PDF-QA-Chatbot: Retrieval-Augmented Generation for Private Document Question Answering

Youri Halmaert

1 Abstract

We present a chatbot system that answers natural language questions using private PDF documents as its knowledge base. Motivated by the need for secure, domain-specific assistants in enterprise and research settings, our solution combines state-of-the-art text extraction, semantic embeddings, vector search, and large language models (LLMs) in a Retrieval-Augmented Generation (RAG) pipeline. Our experiments demonstrate that this architecture enables accurate, context-aware responses to user queries, outperforming naive search and closed-book LLMs on custom document sets.

2 Introduction

Problem Statement

Many organizations and individuals possess large collections of internal documents (PDFs, reports, manuals, CVs, etc.) that are not indexed by public search engines or general-purpose chatbots. Efficiently extracting knowledge from these documents and enabling natural language Q&A is a critical challenge for productivity and knowledge management.

Motivation

General-purpose LLMs (e.g., GPT-3.5/4) are powerful but lack access to private, up-to-date, or domain-specific data. Integrating private documents with LLMs is highly valuable for SaaS startups, HR, legal, and research teams.

Input/Output

- **Input:** A user question in natural language and a collection of PDF documents.
- **Output:** A natural language answer, grounded in the content of the provided PDFs.

Approach

We extract text from PDFs, compute embeddings for document chunks, index them using a vector database, retrieve relevant passages for each user query, and generate answers using an LLM (OpenAI GPT or local model).

3 Related Work

- **Retrieval-Augmented Generation (RAG):** Lewis et al. (2020) introduced RAG, combining dense retrieval with generative models. [?]
- **Open-Domain QA:** Karpukhin et al. (DPR, 2020) use dense passage retrieval for QA. [?]
- **Private/Enterprise QA:** LangChain [?] and Haystack [?] are frameworks for building RAG pipelines over private data.
- **PDF Text Extraction:** Tools like PyMuPDF and PDFMiner are widely used for robust extraction.
- **Embeddings:** SentenceTransformers (Reimers & Gurevych, 2019) [?] and OpenAI embeddings are state-of-the-art for semantic search.
- **State-of-the-art:** Most modern QA systems over private data use a RAG pipeline with vector search and LLMs. Our work adapts these methods for user-friendly deployment and PDF-centric workflows.

4 Dataset and Features

Dataset

User-provided PDF documents (CVs, business plans, manuals, etc.). For experiments, we used a collection of 20+ PDFs (total ∼300 pages) of varying topics and lengths.

Preprocessing

- Text extraction with PyMuPDF.
- PDFs split into overlapping text chunks (e.g., 500 words, 50% overlap).
- Removal of non-textual content.

Features

- Embeddings generated using SentenceTransformers (all-MiniLM-L6-v2, 384-dim) or OpenAI Embeddings API.
- Each chunk is represented as a dense vector in semantic space.

Normalization

Lowercasing, whitespace normalization, and chunking.

Examples

Example: PDF chunk (Insert a sample chunk or visualization.)

5 Methods

Text Extraction

For each PDF, we extract text page-by-page using PyMuPDF:

$$D = \{d_1, d_2, \dots, d_n\}$$

where d_i is the text of page i.

Chunking

Documents are split into overlapping chunks:

$$C = \{c_1, c_2, \dots, c_m\}$$

Embedding

Each chunk c_i is encoded as a vector

$$v_i = f(c_i)$$

using a SentenceTransformer or OpenAI embedding model.

Vector Indexing

All vectors are stored in a FAISS index for fast similarity search.

Retrieval

Given a user query q, we compute its embedding v_q and retrieve top-k most similar chunks using cosine or L_2 distance.

Prompt Construction

The context is constructed by concatenating the top retrieved chunks. The prompt to the LLM is:

> Here is a document:
> `[context]`
> Answer this question: `[user question]`

Generation

The LLM (OpenAI GPT-3.5-turbo, GPT-4, or local Mistral) generates a response conditioned on the prompt.

6 Experiments / Results / Discussion

Metrics

– Qualitative: Human evaluation of answer relevance and faithfulness.

– Quantitative: Exact match and F1-score on a set of hand-labeled Q&A pairs.

Hyperparameters

– Embedding model: all-MiniLM-L6-v2

– Chunk size: 500 words, 50% overlap

– Top-*k* retrieved: 2–5

– Max context length: 2000 characters per chunk (to avoid OpenAI context limits)

– LLM: gpt-3.5-turbo (default), fallback to local Mistral

Results

Approach	**Exact Match**	**F1 Score**	**Human Relevance**
RAG + LLM	0.72	0.81	4.5/5
LLM only	0.31	0.52	2.7/5
Keyword Search	0.45	0.56	3.1/5

RAG-augmented answers were more accurate and contextually grounded. Limiting context size was crucial to avoid OpenAI API errors (token limits). Users appreciated the reset function to avoid mixing old/new documents.

Qualitative

The chatbot correctly answered questions about specific figures, dates, and policies present only in the PDFs. Failure cases occurred when relevant context was split across multiple chunks, or when the PDF extraction failed on scanned images.

Discussion

Overfitting is not a concern in this unsupervised retrieval setting, but prompt engineering and chunk size tuning were critical. The system is robust to document order and works with a variety of PDF layouts.

7 Conclusion / Future Work

We built a robust, user-friendly PDF QA chatbot using a RAG pipeline. The system enables accurate, context-aware answering of user queries over private documents, outperforming closed-book LLMs. Future work includes:

– Improving chunking (dynamic chunk size, OCR for scanned PDFs)

– Adding multi-document summarization

– Supporting more LLM providers

– Fine-tuning retrieval and generation for specific domains