Set up the env through google colab to run the code

```
# Install required libraries
!pip install openai==0.28 langchain faiss-cpu sentence-transformers tiktoken pdfplumber
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.5/48.5 kB 2.0 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pdfminer.six==20231228-
Requirement already satisfied: cryptography>=36.0.0 in /usr/local/lib/python3.10/dist-packages (from pdfminer.six==20231228->pdfp
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (2
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (6.1.0
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.17.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0->sentenc
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.4.0,>=0.3.1
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>=0.1.17->langch
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>=0.1.17->l
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.7.4->la
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.7.4->lan
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain) (
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence-transformers) (3
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence-transformers) (3.1
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence-transformer
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch>=1.11.0->
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.41.0->s
Requirement already satisfied: tokenizers<0.21,>=0.20 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.41.
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->sentence-transformers
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->sentence-trans
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography>=36.0.0->pdfminer.six==20
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.17-
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.1
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain-
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.11.0->sentence-t
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography>=36.0.0->pdfmi
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx<1,>=0.23.0->langsmith
Downloading openai-0.28.0-py3-none-any.whl (76 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.5/76.5 kB 324.7 kB/s eta 0:00:00
Downloading faiss_cpu-1.9.0.post1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (27.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.5/27.5 MB 10.9 MB/s eta 0:00:00
Downloading tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 13.5 MB/s eta 0:00:00
Downloading pdfplumber-0.11.4-py3-none-any.whl (59 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.2/59.2 kB 2.4 MB/s eta 0:00:00
Downloading pdfminer.six-20231228-py3-none-any.whl (5.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.6/5.6 MB 28.6 MB/s eta 0:00:00
Downloading pypdfium2-4.30.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 23.4 MB/s eta 0:00:00
Installing collected packages: pypdfium2, faiss-cpu, tiktoken, pdfminer.six, pdfplumber, openai
  Attempting uninstall: openai
    Found existing installation: openai 1.54.4
    Uninstalling openai-1.54.4:
      Successfully uninstalled openai-1.54.4
Successfully installed faiss-cpu-1.9.0.post1 openai-0.28.0 pdfminer.six-20231228 pdfplumber-0.11.4 pypdfium2-4.30.0 tiktoken-0.8
```

```
# Import libraries
import openai
import faiss
import tiktoken
import pdfplumber
import random
import numpy as np
import pandas as pd
from langchain.text_splitter import RecursiveCharacterTextSplitter
from sentence_transformers import SentenceTransformer

# Initialize OpenAI API key
openai.api_key = "YOUR_OPENAI_API_KEY"  # Replace with your API key
```

```
/usr/local/lib/python3.10/dist-packages/sentence_transformers/cross_encoder/CrossEncoder.py:13: TqdmExperimentalWarning: Using `tqdm
  from tqdm.autonotebook import tqdm, trange
```

```
# Function to extract text from a PDF file
def extract_text_from_pdf(file_path):
    text = ""
    with pdfplumber.open(file_path) as pdf:
        for page in pdf.pages:
            text += page.extract_text()
    return text
```

Check if the pdf is upload successfully, you can also replace the document with any pdf that you have, MAKE SURE TO HAVE THE RIGHT PATH OF THE FILE!

```
# Extract text from the uploaded PDF
document_text = extract_text_from_pdf('/ORD_Use_Agreement.pdf')

# Check if the text extraction was successful
if document_text:
    print("PDF successfully uploaded and text extracted!")
    # Print the first 1000 characters to verify the content
    print("\nSample Extracted Text (First 1000 characters):\n")
    print(document_text[:1000])  # Print only the first 1000 characters
else:
    print("Failed to extract text from the PDF. Please check the file path and try again.")
```

```
PDF successfully uploaded and text extracted!

Sample Extracted Text (First 1000 characters):

CHICAGO- O'HARE INTERNATIONAL AIRPORT
*************************************
AMENDED AND RESTATED
AIRPORT USE AGREEMENT
AND TERMINAL FACILITIES LEASE
************************************
(As Amended through 2001 – Unofficial Version)
J54154-2 C:\Documents and Settings\OM00022\Local
Settings\Temp\XPgrpwise\ohareuseagreementamendedandrestatedunoff
icialversion.wpdTABLE OF CONTENTS
Page
ARTICLE I DEFINITIONS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
Section 1.01 - Definitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
Section 1.02 - Interpretation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
Section 1.03 - Incorporation of Exhibits . . . . . . . . . . . . . . . . . . . . . . . . 18
ARTICLE II TERM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
Section 2.01 - Term of Agreement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
ARTICLE III
```

Text splitting function for optimization

```
# Function to split text by tokens using tiktoken
def split_text_by_tokens(text, token_limit):
    tokenizer = tiktoken.get_encoding("cl100k_base")
    tokens = tokenizer.encode(text)
    chunks = []
    for i in range(0, len(tokens), token_limit):
        chunk = tokens[i:i + token_limit]
        chunks.append(tokenizer.decode(chunk))
    return chunks
doc_chunk = split_text_by_tokens(document_text, token_limit=512)
```

Test if text split works, if it works it should print partial words chunck from the document, you can also change the index number to go through different words chunck of the file as well

```
print("Number of Chunks Created:", len(doc_chunk))
print("\nSample Chunk (First 100 characters of the first chunk):\n")
print(doc_chunk[25][:100])  # Print the first 100 characters of the first chunk
```

```
Number of Chunks Created: 137

Sample Chunk (First 100 characters of the first chunk):

 principal amount of $8,000,000.
(65) "1959 Terminal Lease Agreement" means the lease, if any, of te
```

DO NOT RUN THIS! Create the embeddings (This is the first approach mentioned in the paper but due to the price of OpenAI token we gonna use another model, this code only shows the how to create embeddings with OpenAI)

```
'''
# expensive
# Initialize OpenAI API key
openai.api_key = ""  # Replace with your OpenAI API key

# Function to generate embeddings for text chunks using the new API interface
def get_embeddings(texts):
    response = openai.Embedding.create(
        model="text-embedding-ada-002",  # Use the correct embedding model
        input=texts
    )
    # Extract and return the embeddings
    return [embedding["embedding"] for embedding in response["data"]]

# Example: Generate embeddings for your text chunks
embeddings = get_embeddings(doc_chunk)
'''
```

```
'\n# expensive\n# Initialize OpenAI API key\nopenai.api_key = ""  # Replace with your OpenAI API key\n\n# Function to generate embe
ddings for text chunks using the new API interface\ndef get_embeddings(texts):\n    response = openai.Embedding.create(\n        mo
del="text-embedding-ada-002",  # Use the correct embedding model\n        input=texts\n    )\n    # Extract and return the embeddin
```

As mentioned earlier here we choose OPEN RESOURCE embeddings since it's free!

```
# Load a pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')  # You can choose another model if needed

# Generate embeddings for your text chunks
embeddings = model.encode(doc_chunk)

print("Generated embeddings using free model successfully!")
```

```
/usr/local/lib/python3.10/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
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(
modules.json: 100%                          349/349 [00:00<00:00, 20.2kB/s]
config_sentence_transformers.json: 100%     116/116 [00:00<00:00, 5.64kB/s]
README.md: 100%                             10.7k/10.7k [00:00<00:00, 479kB/s]
sentence_bert_config.json: 100%             53.0/53.0 [00:00<00:00, 2.06kB/s]
config.json: 100%                           612/612 [00:00<00:00, 37.3kB/s]
model.safetensors: 100%                     90.9M/90.9M [00:00<00:00, 148MB/s]
tokenizer_config.json: 100%                 350/350 [00:00<00:00, 20.3kB/s]
vocab.txt: 100%                             232k/232k [00:00<00:00, 1.79MB/s]
tokenizer.json: 100%                        466k/466k [00:00<00:00, 3.49MB/s]
special_tokens_map.json: 100%               112/112 [00:00<00:00, 5.76kB/s]
1_Pooling/config.json: 100%                 190/190 [00:00<00:00, 8.08kB/s]
Generated embeddings using free model successfully!
```

Create vector database with the embeddings + keep the original chunk for the key words matching method

```
# Initialize the FAISS index for vector-based retrieval
dimension = len(embeddings[0])  # The dimension of the embeddings
index = faiss.IndexFlatL2(dimension)  # Using L2 distance for similarity search

# Convert embeddings to a NumPy array and add to the index
index.add(np.array(embeddings, dtype=np.float32))
print("Embeddings added to FAISS index successfully!")
```

```
Embeddings added to FAISS index successfully!
```

Vectorize user query (Vectorize user's questions)

```
# Function to vectorize the user query
def vectorize_query(query):
    # Use the same model (sentence-transformers) to create the embedding
    query_embedding = model.encode([query])[0]
    return np.array([query_embedding], dtype=np.float32)  # Convert to NumPy array
```

```
# Function to retrieve top-k relevant chunks from the vector database
def retrieve_top_k_chunks(query_embedding, top_k=3):
    distances, indices = index.search(query_embedding, top_k)  # Search FAISS index
    top_chunks = [doc_chunk[i] for i in indices[0]]  # Retrieve the corresponding text chunks
    return top_chunks
```

Combine the top-k related words chunck from the document and give chatgpt a new prompt (So now the new prompt that ChatGPT recieved includes the original question and the top-k related words chunk from the document)

```
# Function to format the context for ChatGPT
def format_context_for_chatgpt(user_query, retrieved_chunks):
    context = "\n".join(retrieved_chunks)  # Combine the chunks into a single string
    prompt = f"Question: {user_query}\n\nContext:\n{context}\n\nAnswer:"
    return prompt
```

***Make sure to have your OpenAI API key here! Each call would cost around $0.005, to access this make sure you have token money in ur account ***

```
# Function to get an answer from ChatGPT
openai.api_key = "" ## Important! to successfully run this part of code you need an OpenAI API key and make sure there's purchased token
def get_chatgpt_response(prompt):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",  # Use the new model
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        max_tokens=300,  # Adjust the max_tokens as needed
        temperature=0.7  # Adjust temperature for more or less creative answers
    )
    return response['choices'][0]['message']['content'].strip()
```

Call RAG while accessing ChatGPT, you can also change questions as well

```
# Complete RAG workflow function
def rag_chatbot(user_query, top_k=5):
    # Step 1: Vectorize the user query
    query_embedding = vectorize_query(user_query)
    # Step 2: Retrieve top-k relevant chunks
    retrieved_chunks = retrieve_top_k_chunks(query_embedding, top_k)

    # Step 3: Format the context for ChatGPT
    prompt = format_context_for_chatgpt(user_query, retrieved_chunks)

    # Step 4: Get the response from ChatGPT
    answer = get_chatgpt_response(prompt)

    return answer

# Example usage
user_question = "What are the key concepts discussed in the document?"
answer = rag_chatbot(user_question)
print(f"Answer from ChatGPT:\n{answer}")
```

```
Answer from ChatGPT:
The key concepts discussed in the document include definitions and interpretations of terms used in the agreement, the term of the a
```