實作教學
LangChain RAG

RAG

- langchain_rag_doc.py >> 這個_doc代表是匯入文件檔案doc

1.你會需要import以下套件

- from langchain.chains.combine_documents import create_stuff_documents_chain
- from langchain.chains import create_retrieval_chain
- from langchain_core.prompts import ChatPromptTemplate
- from langchain_community.llms import Ollama
- from langchain_community.embeddings import OllamaEmbeddings
- from langchain_community.vectorstores import FAISS
- from langchain_core.documents import Document
- from langchain_community.document_loaders import PyPDFLoader
- from langchain.text_splitter import CharacterTextSplitter

- from langchain_core.documents import Document
- from langchain_community.document_loaders import PyPDFLoader

2.建立模型和文件

- # 初始化Ollama模型
- llm = Ollama(model='llama3')

- # 建立文件列表，每個文件包含一段文字內容
- docs = [
- Document(page_content='曼德珍珠奶茶草：這種植物具有強大的魔法屬性，常用於恢復被石化的受害者。'),
- Document(page_content='山羊可愛蓮花石：是一種從山羊胃中取出的石頭，可以解百毒。在緊急情況下，它被認為是最有效的解毒劑。'),
- Document(page_content='日本小可愛佐籐鱗片：這些鱗片具有強大的治愈能力，常用於製作治療藥水，特別是用於治療深層傷口。'),
-]

文件中的內容為公司自己獨特的專業內容，一般普世通俗的內容，Llama3本來就已經知道了~

3.設定文本分割器

- # 設定文本分割器，chunk_size是分割的大小，chunk_overlap是重疊的部分
- text_splitter = CharacterTextSplitter(chunk_size=20, chunk_overlap=5)
- documents = text_splitter.split_documents(docs) # 將文件分割成更小的部分

3.設定文本分割器

chunk_size (塊大小)

- 定義: 每個分割塊的大小，以字符數量為單位。
- 作用: 決定每個文本塊包含多少字符。

chunk_overlap (塊重疊)

- 定義: 相鄰文本塊之間重疊的字符數量。
- 作用: 確保每個分割後的文本塊之間有一些重疊部分，以保證連貫性和上下文不丟失。

通過設置塊重疊部分，我們可以確保每個分割後的文本塊仍然包含足夠的上下文信息，避免因切割造成的信息丟失或語義斷裂。

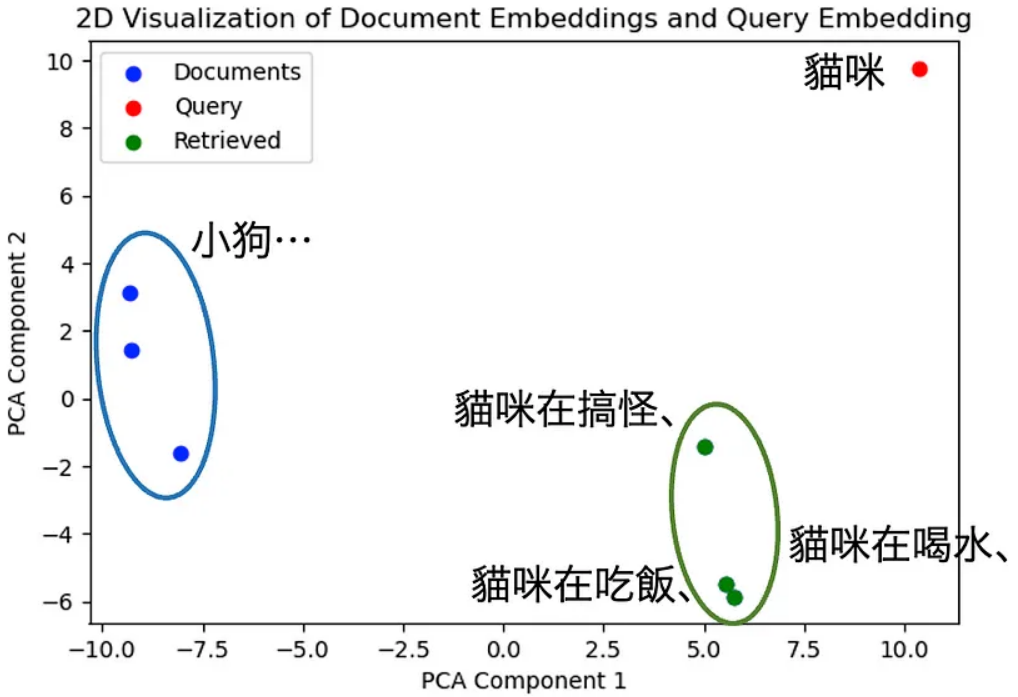
4.建置embeddings和向量資料庫

- # 初始化嵌入模型
- embeddings = OllamaEmbeddings()

- # 使用FAISS建立向量資料庫
- vectordb = FAISS.from_documents(docs, embeddings)
- # 將向量資料庫設為檢索器
- retriever = vectordb.as_retriever()

Faiss簡介

- Faiss（Facebook AI Similarity Search）是一個由Facebook AI Research開發的開源庫，用於高效的相似性搜索和密集向量（dense vector）的聚類。
- 它在大規模資料集上提供快速的向量檢索和聚類，特別適合在機器學習和深度學習應用中使用。



5.設定提示模板

- # 設定提示模板，將系統和使用者的提示組合
- prompt = ChatPromptTemplate.from_messages([
- ('system', 'Answer the user\'s questions in Chinese, based on the context provided below:\n\n{context}'),
- ('user', 'Question: {input}'),
-])

6.llm和提示模板結合

- # 創建文件鏈，將llm和提示模板結合
  document_chain = create_stuff_documents_chain(llm, prompt)

  # 創建檢索鏈，將檢索器和文件鏈結合
  retrieval_chain = create_retrieval_chain(retriever, document_chain)

7.從用戶輸入中獲取問題，並用retrieval_chain來回答

```
context = []
input_text = input('>>> ')
while input_text.lower() != 'bye':
    response = retrieval_chain.invoke({
        'input': input_text,
        'context': context
    })
    print(response['answer'])
    context = response['context']
    input_text = input('>>> ')
```

retrieval_chain.invoke 是執行這條chain的代碼，以前的教學或許會看到.run，但最新版本的LangChain會慢慢用.invoke當主流。

RUN程式，開始問Llama3問題吧:>

- author:/app# python3 langchain_rag_doc.py
- >>> 請告訴我珍珠奶茶是？
- 🤔

- 曼德珍珠奶茶草：這種植物具有強大的魔法屬性，常用於恢復被石化的受害者。

實作教學2

- ***langchain_rag_pdf.py >> 這個_doc代表是匯入文件檔案doc***
- 把前面建立模型和文件的程式碼，更改為以下。

- ```
  # 初始化Ollama模型
  llm = Ollama(model='llama3')

  # 載入並分割PDF文件
  loader = PyPDFLoader("文件名稱.pdf")
  docs = loader.load_and_split()
  ```

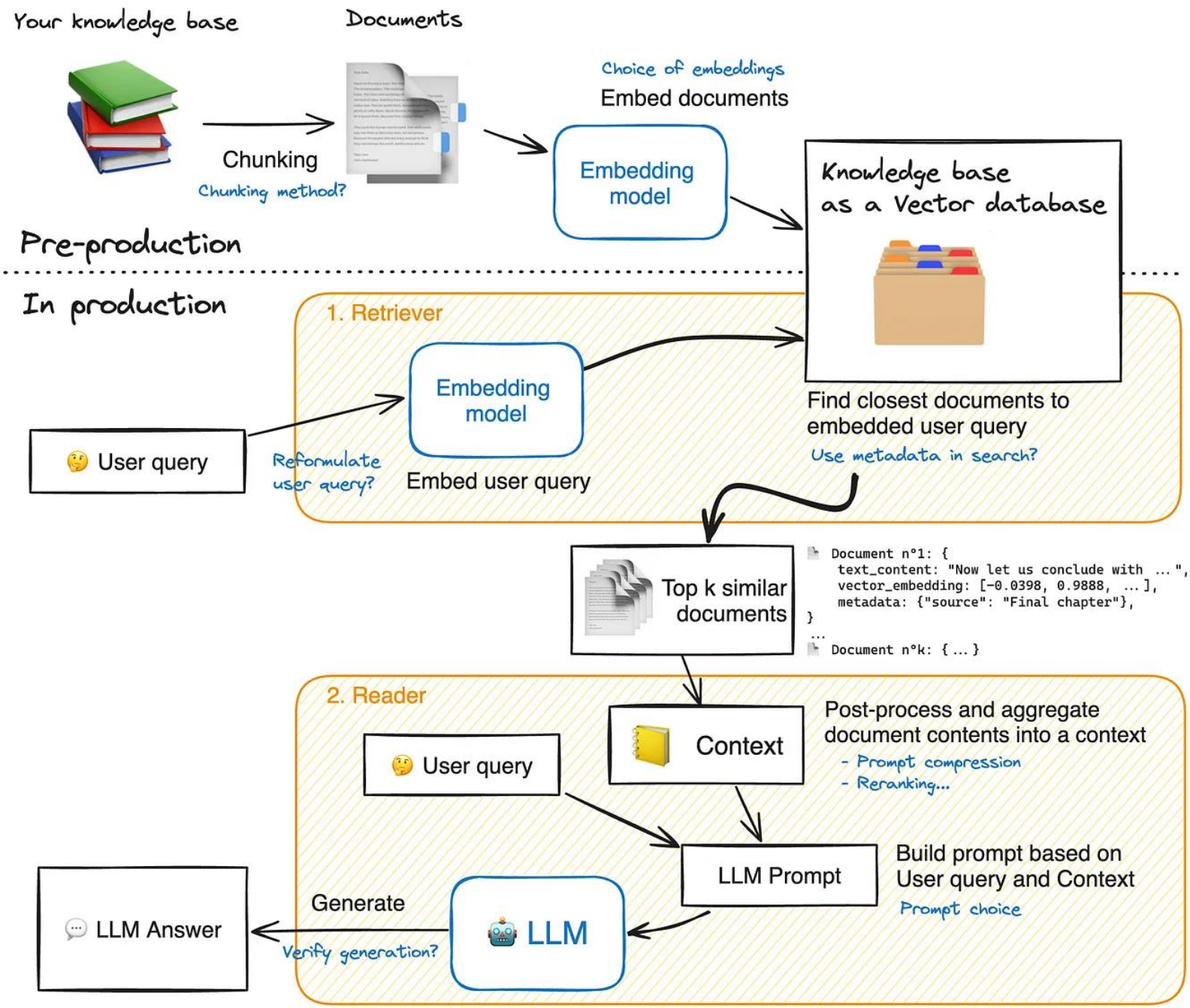
https://github.com/weitsung50110/Huggingface_Langchain_kit/blob/master/pdf_test.pdf
  ```

Demo

- loader = PyPDFLoader("pdf_test.pdf")

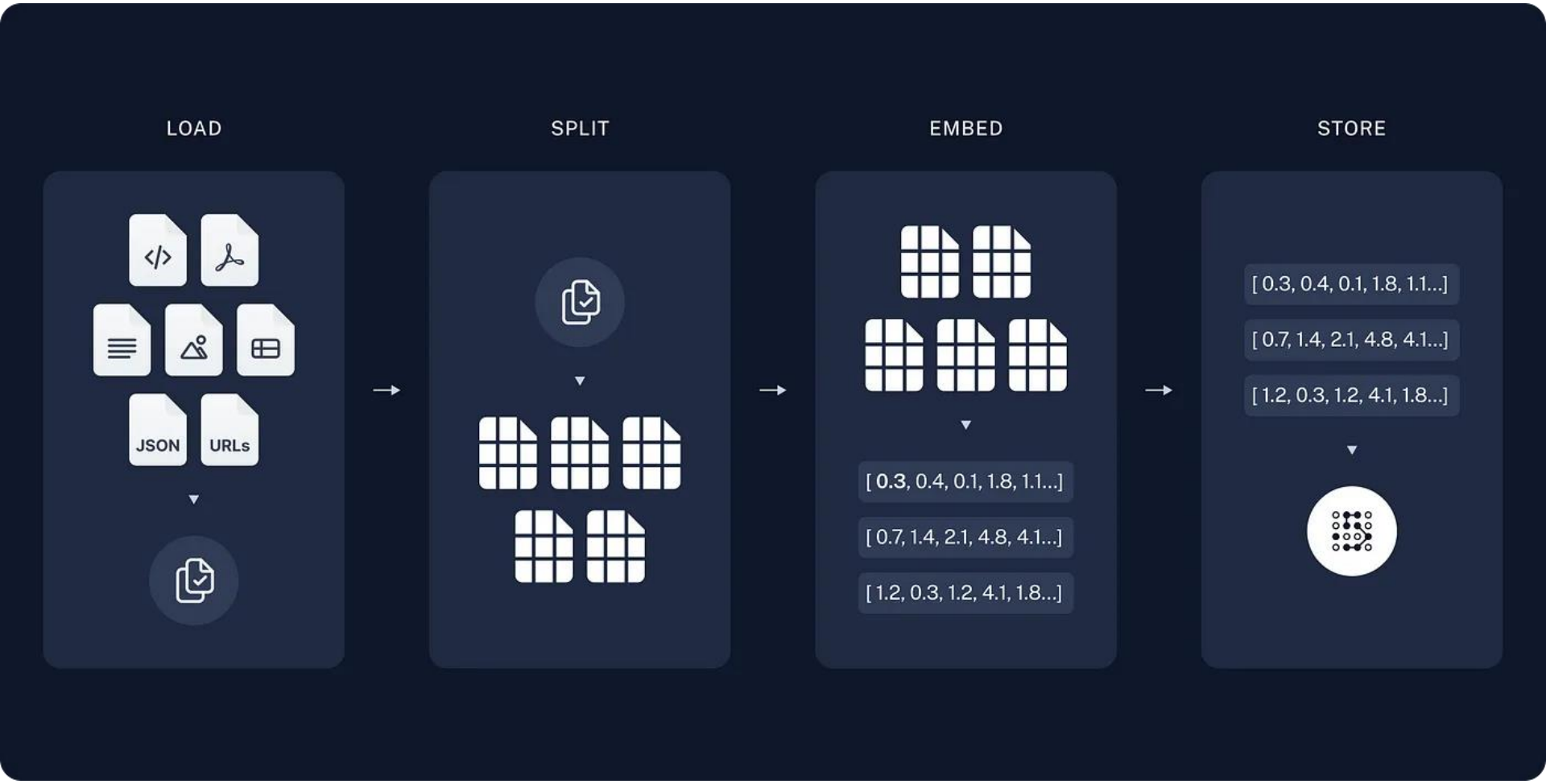
- author:/app# python3 langchain_rag_pdf.py
  >>> 請給我50字的摘要
  CN
  Wei tsung chang 的簡介：碩士畢業、AI軟體工程師，熱心工作，喜歡撰寫教學文章。他的技術栈包括Python、JavaScript、C#、Kotlin等程式語言，以及 Git 版本控制、Docker 等知識。
  (Translation: Introduction to Wei tsung chang: Master's degree holder, AI software engineer, enthusiastic about work, likes writing tutorial articles. His technical stack includes programming languages such as Python, JavaScript, C#, Kotlin, and version control tools like Git.)CN

  >>> 文件中軟體工程師的名字叫什麼？
  🤔
  Answer: Wei Tsung Chang 碩士畢業，同樣可以稱呼為 Weiberson。
  (The software engineer's name is Wei Tsung Chang, and can also be called Weiberson.)


Your knowledge base
Documents
Chunking
Chunking method?
Choice of embeddings
Embed documents
Embedding model
Knowledge base as a Vector database
Pre-production
In production
1. Retriever
Embedding model
User query
Reformulate user query?
Embed user query
Find closest documents to embedded user query
Use metadata in search?
Top k similar documents
Document n°1: {
text_content: "Now let us conclude with ...",
vector_embedding: [-0.0398, 0.9888, ...],
metadata: {"source": "Final chapter"},
}
...
Document n°k: {...}
2. Reader
User query
Context
Post-process and aggregate document contents into a context
- Prompt compression
- Reranking...
LLM Prompt
Build prompt based on User query and Context
Prompt choice
LLM
Generate
Verify generation?
LLM Answer

Step-by-step guide on how to summarize PDFs with GPT

Cost: Not Free, only available with GPT Premium


LOAD
SPLIT
EMBED
STORE
JSON
URLs
[0.3, 0.4, 0.1, 1.8, 1.1...]
[0.7, 1.4, 2.1, 4.8, 4.1...]
[1.2, 0.3, 1.2, 4.1, 1.8...]
[0.3, 0.4, 0.1, 1.8, 1.1...]
[0.7, 1.4, 2.1, 4.8, 4.1...]
[1.2, 0.3, 1.2, 4.1, 1.8...]