DEEP LEARNING

DOCUMIND-AI

DR. AYMAN HELMY
ENG.MOHAMMED BAHGAT

ADHAM HANY	221017671
NOUR-ELDIN AHMED	221017962
MOSTAFA FARIED	221027839
AHMED REDA	221027533

CONTENTS

WHAT IS DOCUMIND-AI?

Project Title: DocuMindAI - Intelligent Document Q&A System
Objective: Build a Retrieval-Augmented Generation (RAG) system that allows users to chat with PDF documents using a fine-tuned Deep Learning model. Tech Stack: Python, Hugging Face Transformers (DistilBERT), FAISS (Vector DB), Streamlit (GUI), Google Colab.

WHAT'S THE PROBLEM?

The Problem: "The Static Document Crisis"

1. Information Overload We live in an age where knowledge is locked inside millions of digital documents. Students have massive textbooks, researchers have endless papers, and companies have thousands of contracts and reports.

WHAT'S THE PROBLEM?

2. The Limitation of Standard Search Currently, the only way to find information in these PDFs is by using "Ctrl+F" (Keyword Search).

- **It is "Dumb":** If you search for "Global Warming," it won't find paragraphs that talk about "Climate Change" unless the exact words match.
- **No Context:** It cannot answer questions like "What are the main causes of the failure?" or "Summarize the third chapter." It only highlights words; it doesn't understand meaning.

WHAT'S THE PROBLEM?

3. The Result: Wasted Productivity Professionals and students waste countless hours scrolling through hundreds of pages to find a single specific answer. The information is there, but accessing it is slow, manual, and inefficient. We have "smart" phones but we are still interacting with "dumb" documents.

WHAT OUR MODEL OFFERS!

1. Intelligent Q&A (Beyond Keywords) Unlike traditional tools that only find matching words (Ctrl+F), our model understands natural language. You can ask, "What are the main risks mentioned in Chapter 3?" and it will generate a coherent, human-like answer.

2. Contextual Understanding Powered by a fine-tuned DistilBERT, the model doesn't just read text; it understands context. It can distinguish between similar terms based on how they are used in a sentence, ensuring high accuracy.

3. Instant Information Retrieval We utilize FAISS (Facebook AI Similarity Search) to index documents. This allows the system to scan through hundreds of pages and retrieve the exact paragraph needed in milliseconds, saving the user hours of manual searching.

4. User-Friendly Interaction We provide a clean, "SaaS-style" web interface (built with Streamlit). Users don't need to know any code; they simply drag and drop a PDF and start chatting immediately.

5. Smart Document Summarization The system can synthesize information from multiple parts of a document to provide concise summaries, helping users grasp complex technical or legal documents quickly.

WHAT OUR MODEL OFFERS!

PROJECTS CREATORS

Adham Hany

Fine-tuned the DistilBERT model on the SQuAD dataset, teaching the AI how to understand and answer questions.

Nour Ahmed

Built the retrieval pipeline using FAISS to index PDF documents and find relevant answers in milliseconds.

Mostafa Faried

Designed the professional Streamlit interface, creating the "DocuMind" visual identity and user experience.

Ahmed Reda

Optimized performance with caching and deployed the project live to the web using Ngrok.

OUR MODEL!

```
[12] 33s
!pip install -q transformers datasets accelerate torch faiss-cpu sentence-transformers PyPDF2 streamlit pyngrok
!npm install localtunnel
```

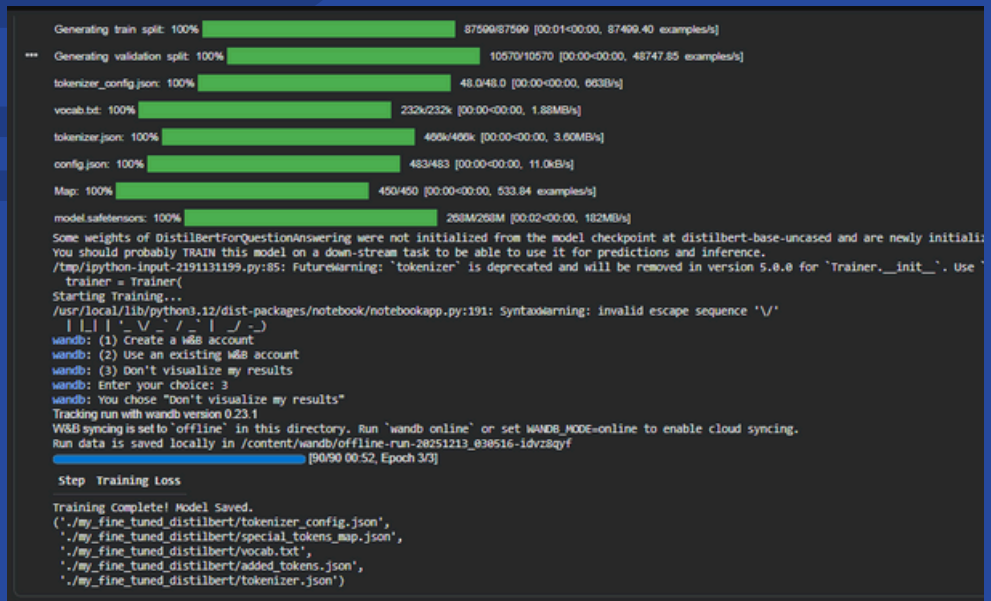
```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 56.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 12.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.0/9.0 MB 62.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 71.1 MB/s eta 0:00:00
added 22 packages in 5s
3 packages are looking for funding
  run `npm fund` for details
```

```
[13] 3m
import torch
from datasets import load_dataset
from transformers import DistilBertTokenizerFast, DistilBertForQuestionAnswering, Trainer, TrainingArguments

# 1. Load Dataset (SQuAD - Stanford Question Answering Dataset)
print("Loading dataset...")
dataset = load_dataset("squad", split="train[:500]")  # Using small subset for speed
train_test_split = dataset.train_test_split(test_size=0.1)
train_dataset = train_test_split["train"]
eval_dataset = train_test_split["test"]

# 2. Preprocessing
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')

def prepare_train_features(examples):
    # Tokenize questions and contexts
    tokenized_examples = tokenizer(
        examples["question"],
        examples["context"],
        truncation="only_second",
        max_length=384,
        stride=128,
        return_overflowing_tokens=True,
        return_offsets_mapping=True,
        padding="max_length",
    )

    sample_mapping = tokenized_examples.pop("overflow_to_sample_mapping")
    offset_mapping = tokenized_examples.pop("offset_mapping")

    tokenized_examples["start_positions"] = []
    tokenized_examples["end_positions"] = []

    for i, offsets in enumerate(offset_mapping):
        input_ids = tokenized_examples["input_ids"][i]
        cls_index = input_ids.index(tokenizer.cls_token_id)
        sequence_ids = tokenized_examples.sequence_ids(i)
        sample_index = sample_mapping[i]
        answers = examples["answers"][sample_index]

        if len(answers["answer_start"]) == 0:
            tokenized_examples["start_positions"].append(cls_index)
            tokenized_examples["end_positions"].append(cls_index)
        else:
            start_char = answers["answer_start"][0]
            end_char = start_char + len(answers["text"][0])
            token_start_index = 0
            while sequence_ids[token_start_index] != 1:
                token_start_index += 1
            token_end_index = len(input_ids) - 1
```

```
[13] 3m
            token_end_index = len(input_ids) - 1
            while sequence_ids[token_end_index] != 1:
                token_end_index -= 1

            if not (offsets[token_start_index][0] <= start_char and offsets[token_end_index][1] >= end_char):
                tokenized_examples["start_positions"].append(cls_index)
                tokenized_examples["end_positions"].append(cls_index)
            else:
                while token_start_index < len(offsets) and offsets[token_start_index][0] <= start_char:
                    token_start_index += 1
                tokenized_examples["start_positions"].append(token_start_index - 1)
                while offsets[token_end_index][1] >= end_char:
                    token_end_index -= 1
                tokenized_examples["end_positions"].append(token_end_index + 1)

    return tokenized_examples

tokenized_datasets = train_dataset.map(prepare_train_features, batched=True, remove_columns=train_dataset.column_names)

# 3. Define Model
model = DistilBertForQuestionAnswering.from_pretrained("distilbert-base-uncased")

# 4. Training Arguments
args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,  # Increase epochs if needed
    per_device_train_batch_size=16,
    learning_rate=2e-5,
    weight_decay=0.01,
    logging_dir='./logs',  # Optional: helps with logging
    logging_steps=500,     # Optional: number of steps to log
    save_steps=500,
)

# 5. Trainer
trainer = Trainer(
    model=model,
    args=args,
    train_dataset=tokenized_datasets,
    tokenizer=tokenizer,
)

print("Starting Training...")
trainer.train()
print("Training Complete! Model Saved.")

# Save the fine-tuned model
model.save_pretrained("./my_fine_tuned_distilbert")
tokenizer.save_pretrained("./my_fine_tuned_distilbert")
```

```
Loading dataset...
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as se
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
README.md: 7.62k/? [00:00<00:00, 203kB/s]
plain_text/train-00000-of-00001.parquet: 100% 14.5M/14.5M [00:00<00:00, 17.7MB/s]
plain_text/validation-00000-of-00001.par(...): 100% 1.82M/1.82M [00:00<00:00, 7.13MB/s]
Generating train split: 100% 87599/87599 [00:01<00:00, 87499.40 examples/s]
Generating validation split: 100% 10570/10570 [00:00<00:00, 48747.85 examples/s]
```

OUR MODEL!



```
Generating train split: 100% 87599/87599 [00:01<00:00, 87499.40 examples/s]
Generating validation split: 100% 10570/10570 [00:00<00:00, 48747.85 examples/s]
tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 663B/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 1.88MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 3.60MB/s]
config.json: 100% 483/483 [00:00<00:00, 11.0kB/s]
Map: 100% 450/450 [00:00<00:00, 533.84 examples/s]
model.safetensors: 100% 268M/268M [00:02<00:00, 182MB/s]
Some weights of DistilBertForQuestionAnswering were not initialized from the model checkpoint at distilbert-base-uncased and are newly initiali
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/tmp/ipython-input-2191131199.py:85: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `
  trainer = Trainer(
Starting Training...
/usr/local/lib/python3.12/dist-packages/notebook/notebookapp.py:191: SyntaxWarning: invalid escape sequence '\/'
  | |_| | '_ \/ _` / _` |  _/ -_)
wandb: (1) Create a W&B account
wandb: (2) Use an existing W&B account
wandb: (3) Don't visualize my results
wandb: Enter your choice: 3
wandb: You chose "Don't visualize my results"
Tracking run with wandb version 0.23.1
W&B syncing is set to `offline` in this directory. Run `wandb online` or set WANDB_MODE=online to enable cloud syncing.
Run data is saved locally in /content/wandb/offline-run-20251213_030516-idvz8qyf
[90/90 00:52, Epoch 3/3]
 Step  Training Loss

Training Complete! Model Saved.
('./my_fine_tuned_distilbert/tokenizer_config.json',
 './my_fine_tuned_distilbert/special_tokens_map.json',
 './my_fine_tuned_distilbert/vocab.txt',
 './my_fine_tuned_distilbert/added_tokens.json',
 './my_fine_tuned_distilbert/tokenizer.json')
```

```
!apt-get install -y libfaiss-dev
!pip install faiss-cpu
!apt-get install -y libfaiss-dev
import faiss
import numpy as np
from sentence_transformers import SentenceTransformer
from transformers import pipeline
from PyPDF2 import PdfReader

# 1. Initialize Components
embedder = SentenceTransformer('all-MiniLM-L6-v2')  # Sentence Embedding Model
qa_pipeline = pipeline("question-answering", model="./my_fine_tuned_distilbert", tokenizer="./my_fine_tuned_distilbert")

def process_pdf(pdf_file):
    """Extract text from PDF and chunk it."""
    reader = PdfReader(pdf_file)
    text = ""
    for page in reader.pages:
        text += page.extract_text()

    # Simple chunking by splitting on newlines or length
    chunks = [text[i:i+500] for i in range(0, len(text), 500)]
    return chunks

def create_faiss_index(chunks):
    """Create FAISS index from text chunks."""
    embeddings = embedder.encode(chunks)
    d = embeddings.shape[1]
    index = faiss.IndexFlatL2(d)
```

```
[14]
✓ 33s
    # Simple chunking by splitting on newlines or length
    chunks = [text[i:i+500] for i in range(0, len(text), 500)]
    return chunks

def create_faiss_index(chunks):
    """Create FAISS index from text chunks."""
    embeddings = embedder.encode(chunks)
    d = embeddings.shape[1]
    index = faiss.IndexFlatL2(d)
    index.add(np.array(embeddings))
    return index, chunks

def get_answer(question, index, chunks):
    """Retrieve context and generate answer."""
    # 1. Retrieve
    q_embedding = embedder.encode([question])
    D, I = index.search(np.array(q_embedding), k=5)  # Get top 3 relevant chunks

    context = " ".join([chunks[i] for i in I[0]])

    # 2. Read (Answer Extraction)
    result = qa_pipeline(question=question, context=context)
    return result['answer'], context
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  libfaiss-dev
0 upgraded, 1 newly installed, 0 to remove and 41 not upgraded.
Need to get 949 kB of archives.
After this operation, 6,224 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfaiss-dev amd64 1.7.2-5 [949 kB]
Fetched 949 kB in 1s (1,676 kB/s)
Selecting previously unselected package libfaiss-dev:amd64.
(Reading database ... 121689 files and directories currently installed.)
Preparing to unpack .../libfaiss-dev_1.7.2-5_amd64.deb ...
Unpacking libfaiss-dev:amd64 (1.7.2-5) ...
Setting up libfaiss-dev:amd64 (1.7.2-5) ...
Requirement already satisfied: faiss-cpu in /usr/local/lib/python3.12/dist-packages (1.13.1)
Requirement already satisfied: numpy<3.0,>=1.25.0 in /usr/local/lib/python3.12/dist-packages (from faiss-cpu) (2.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from faiss-cpu) (25.0)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libfaiss-dev is already the newest version (1.7.2-5).
0 upgraded, 0 newly installed, 0 to remove and 41 not upgraded.
modules.json: 100% 340/340 [00:00<00:00, 11.9kB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 3.25kB/s]
README.md: 10.5k/? [00:00<00:00, 238kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 1.48kB/s]
config.json: 100% 612/612 [00:00<00:00, 15.7kB/s]
model.safetensors: 100% 90.9M/90.9M [00:01<00:00, 79.9MB/s]
tokenizer_config.json: 100% 350/350 [00:00<00:00, 11.0kB/s]
vocab.txt: 232k/? [00:00<00:00, 5.38MB/s]
tokenizer.json: 466k/? [00:00<00:00, 8.25MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 5.92kB/s]
config.json: 100% 190/190 [00:00<00:00, 3.44kB/s]
Device set to use cuda:0
```

OUR MODEL!

```
special_tokens_map.json: 100% 112/112 [00:00<00:00, 5.92kB/s]
config.json: 100% 190/190 [00:00<00:00, 3.44kB/s]
Device set to use cuda:0
```

```
[15]
```

```
%%writefile app.py
import streamlit as st
import faiss
import numpy as np
from sentence_transformers import SentenceTransformer
from transformers import pipeline
from PyPDF2 import PdfReader
import os

# --- Configuration ---
st.set_page_config(page_title="Document Q&A System", layout="centered")

# --- Load Models (Cached) ---
@st.cache_resource
def load_models():
    embedder = SentenceTransformer('all-MiniLM-L6-v2')
    qa_pipeline = pipeline("question-answering", model="bert-large-uncased-whole-word-masking-finetuned-squad", tokenizer="bert-large-uncased")

    return embedder, qa_pipeline

embedder, qa_pipeline = load_models()

# --- Helpers ---
def process_pdf(file):
    reader = PdfReader(file)
    text = ""
    for page in reader.pages:
        text += page.extract_text() or ""
    # Use larger chunks
    chunks = [text[i:i+1000] for i in range(0, len(text), 1000)]  # Bigger chunks
    return chunks

# --- UI ---
st.markdown("<h1 style='text-align: center; color: #2c3e50;'>🧠 Document Q&A System (RAG)</h1>", unsafe_allow_html=True)

if 'index' not in st.session_state:
    st.session_state.index = None
if 'chunks' not in st.session_state:
    st.session_state.chunks = []

with st.container():
    st.markdown("### 📄 Document Source")
    uploaded_file = st.file_uploader("Upload your PDF", type=['pdf'])

    if uploaded_file and not st.session_state.index:
        with st.spinner("Processing & Indexing..."):
            chunks = process_pdf(uploaded_file)
            embeddings = embedder.encode(chunks)
            index = faiss.IndexFlatL2(embeddings.shape[1])
            index.add(np.array(embeddings))

            st.session_state.index = index
            st.session_state.chunks = chunks
            st.success("Document Indexed Successfully!")

with st.container():
    st.markdown("### 💬 Q&A Interface")
    query = st.chat_input("Ask a question about the document...")

    if query:
        with st.chat_message("user"):
```

```
[15]
```

```
if 'chunks' not in st.session_state:
    st.session_state.chunks = []

with st.container():
    st.markdown("### 📄 Document Source")
    uploaded_file = st.file_uploader("Upload your PDF", type=['pdf'])

    if uploaded_file and not st.session_state.index:
        with st.spinner("Processing & Indexing..."):
            chunks = process_pdf(uploaded_file)
            embeddings = embedder.encode(chunks)
            index = faiss.IndexFlatL2(embeddings.shape[1])
            index.add(np.array(embeddings))

            st.session_state.index = index
            st.session_state.chunks = chunks
            st.success("Document Indexed Successfully!")

with st.container():
    st.markdown("### 💬 Q&A Interface")
    query = st.chat_input("Ask a question about the document...")

    if query:
        with st.chat_message("user"):
            st.write(query)

        if st.session_state.index:
            # RAG Retrieval
            q_embed = embedder.encode([query])
            D, I = st.session_state.index.search(np.array(q_embed), k=3)
            context = " ".join([st.session_state.chunks[i] for i in I[0]])

            # AI Answer
            result = qa_pipeline(question=query, context=context)

            with st.chat_message("assistant"):
                st.write(result['answer'])
                with st.expander("View Context"):
                    st.write(context)
        else:
            st.error("Please upload a document first.")
```

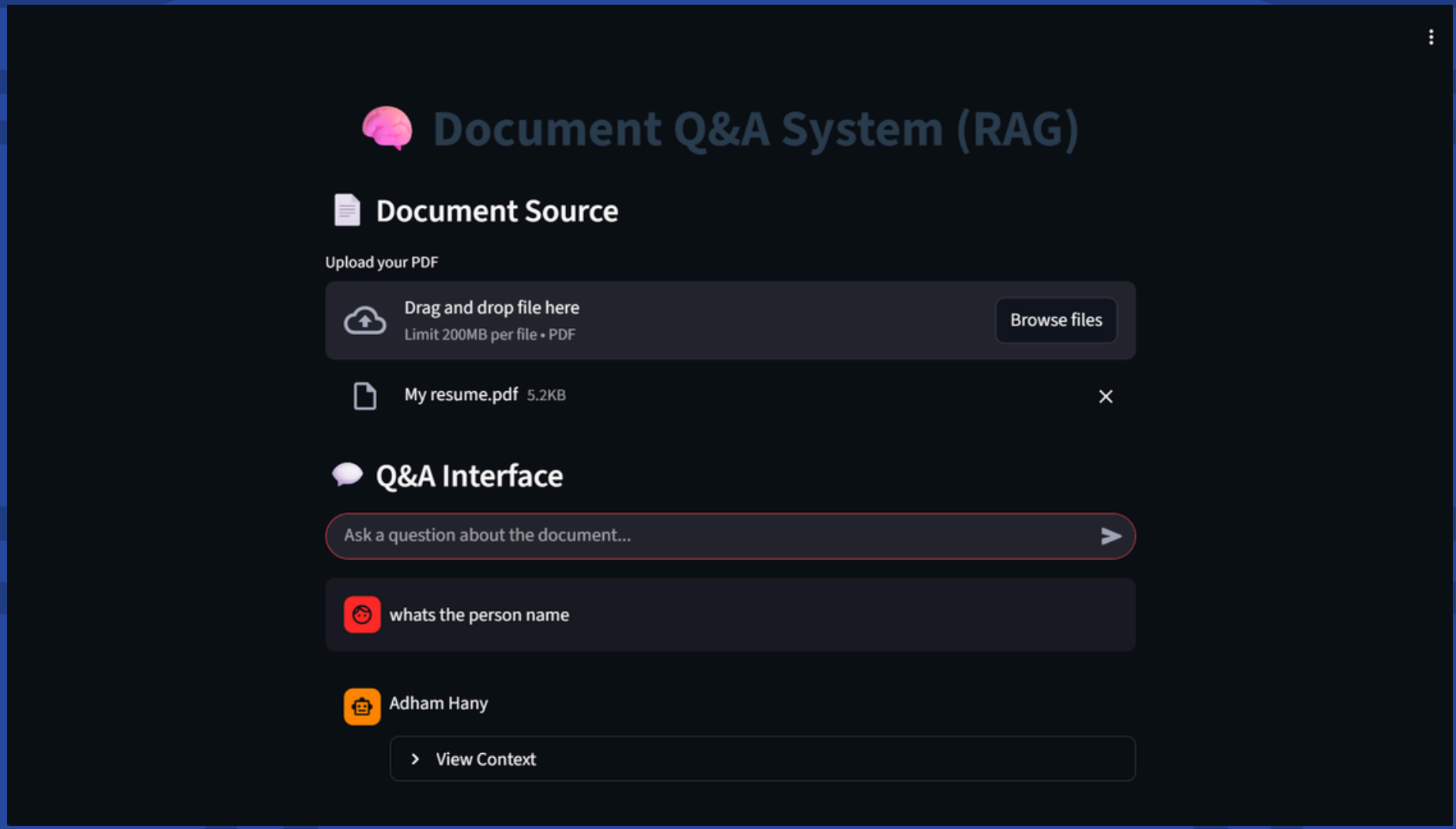
```
Overwriting app.py
```

```
[16]
```

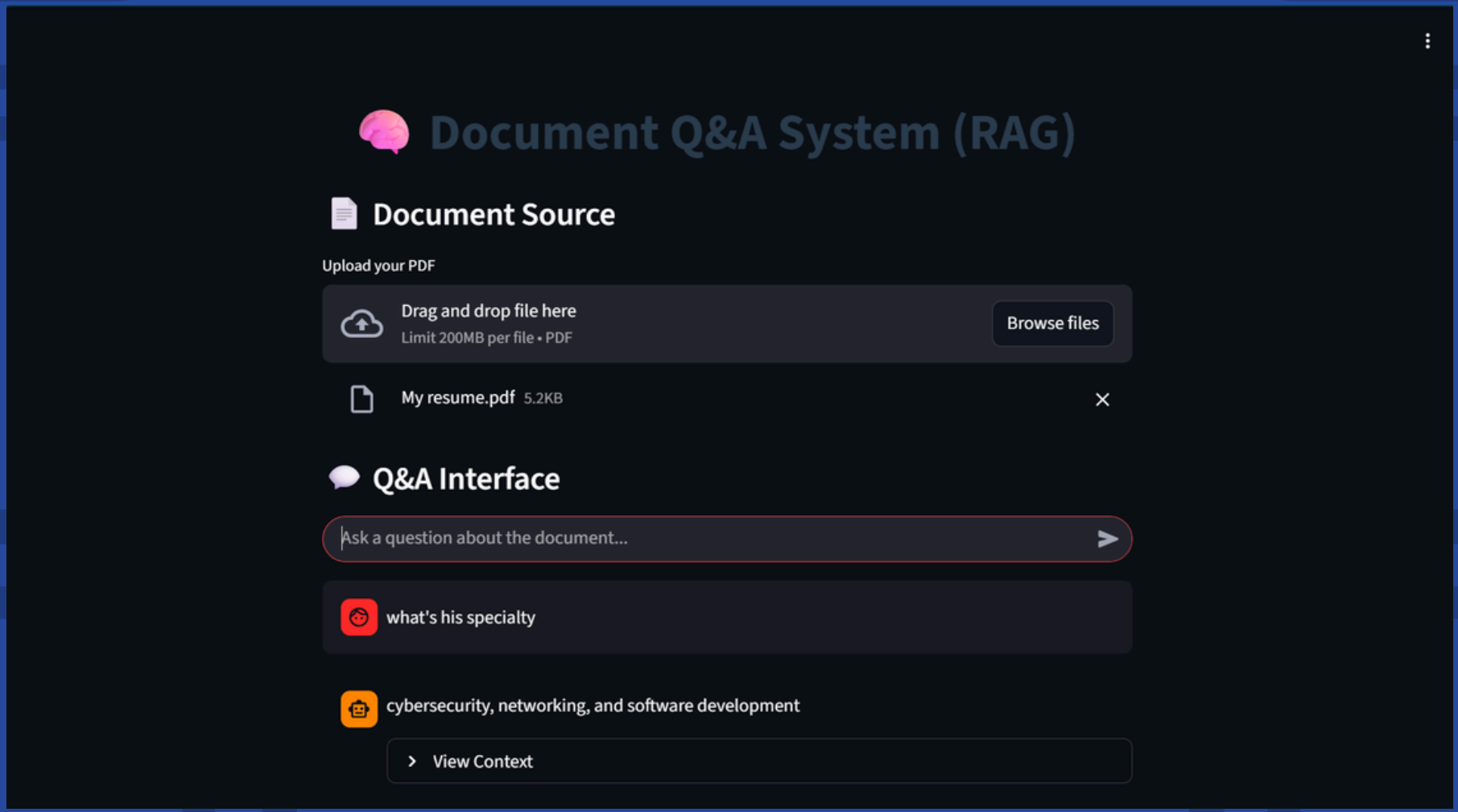
```
from pyngrok import ngrok
import os

# Set your ngrok authtoken (replace 'YOUR_AUTH_TOKEN' with the token you copied)
ngrok.set_auth_token('36lgZcNFuhX77t8AFgepxgzI0pl_22DjRc7oVpt9toqQLEWWj')

# Set up the ngrok tunnel (Streamlit runs on port 8501 by default)
public_url = ngrok.connect(8501)

# Run the Streamlit app in the background
os.system('streamlit run app.py &')

# Display the public URL to access the app
print('Streamlit is live at:', public_url)
```

```
Streamlit is live at: NgrokTunnel: "https://laurette-improper-ronnie.ngrok-free.dev" -> "http://localhost:8501"
```

OUR MODEL!


Document Q&A System (RAG)
Document Source
Upload your PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
My resume.pdf 5.2KB
Q&A Interface
Ask a question about the document...
whats the person name
Adham Hany
View Context

OUR MODEL!


Document Q&A System (RAG)
Document Source
Upload your PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
My resume.pdf 5.2KB
Q&A Interface
Ask a question about the document...
what's his specialty
cybersecurity, networking, and software development
View Context

THANK YOU!