



Advisor: Dr. Chang, Henry



LANGCHIAN CHAT WITH YOUR DATA SFBU CHATBOT

Presented by: Shin Cao

San Francisco Bay University



OVERVIEW

- What is RAG?
- What is Langchain?
- Process Explained
- Installation & Setup

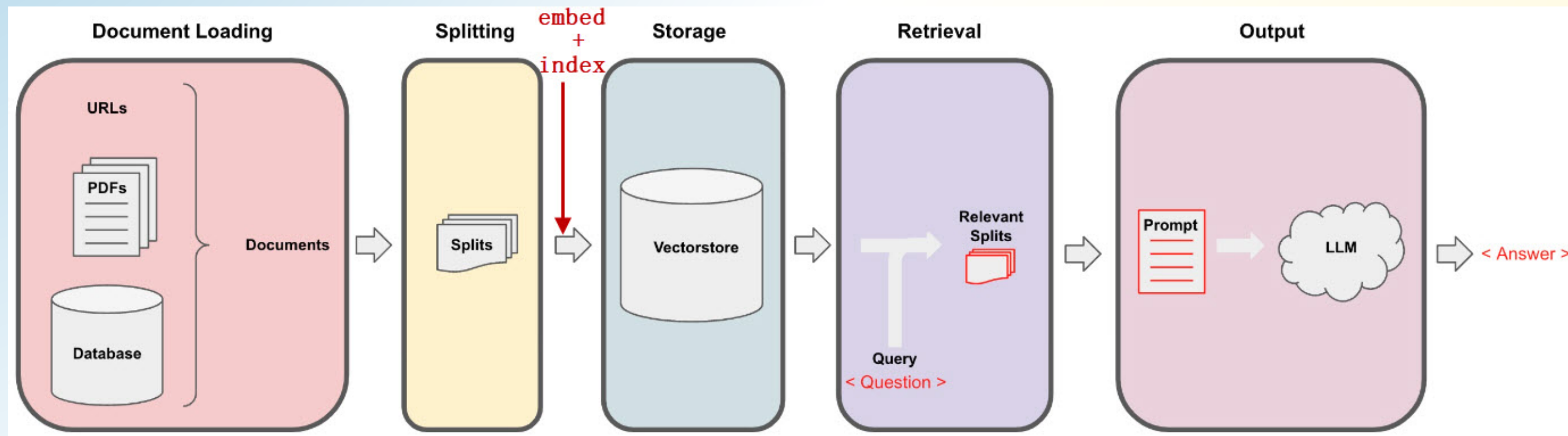


WHAT IS RAG?

- **Brief overview:** "RAG combines the best of retrieval-based and generative AI to answer questions more accurately."
- **Highlight its significance:** "It enriches AI's responses by grounding them in specific, relevant information from a broad dataset."



HOW DOES RAG WORK?





WHAT IS LANGCHAIN?

- "LangChain: A framework designed to simplify the development of language AI applications."
- "Purpose: Facilitates easy integration of language models with external data sources and services."

PROCESS EXPLAINED

STEP 1: CREATE BASIC RETRIEVER



- Load PDF
- Split into different documents
- Embed
- Vector store using chroma

```
loader = PyPDFLoader("2023Catalog.pdf")
documents = loader.load_and_split()
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=150)
docs_split = text_splitter.split_documents(documents)
embeddings = OpenAIEmbeddings()
persist_directory = 'docs/chroma/'
vectordb = Chroma.from_documents(documents=docs_split, embedding=embeddings, persist_directory=persist_directory)
retriever = vectordb.as_retriever()
```


PROCESS EXPLAINED



STEP 2: CREATE HISTORY AWARE RETRIEVER

```
from langchain.chains import create_history_aware_retriever
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder

contextualize_q_system_prompt = """Given a chat history and the latest user question \
which might reference context in the chat history, formulate a standalone question \
which can be understood without the chat history. Do NOT answer the question, \
just reformulate it if needed and otherwise return it as is."""
contextualize_q_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", contextualize_q_system_prompt),
        MessagesPlaceholder("chat_history"),
        ("human", "{input}"),
    ]
)
history_aware_retriever = create_history_aware_retriever(
    llm, retriever, contextualize_q_prompt
)
```

PROCESS EXPLAINED

STEP 3: COMBINE TWO



```
from langchain.chains import create_retrieval_chain
from langchain.chains.combine_documents import create_stuff_documents_chain

qa_system_prompt = """You are an assistant for question-answering tasks. \
Use the following pieces of retrieved context to answer the question. \
If you don't know the answer, just say that you don't know. \
Use three sentences maximum and keep the answer concise.\

{context}"""
qa_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", qa_system_prompt),
        MessagesPlaceholder("chat_history"),
        ("human", "{input}"),
    ]
)

question_answer_chain = create_stuff_documents_chain(llm, qa_prompt)

rag_chain = create_retrieval_chain(history_aware_retriever, question_answer_chain)
```


PROCESS EXPLAINED



STEP 4: STATEFULLY MANAGE CHAT HISTORY

```
### Statefully manage chat history ###
store = {}

def get_session_history(session_id: str) -> BaseChatMessageHistory:
    if session_id not in store:
        store[session_id] = ChatMessageHistory()
    return store[session_id]

conversational_rag_chain = RunnableWithMessageHistory(
    rag_chain,
    get_session_history,
    input_messages_key="input",
    history_messages_key="chat_history",
    output_messages_key="answer",
)
```



INSTALLATION & SETUP




View README on my GitHub:

https://github.com/Shining-in-galaxies/Langchain_Chat_With_Your_Data_SFBU_text



Thanks For Watching

Shin Cao

-  shin.ccx@outlook.com
-  <https://github.com/Shining-in-galaxies>
-  San Francisco Bay University

