nsfinalprojectcode

November 20, 2023

[1]:
```
#imports

from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.chains.question_answering import load_qa_chain
from langchain.llms import OpenAI
import os
import time
```

[2]:
```
#creating the environment

import os
os.environ["OPENAI_API_KEY"] = "API_KEY"
```

[3]:
```
# Folder containing PDF files
pdf_folder = '/Users/navyakamireddy/Documents/Network Security Project Data'
```

[4]:
```
# Function to extract text from PDF using PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, 'rb') as file:
        doc_reader = PdfReader(file)
        extracted_text = ''
        for page in doc_reader.pages:
            extracted_text += page.extract_text()
    return extracted_text
```

[5]:
```
# Function to generate questions from text
def generate_questions(text):
    sentences = text.split('.')
    questions = [f"What is {sentence}?" for sentence in sentences if sentence.
 ↪strip()]
    return questions
```

[6]:
```python
# Function to process a single page
def process_page(page):
    try:
        return page.extract_text()
    except Exception as e:
        print(f"Error extracting text from page: {e}")
        return ""
```

[7]:
```python
# Function to generate an answer from ChatGPT
def generate_chatgpt_answer(question):
    # Set your OpenAI GPT-3 API key
    openai.api_key = "API_KEY"

    # Define the prompt for ChatGPT
    prompt = f"Question: {question}\nAnswer:"

    # Use OpenAI's completion API to generate an answer
    response = openai.Completion.create(
        engine="text-davinci-003",  # You can experiment with different engines
        prompt=prompt,
        max_tokens=100,  # Adjust as needed
        n=1,  # Number of completions to generate
    )

    # Extract the generated answer from the API response
    chatgpt_answer = response.choices[0].text.strip()

    return chatgpt_answer
```

[8]:
```python
# Folder containing PDF files
pdf_folder = '/Users/navyakamireddy/Documents/Network Security Project Data'
```

[9]:
```python
batch_size = 100
```

[ ]:
```python
# Process each PDF file in the folder
for pdf_file in os.listdir(pdf_folder):
    if pdf_file.endswith(".pdf"):
        pdf_path = os.path.join(pdf_folder, pdf_file)

        try:
            start_time = time.time()

            # Extract text from the PDF
            extracted_text = extract_text_from_pdf(pdf_path)

            # Split text into smaller chunks
            text_splitter = CharacterTextSplitter(
```

```
            separator="\n",
            chunk_size=500,
            chunk_overlap=100,
            length_function=len,
        )
        texts = text_splitter.split_text(extracted_text)

        # Generate questions from the extracted text
        questions = generate_questions(extracted_text)

        # Embeddings and vector search
        embeddings = OpenAIEmbeddings()
        docsearch = FAISS.from_texts(texts, embeddings)

        # Load question-answering chain
        openai_api_key = os.environ["OPENAI_API_KEY"]
        os.environ["OPENAI_API_KEY"] = "API_KEY"

        llm = OpenAI(temperature=0.7, openai_api_key=openai_api_key)
        chain = load_qa_chain(llm, chain_type="stuff")

        # Ask the user for the question
        user_question = input("Enter your Question: ")

        # Use vector search to find relevant documents for the question
        docs = docsearch.similarity_search(user_question)

        # Run the question-answering pipeline on the relevant documents
        answer_from_document = chain.run(input_documents=docs,␣
↪question=user_question)

        # Check if the answer from the document is not present
        if answer_from_document.strip():
            # Print the question and the corresponding answer
            print(f"Question: {user_question}")
            print(f"Answer from {pdf_path}: {answer_from_document}")
            answer_source = "Document"
        else:
            # Use ChatGPT for answering
            chatgpt_answer = generate_chatgpt_answer(user_question)
            print(f"Answer from ChatGPT for {pdf_path}: {chatgpt_answer}")
            answer_source = "ChatGPT Answer"

        # Add the answer source to your data or logging system for␣
↪traceability
        # This variable 'answer_source' will indicate whether the answer␣
↪came from the document or ChatGPT
```

```
            print(f"PDF: {pdf_path}")
            print(f"Answer source: {answer_source}")
            print("=" * 50)

        except Exception as e:
            print(f"Error processing {pdf_path}: {str(e)}")
            continue
```

```
Enter your Question: does Mac provide integrity

Created a chunk of size 1294, which is longer than the specified 500

Question: does Mac provide integrity
Answer from /Users/navyakamireddy/Documents/Network Security Project
Data/Lecture 19(1).pdf:  Yes. An attacker cannot tamper with the message without
being detected.
PDF: /Users/navyakamireddy/Documents/Network Security Project Data/Lecture
19(1).pdf
Answer source: Document
==================================================
Enter your Question: what are the ingredients of symmetric encryption
Question: what are the ingredients of symmetric encryption
Answer from /Users/navyakamireddy/Documents/Network Security Project
Data/Lecture 3.pdf:  The ingredients of symmetric encryption are plaintext,
encryption algorithm, secret key, ciphertext, and decryption algorithm.
PDF: /Users/navyakamireddy/Documents/Network Security Project Data/Lecture 3.pdf
Answer source: Document
==================================================
```

[ ]:

[ ]: