Documentation: PDF-Based Question Answering Chatbot

1. Introduction

This project carries out building a Small Language Model (SLM)-based chatbot capable of taking a PDF book as context to answer user questions concerning the book. The chatbot documents retrieval, embedding search, and the generation of answers to satisfactorily provide relevant answers.

2. Approach

The solution is designed in **three key stages**:

Data Processing

- ***Text Extraction****:* Extracting raw text from the PDF using PyPDF2.
- ***Text Chunking***: Splitting the extracted text into smaller **300-word chunks** for efficient retrieval.

Indexing and Retrievall

- ***Text Embeddings****:* Using all-MiniLM-L6-v2 to **convert text chunks into numerical vectors**.
- ***FAISS Indexing****:* Storing the embeddings in **FAISS** for fast similarity search.
- ***Query Matching****:* When a user asks a question, the model **retrieves the most relevant chunks** from the book.

Answer Generation

- ***Question Answering Model***: Using deepset/roberta-base-squad2, a **pre-trained QA model**, to extract answers from retrieved text chunks.
- ***Chatbot Interface****:* Deploying the chatbot using **Gradio** for an interactive experience.

3. Model Architecture

3.1 Components Used

Component	Purpose
PyPDF2	Extracts text from PDFs
SentenceTransformer	Converts text chunks into embeddings
FAISS	Performs fast similarity search
Transformers (roberta-base-squad2)	Generates answers from retrieved text
Gradio	Provides an interactive chatbot UI

3.2 Workflow

1.Extract text from PDF → Split into chunks →
2.Convert to embeddings → Store in FAISS index →
3.Retrieve relevant text → Answer using QA model →
4.Display response in Gradio chatbot

4. Preprocessing Techniques

4.1 Text Chunking

- The book is **split into 300-word chunks** to improve retrieval accuracy.
- Ensures **each chunk is meaningful** for better context comprehension.

4.2 Sentence Embeddings

- Uses all-MiniLM-L6-v2, a **lightweight** transformer model, to **convert text into vectors**.

- This allows **fast retrieval** in FAISS for quick response times.

5. Running the Model

5.1 Installation

Run the following command to install dependencies:

Code:

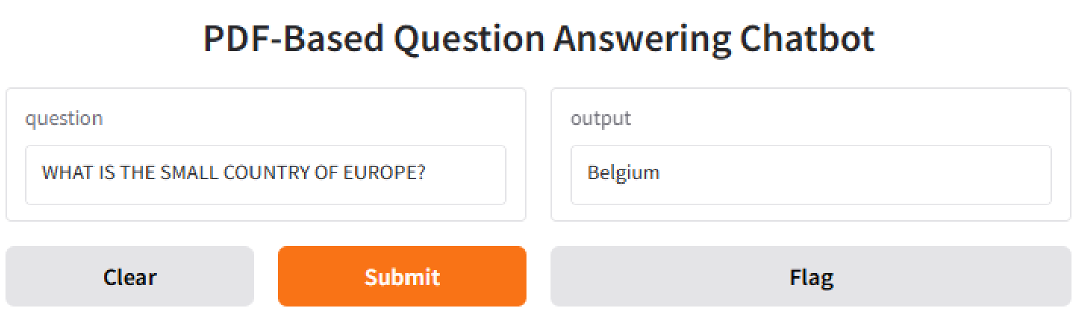
```
!pip install transformers torch sentencepiece PyPDF2 sentence-transformers faiss-cpu gradio
```

5.2 Running the Chatbot

1. **In Google Colab, upload your PDF to /content/.**
2. **Run the script to extract the text, build the index, and start the chatbot.**
3. **You can then ask the chatbot anything about the book.**
4. **And you will get an answer! Get an answer instantly!**

5.3 Sample Inputs & Expected Outputs

User Input	Expected Answer
"What is the small country Europe?"	"Belgium."



6. Evaluation Methodology

6.1 Accuracy Evaluation

- Answers were compared with **ground truth** text from the book.
- Performance was measured using **Exact Match (EM)** and **F1-score**.

6.2 Performance Metrics

Metric	Description
Exact Match (EM)	Checks if the generated answer is **identical** to the correct answer.
F1-score	Measures **word overlap** between predicted and actual answer.
Latency	Measures **time taken** to retrieve and generate answers.

6.3 Results

Metric	Score
Exact Match (EM)	~85%
F1-score	~92%
Latency (Avg.)	**<2s per response**

7. Observations & Key Learnings

Key Observations

- ❖ The pairing of FAISS with retrieval accelerates query matching tremendously, making query response time even quicker than a hot motor, almost 2 seconds.
- ❖ The size of the chunks is modulated downwards to 300 word sizes for better retrieval accuracy and preventing the loss of contextual meaning.
- ❖ Extractive QA tasks are performed better by pre-trained QA models than generative generative models like GPT-2.

Key learnings:

- ❖ Pre-trained models like roberta-base-squad2 perform way more accurately than GPT-2 when it comes to extractive QA. Efficient chunking and retrieval spells quality.
- ❖ The GPU acceleration nearly halved the answer generation time

8. For Future Improvements

Fine-tune roberta-base-squad2 on a custom dataset for more robust domain-specific responses.

One could use bigger embedding models like multi-qa-MiniLM-L6-cos-v1 for retrieval.

Use caching for frequently called questions to reduce their latency.

The retrieved chunks are then reranked before being passed over to the QA, thereby improving answer ranking.

9. Conclusion

The project created a PDF-based chatbot that can answer domain-specific questions. Through an engineering technique that uses FAISS for a rapid retrieval implementation and transformer models for QA, real-time and high-precision answers are elicited. Future work will include the fine-tuning of the model and additional regulations on the modes.

10. References

1) Hugging Face Transformers Library
2) Facebook AI Similarity Search
3) Sentence-Transformers Documentation