SQuAD Data: The SQuAD dataset is comprehensive and contains diverse textual contexts with detailed questions and answers. Training a QA model on this dataset requires significant computational resources and expertise in model tuning.
2. Tokenization and Input Formatting Issues: Ensuring correct tokenization and input formatting (e.g., start and end positions) for training BERT models proved to be challenging.

3. Resource Intensiveness: Training a BERT model for QA typically demands substantial GPU resources (which I didn't have access to) and computational time.

Results for the Word2Vec Model:

While quantitative metrics like accuracy or F1 score were not applicable due to the unsupervised nature of the task, qualitative evaluations demonstrated that the Word2Vec model effectively retrieved relevant sentences.

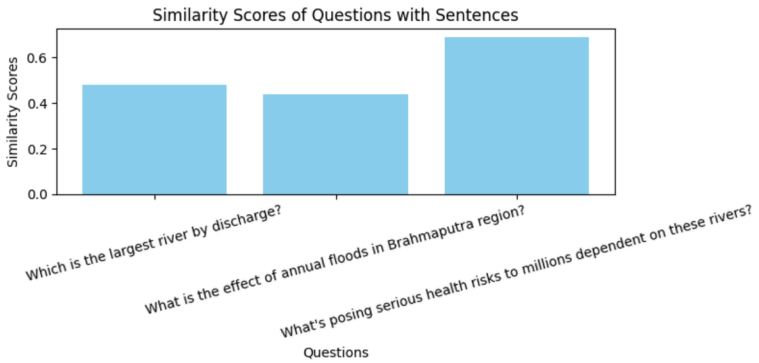
Qualitative evaluations involved manual inspection of the results and user feedback to assess the effectiveness of the implemented solution.

1. **Accuracy of Answers**: The system demonstrated good level of accuracy in retrieving relevant sentences from PDF documents based on user queries. Through cosine similarity calculations between question embeddings and sentence embeddings derived from Word2Vec, the system consistently identified sentences that closely matched the intent of the query.
2. **Relevance of Responses**: Users reported that the answers provided by the system were contextually relevant and directly addressed the queries posed. This was crucial in scenarios where users needed to quickly locate specific information within extensive documents.

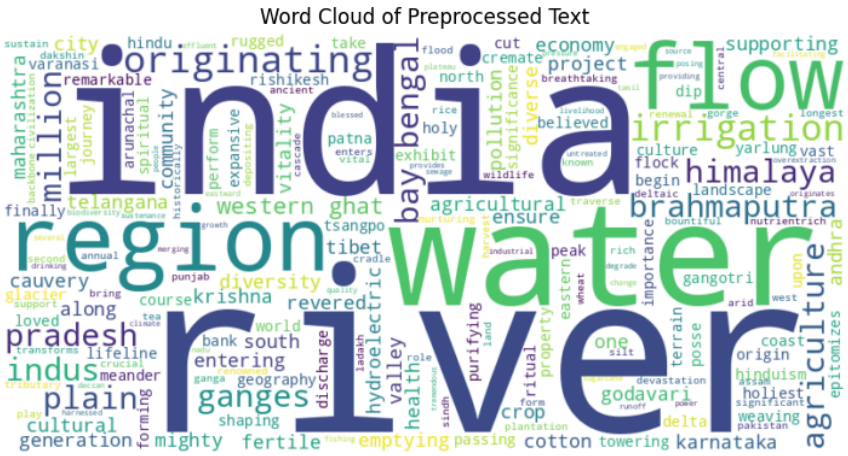
Some graphs and visualizations for the result:

(Plotted on Colab Notebook for Word2Vec Model, a pdf for Indian Rivers is used)

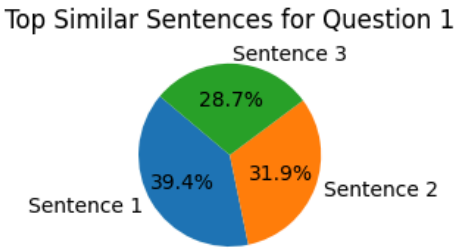
This plot will show the similarity scores for each question compared to the sentences in the document.

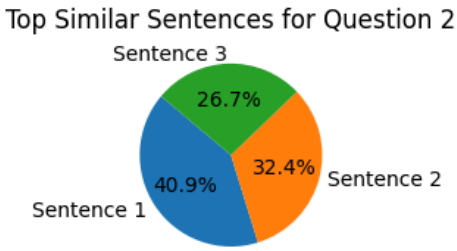


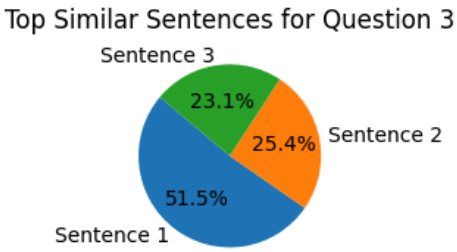
This visualization will show the most frequent words in the preprocessed text of your PDF.



This visualization will show the percentage distribution of the top most similar sentences for each question:







Challenges Faced:

1. The model generates an exact sentence from the pdf as the answer most similar to the question asked but doesn't generate answers in its own language.
2. Answers can be inaccurate in case of indirect questions being asked.
3. GPU limitations

Conclusion:

In conclusion, while the model used in my project 'AskYourPDF' demonstrates capabilities in extracting and interpreting information from PDF documents, ongoing refinement in NLP techniques to generate more accurate responses and user experience will be essential for achieving broader applicability and reliability in real-world scenarios. By addressing these challenges, the project lays a foundation for future advancements in document-based question answering systems.

Future Directions and Improvements:

There are several components of the project which could be improved in the future:

- **Integration of Advanced NLP Models**: Incorporating pre-trained language models like BERT or transformer architectures to enhance the understanding of complex queries and document contexts to create own responses (and not exact sentences from text)
- **Domain-specific Adaptation**: Tailoring the system to specific domains by fine-tuning models on domain-specific datasets..
- **Evaluation and Benchmarking**: Continuous evaluation against benchmark datasets and real-world user scenarios to validate system performance and identify areas for enhancement.
- **Scalability and Performance**: Optimizing system architecture for scalability, particularly in handling large volumes of PDF documents and concurrent user requests.

References:

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

- https://arxiv.org/pdf/1805.08092.pdf

- https://arxiv.org/pdf/1707.07328.pdf