In [ ]:
```
### Installs ###
# %pip install langchain openai pypdf faiss-cpu python-dotenv tiktoken
```

Documentation

NEED AN OPENAI API KEY TO RUN

Environment Setup

- Load environment variables using `dotenv`.
- Import necessary libraries like `os`, `glob`, and various modules from `langchain`.

Finding PDF Files

- Function `find_pdf_files(directory)` lists all PDF files in a specified directory and subdirectories.
- Uses `glob.glob` for searching and sorts the absolute paths of the PDF files.

Loading and Splitting Text

- `load_and_split(path)` loads a PDF and splits its text into chunks.
- Utilizes `PyPDFLoader` for reading PDF content.
- `CharacterTextSplitter` breaks text into 1000 character chunks, separated by newline (`\n`).

Setup PDF Files and Embeddings

- Defines a list of file paths to various PDF documents.
- Creates an embedding model using `OpenAIEmbeddings`.

Creating and Merging Vector Stores

- For each PDF, the text is loaded, split, and converted into vector representations using `FAISS`.
- Merges these vector stores into one primary store.

Saving and Loading Vector Store

- Saves the merged vector store locally.
- Reloads it using `FAISS.load_local`.

Demonstration with QA System

- Sets up a QA system using `RetrievalQA` with an `OpenAI` language model and the loaded vector store.
- Runs example queries to demonstrate information retrieval from the processed PDF documents.

This setup establishes a document retrieval system using embeddings and a QA model, capable of answering questions based on content from the loaded PDF documents.

In [ ]:
```
from dotenv import load_dotenv
load_dotenv()

import os
import glob
```

In [ ]:
```
from langchain.document_loaders import PyPDFLoader
from langchain.embeddings import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.chains import RetrievalQA
from langchain.llms import OpenAI
```

Setup

In [ ]:
```
def find_pdf_files(directory):
    os.chdir(directory)

    pdf_files = []

    for file in glob.glob('**/*.pdf', recursive=True):
        absolute_path = os.path.abspath(file)
        pdf_files.append(absolute_path)

    pdf_files.sort()

    return pdf_files
```

In [ ]:
```
def load_and_split(path):
    loader = PyPDFLoader(file_path=path)
    documents = loader.load()

    # A chunk size of 1000 characters offers a balance between granularity and manageability
    # It's large enough to contain meaningful units of text (like sentences or paragraphs),
    # but small enough to be easily processed by various algorithms
    # The absence of overlap means each character in the document is only processed once
    text_splitter = CharacterTextSplitter(
            chunk_size=1000, chunk_overlap=0, separator="\n"
        )
    return text_splitter.split_documents(documents=documents)
```

In [ ]:
```
pdf_files = [
    "/Users/aditkapoor/Local Documents/Work/Cognizant/langchain-proj/assets/data/gray-city.pdf",
    "/Users/aditkapoor/Local Documents/Work/Cognizant/langchain-proj/assets/data/irl.pdf",
    "/Users/aditkapoor/Local Documents/Work/Cognizant/langchain-proj/assets/data/singing-peddler.pdf",
    "/Users/aditkapoor/Local Documents/Work/Cognizant/langchain-proj/assets/data/memoirs.pdf",
    "/Users/aditkapoor/Local Documents/Work/Cognizant/langchain-proj/assets/data/small-little-circle.pdf",
    "/Users/aditkapoor/Local Documents/Work/Cognizant/langchain-proj/assets/data/veracious.pdf",
]
```

In [ ]:
```
embeddings = OpenAIEmbeddings()
vectorstores = []

for file_path in pdf_files:
    docs = load_and_split(file_path)

    # I chose to use the FAISS vectorstore because it's the fastest to work with locally as I often
    # needed to rebuild the vectorstore when I made changes to the code. With Pinecone, it was harder
    # to make changes as I had to reset the vectorstore on the website instead of just deleting the locally
    # saved stone (as FAISS manages it)
    vectorstore = FAISS.from_documents(docs, embeddings)
    vectorstores.append(vectorstore)
```

In [ ]:
```
### DEBUG CELL ###
# vectorstores[0].docstore._dict
```

In [ ]:
```
for i in range(1, len(vectorstores)):
    vectorstores[0].merge_from(vectorstores[i])

vectorstores[0].save_local("faiss_index_project")
```

Demonstration

In [ ]:
```
embeddings = OpenAIEmbeddings()
new_vectorstore = FAISS.load_local("faiss_index_project", embeddings)
qa = RetrievalQA.from_chain_type(llm=OpenAI(), chain_type='stuff', retriever=new_vectorstore.as_retriever())
```

In [ ]:
```
result = qa.run("Who did Clyde yet at about giving omniscient third person narrators too much information? If you don't know, say you don't know.")
print(result)
```

```
Clyde's mom.
```

In [ ]:
```
result = qa.run("In the memoir, why was the father sad after receiving the letter?")
print(result)
```

```
 The father was sad after receiving the letter because it was the last letter he would ever receive from his son, who had been shot two weeks earlier.
```

In [ ]:
```
result = qa.run("In the crowded streets of Delhi, what stories do the patches on people's clothing tell, as described in the poem? Reflect on the diversity and history embedded in these patches.")
print(result)
```

```
 The patches on people's clothing in the crowded streets of Delhi tell stories of poverty, of different fabrics from different times, of hand-me-downs, and of people striving to cover up patches that are a
ll too visible. They reflect the diversity and history of the city, of a place full of people with different backgrounds and experiences.
```

In [ ]:
```
result = qa.run("How does \"IRL\" portray the world post-internet and modern relationships?")
print(result)
```

```
IRL portrays the world post-internet and modern relationships as interconnected and interdependent. The author describes how two people were connected through the internet and were finally able to meet in
person. The internet has connected people in ways which would have been impossible before, allowing them to form relationships which transcend physical barriers.
```

In [ ]:
```
result = qa.run("In \"Veracious\", why is the author so mad at his Uncle despite his untimely death? Give me around 5 sentences.")
print(result)
```

```
 In "Veracious", the author is mad at his Uncle Scott for leaving his family behind in such a difficult time. The author feels especially hurt as his Uncle had been doing better and making plans for the fu
ture, like getting a job and moving out of his family home. The author had looked up to his Uncle as a child, and was disappointed that he had left them all so suddenly. The author felt that his Uncle was
selfish for not being able to wait, even in death, and for reducing his mother to a vulnerable state. The author was also angry that his Uncle had left his grandmother, JoJo, all alone in her house with no
one to take care of her.
```

In [ ]:
```
result = qa.run("Describe the Author's relationship with Maggi. Why was he so attached to her?")
print(result)
```

```
 The author felt a strong connection with Maggi. He found comfort in her presence and in the way she interacted with him. He was able to confide in her and talk to her as if she was an understanding perso
n. He felt like he could share his feelings and emotions with her without judgement, and this made him feel less alone.
```

Testing

In [ ]:
```
prompt = ""  # Fill in with your own prompt
result = qa.run(prompt)
print(result)
```