Course Notes

introduction

An LLM knows only what it was trained on at the time it was trained.

Recent articles or personal data are not part of the scope.

This course is about learning how to connect generative AI with personal data to be able to chat with them

LangChain overview

Overview

Open-source developer framework for building LLM applications

Python and TypeScript packages

Focused on composition and modularity

Key value adds:

1. Modular components (and implementations of those components)
2. Use cases - common ways to combine those components together

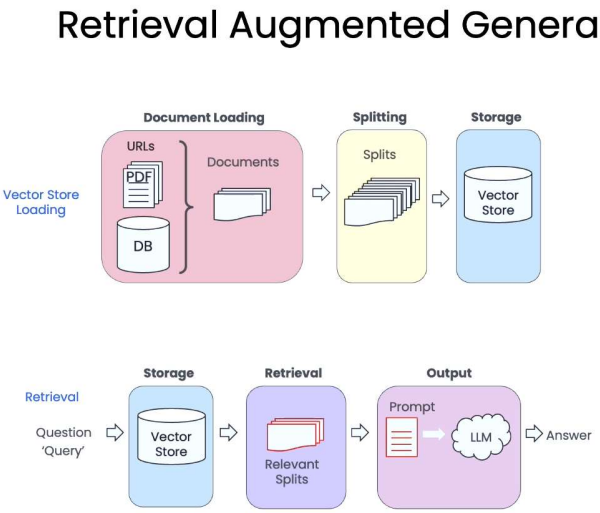
Components

- Prompts
  - Prompt Templates
  - Output Parsers: 5+ implementations
    - Retry/fixing logic
  - Example Selectors: 5+ implementations
- Models
  - LLM's: 20+ integrations
  - Chat Models
  - Text Embedding Models: 10+ integrations
- Indexes
  - Document Loaders: 50+ implementations
  - Text Splitters: 10+ implementations
  - Vector stores: 10+ integrations
  - Retrievers: 5+ integrations/implementations
- Chains
  - Can be used as building blocks for other chains
  - More application specific chains: 20+ different types
- Agents
  - Agent Types: 5+ types
    - Algorithms for getting LLMs to use tools
  - Agent Toolkits: 10+ implementations
    - Agents armed with specific tools for a specific application

RAG (Retrieval Augmented Generation)

Multiples components:

- Document loader to load data from a variety of sources
- Split document into semantic meaningful chunks
- Semantic search to fetch relevant information given a user question (simple method with caveat !!)
- Document retriever to enable the LLM to answer question about a document
- Memory for recording the message exchange between the LLM and the user



Document Loading

To create an app, need to store load the data to a format it can be work with

LangChain has over 80 different type of doc loader!

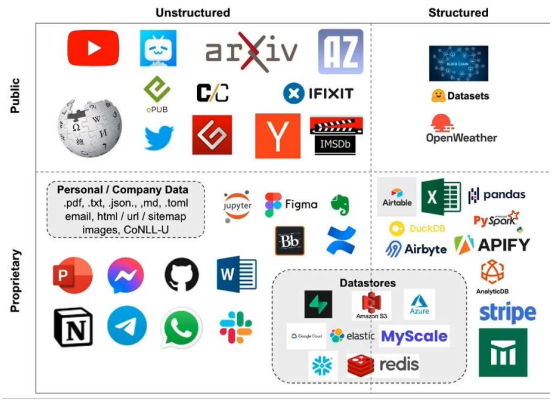
The goal of a loader it to load data into a standard document object (content + metadata)

Loaders

- Loaders deal with the specifics of accessing and converting data
  - Accessing
    - Web Sites
    - Data Bases
    - YouTube
    - arXiv
    - ...
  - Data Types
    - PDF
    - HTML
    - JSON
    - Word, PowerPoint...
- Returns a list of `Document` objects:

```
[
Document(page_content='MachineLearning-Lecture01 \nInstructor (Andrew Ng): Okay. Good morning. Welcome to CS229....',
metadata={'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 0})
...
 Document(page_content='[End of Audio]  \nDuration: 69 minutes  ',
 metadata={'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 21})
]
```

Document Loaders



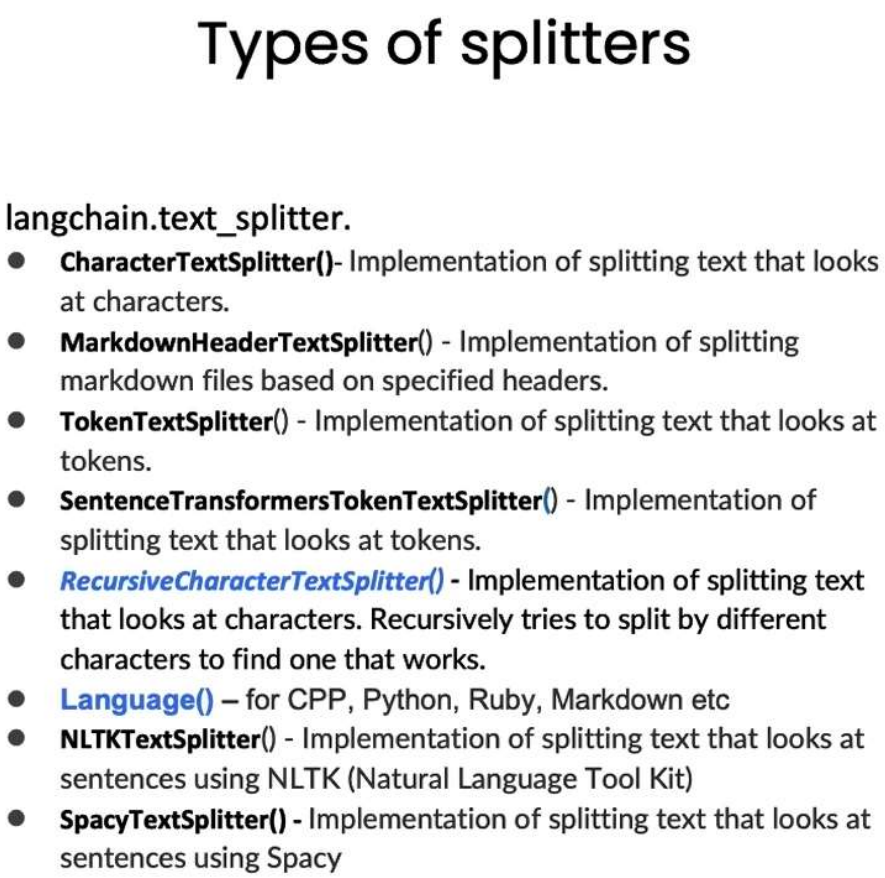
See notebook 01_document_loader.ipynb (loading data from PDF, URLs, Notion database, Youtube Audio)

Document Splitting

- Splitting document previously loaded into smaller chunks
- Sound easy but lots of subtleties

Example of chunking :

- Splitting by a given number of token - > losing context (sentence, paragraph, etc..)
- Splitting with overlap –> keeping a overlap between two chunk, more consistent
- Splitting with context –> Like a markdown file with header, etc..
- Splitting on character -> eg. \n, '.' Or more complex regex



And more on LangChain documentation!

➔ Check notebook 02_document_splitting.ipynb

Vectorstore and embedding

Word embedding / document embedding creates a numerical representation of a text

Text with similar content have similar vectors

Easy to calculate similarity between vector

A vector store is database storing vector easy to query (eg pick the n most similar)

Multiple vector Store are available to store data :

- Chroma is a lightweight in memory vector Stor, good for practice

Using similarity search to find relevant bits to a question. However need to take care of duplicate information to send only excusive inof to the LLM.

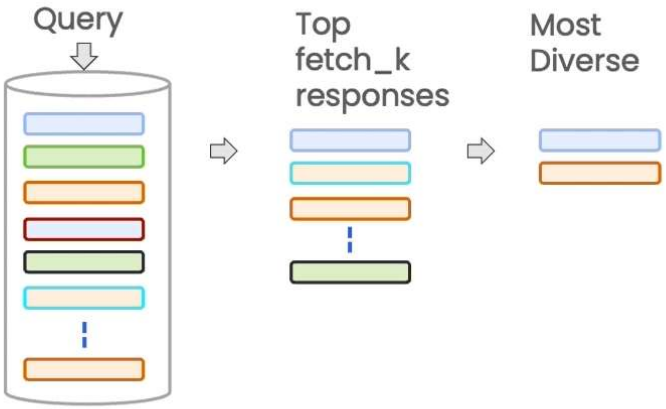
To look in specific sections, like information coming from the 2nd chapter of a book, we need to refine the search using metadata.

➔ Check notebook 03_vectorstores_and_embeddings.ipynb

Retrieval

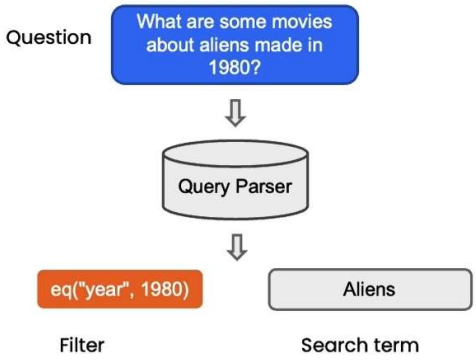
Methods than overcome edge cases that cause problem in semantic search:

Maximum Marginal relevance (MMR) :



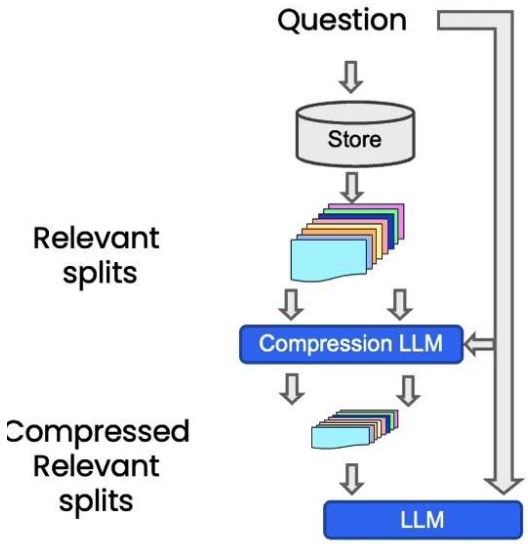
- May not always want the most similar response, to get a larger panel of info
- MMR get the most similar but not only!
- The main ide ais to send a query that return the most similar and then work on selecting diversity out of the most relevant
- Return a final k element
-

LLM aided retrieval



- Related to self query
- Useful when one questions are not only about the content but also about metadata (eg What are soma line movies made in the 1980)
- Using a LLM to split the question into two separate things : a filter (year == 1980) and search term (alien movies)

Compression



Useful to pull only the most useful bit of a chunk

Run the document into a LLM to extract the most important sentences related to the query.

More costly as it run through an LLM, but bringing more relevant that will be sum up

Trade off.

It’s possible to combine technics like MMR + compression to have diverse info that are more precise.

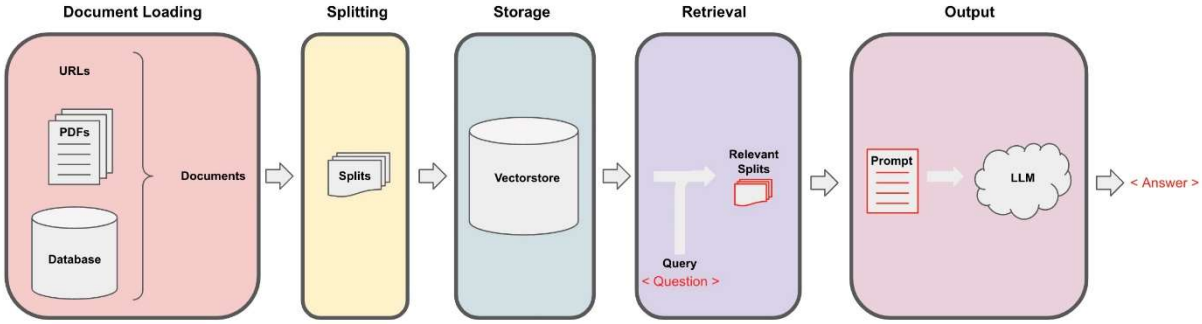
Other Type of retrievers

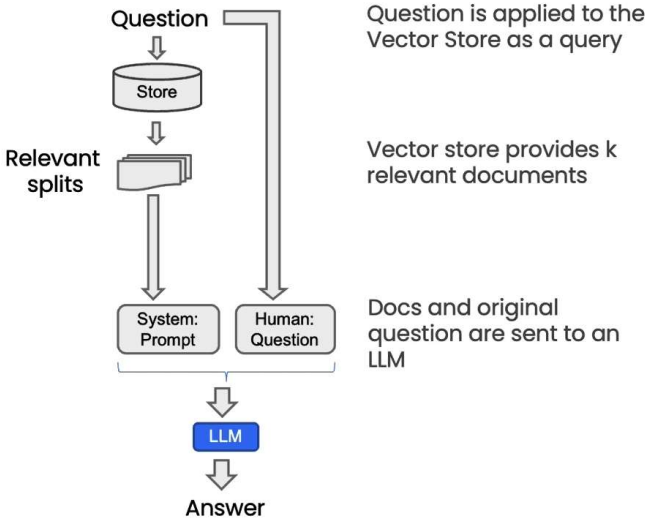
SVM retriever (base on solid vector machine)

TFIDF retriever (based on TF IDF, needs a text split into tokens)

➔ Check notebook 04_retrieval.ipynb

Question Answering

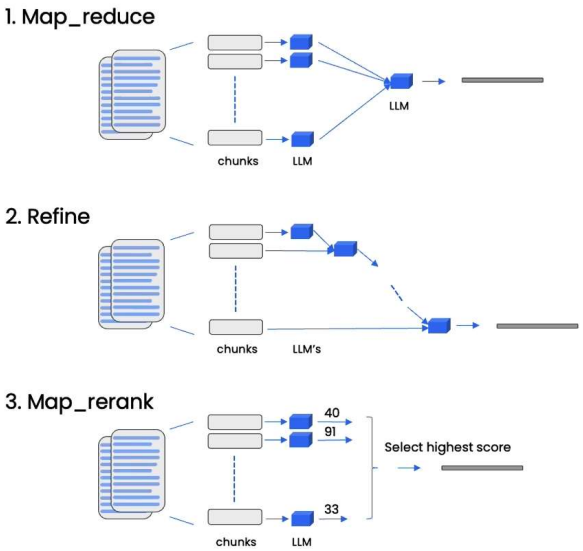




Pass the question and the documents retrieved, an LLM will generate an answer.

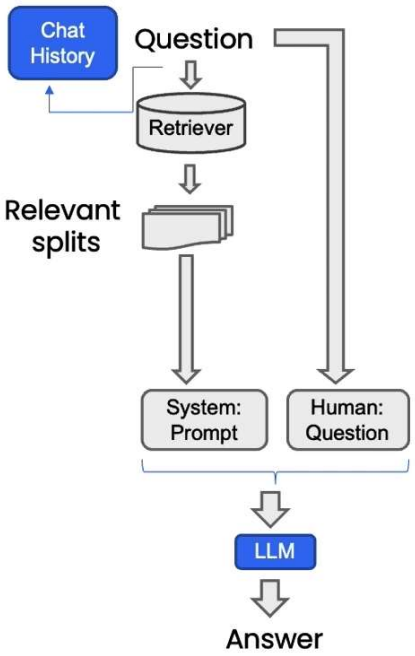
Sometime the document to pass are too long for the LLM token windows

Few methods available to overcome this issue: map_reduce, refine and map_rerank



Refine is longer but gives finer results as it builds on previous already analyzed documents.

Chat



Adding memory to all the chain used before

All components previously mentioned can be combined here

Conclusion

- How to use LangChain from a variety of docs
- Splitt the document into chunks
- Take the chunk and embed them into a vector store
- Retrievers are really fun algorithm!
- QA passing a question and receiving a answer
- Fully functioning chatbot by adding memory to the previous