TECHLENT

高阶大语言模型课程

Huajun Zeng

12/13/2024 - 1/31/2025
（12月27日和1月3日放假，共计6次课）
每周五 5pm-7pm PT / 8pm-10pm ET

TECHLENT

课程安排

Week	Date	Content	Week	Date	Content
1	2024-12-13	Retrieval Augmented Generation (RAG) for LLM ● Why augmenting LLMs? ● Methods for LLM augmentation ● Augmenting LLMs with retrieval ● Augmenting LLMs with fine tuning	4	2025-01-17	Pipeline for LLM Applications: From Code to Products ● Full stack LLM: tools needed for an LLM application ● Case study: build an LLM app from ground
2	2024-12-20	Chatbot Building with LLM APIs ● Environment setup ● Introduction to LLM APIs ● Using chat completion APIs ● Using fine tuning APIs	5	2025-01-24	More LLM Applications and Course Project ● Showcase of potential LLM applications for productivity, creativity and more ● More advanced LLM applications: AI Agent, Multi-modality, etc. ● Introduction to the course project: requirement and discussion
3	2025-01-10	Chatbot Building with LLM Frameworks and Vector Database ● Introduction to Langchain and LlamaIndex ● Case study: a chatbot from Langchain and vector database	6	2025-01-31	Project Presentation ● Student presentation on the course project

家庭作业回顾

- Using your private domain data (for example, a set of research papers, a set of financial report documents, …)
- Write a python program to answer questions about your data
  - Receive question from console and generate responses in streaming way
  - Using OpenAI SDK (or Anthropic)
- Refer to the examples in https://github.com/hzeng-otterai/chatbot-example/tree/main/backend_api

Comments 1

- Great to see you are using a notebook file for the homework!
- Good to see a brief description of the program before the code
- A good domain-specific document for testing the chatbot
- Good to use load_dotenv
- Multiple PDF -> text file, simple and effective implementation
- nest_asyncio.apply() seems a necessary step in Notebook environment! Something I didn't know!
- Domain specific queries looks great!

TECHLENT

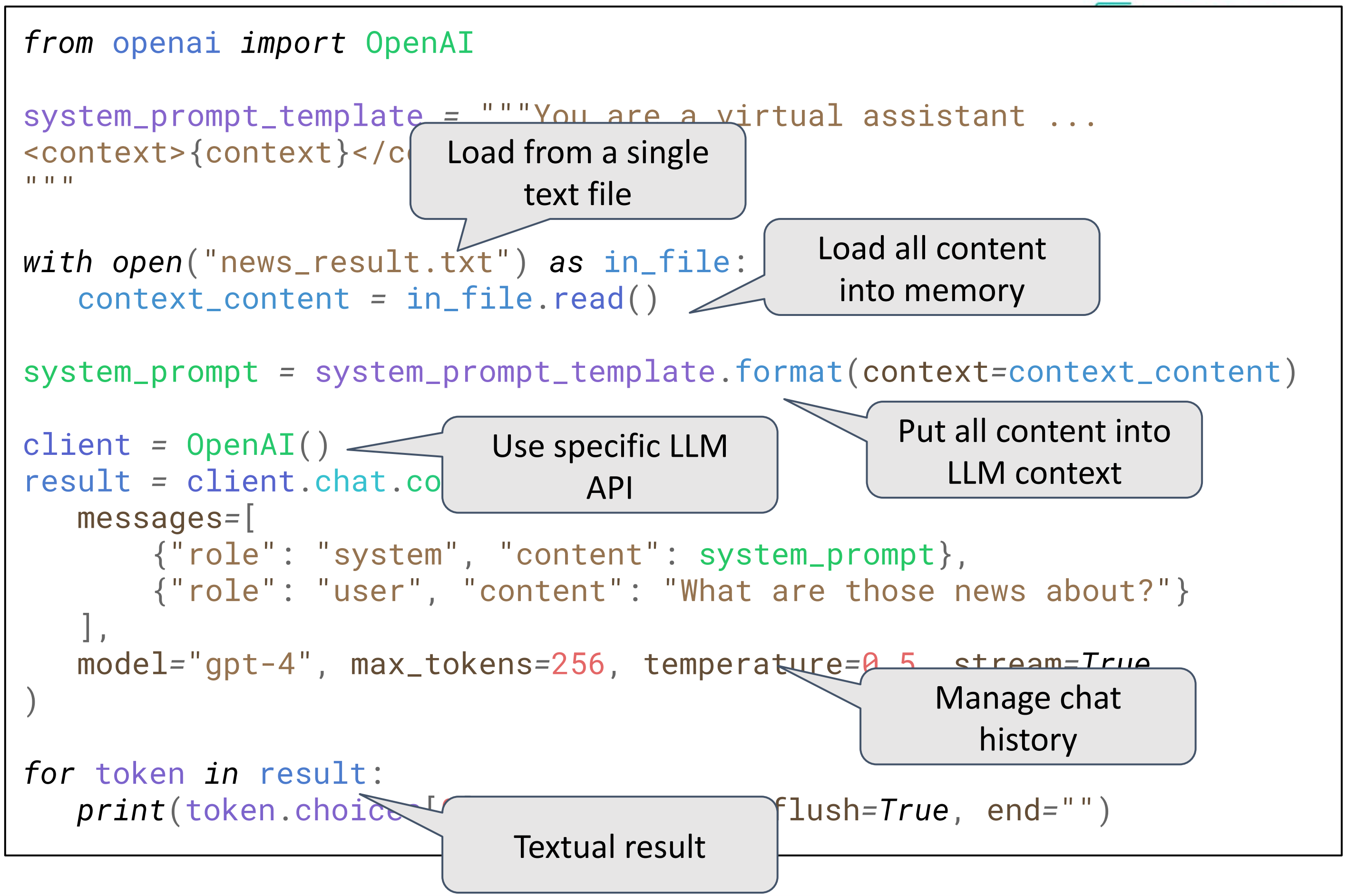
Comments 2

- Good to see using class to wrap the code, looks very clean!
- Good to implement trim_history function (minor improve suggestion: just return history[-self.max_history:] is sufficient)
- Consider use gpt-4o as the default model which is cheaper and more powerful than gpt-4
- Very good to use asyncio.timeout() to manage long running async tasks
- Extract text from PDF files, nice!
- There are some merge conflicts, see those files contains "<<<<<<< HEAD". These need to be resolved before commit to git.

第三课: 建立基于RAG的Chatbot应用

- Using LangChain
- Using Pinecone
- Advanced RAG

使用API调用LLM的缺点



```
from openai import OpenAI

system_prompt_template = """You are a virtual assistant ...
<context>{context}</c
"""

with open("news_result.txt") as in_file:
   context_content = in_file.read()

system_prompt = system_prompt_template.format(context=context_content)

client = OpenAI()
result = client.chat.co
   messages=[
       {"role": "system", "content": system_prompt},
       {"role": "user", "content": "What are those news about?"}
   ],
   model="gpt-4", max_tokens=256, temperature=0.5, stream=True
)

for token in result:
   print(token.choice                          flush=True, end="")
```

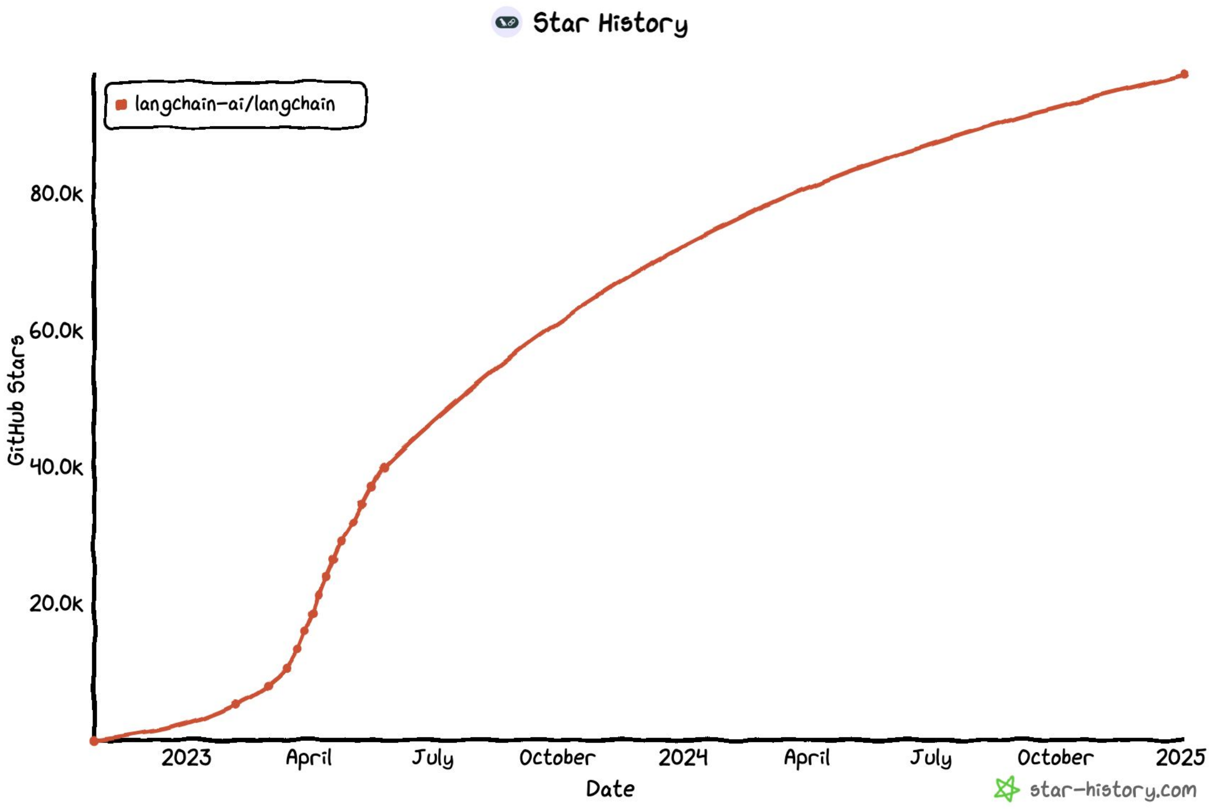
Load from a single text file
Load all content into memory
Put all content into LLM context
Use specific LLM API
Manage chat history
Textual result

为何要用Langchain框架

- Abstraction - LangChain provides an abstraction layer to interact with LLMs, vector DBs and other modules.
- Fast Prototyping - LangChain provides ready-to-use chains for a lot of applications.
- Built-In Instrumentation for enhanced observability and debuggability
- Langchain can also bring side effect because of its abstraction.

Langchain的历史

- Release date: October 2022
- Stars: 97.5k



LangChain代码示例

Refer to `test_00_langchain_example.py`

LangChain的管道操作符“｜”

A declarative way to easily compose chains together

- Streaming support
- Async support
- Optimized parallel execution

- Good for simple chains
- For complex chains such as branching, cycles, multiple agents, use LangGraph instead.

设置环境

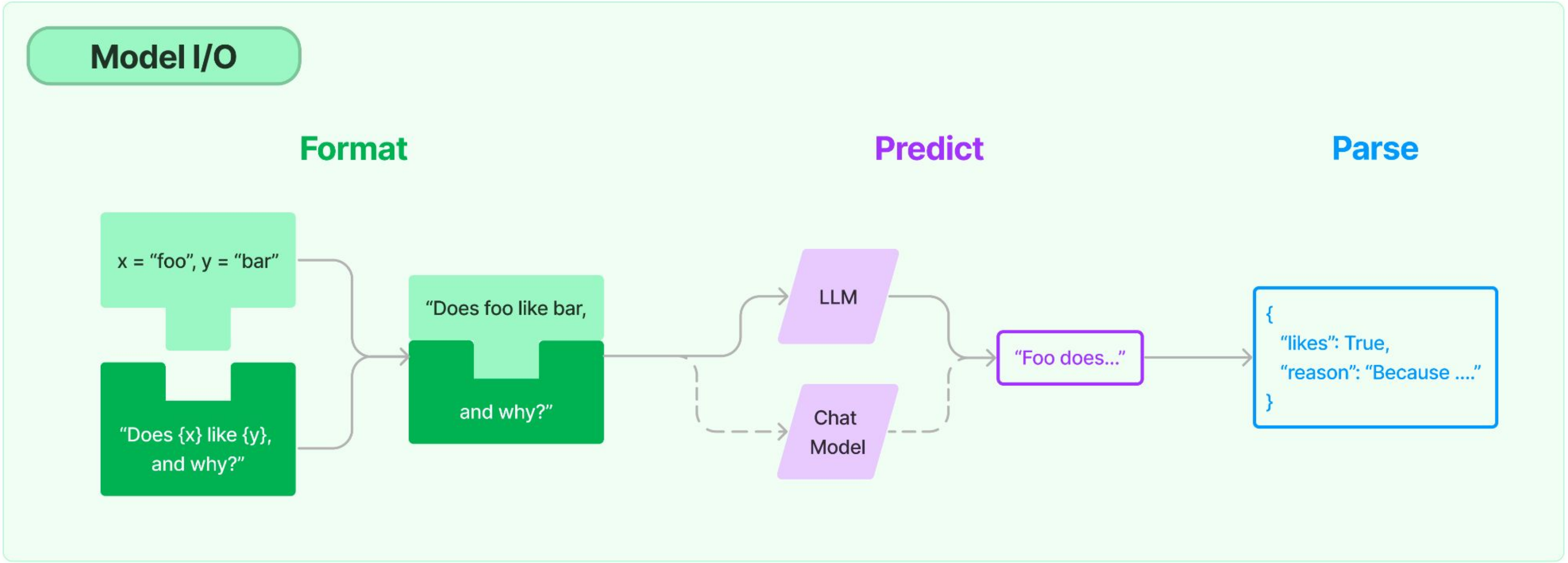
```
$ conda create -n langchain_chatbot_env python=3.11
$ cd chatbot-example/backend_langchain
$ pip install -r requirements.txt
$ export OPENAI_API_KEY=...
$ export PINECONE_API_KEY=...
```

LangChain的模块

- 模型输入输出: Interface with language models
- 检索: Interface with application-specific data
- 组合: Construct sequences of calls

TECHLENT

模型输入输出



调用LLM

Refer to test_01_chat.py

提示词模版

Refer to test_02_chat_with_prompt.py

Few-shot提示词模版

Refer to test_03_chat_with_few_shot_prompt.py

模版格式

f-string

```
fstring_template = "Tell me a {adjective} joke about {content}"
prompt = PromptTemplate.from_template(fstring_template)

prompt.format(adjective="funny", content="basketball")
```

jinja2

```
jinja2_template = "Tell me a {{ adjective }} joke about {{ content }}"
prompt = PromptTemplate.from_template(jinja2_template, template_format="jinja2")

prompt.format(adjective="funny", content="basketball")
```

输出结构解析

Refer to test_04_output_parser.py.

TECHLENT

输出结构解析

- Get format instructions: A method which returns a string containing instructions for how the output of a language model should be formatted.
- Parse: A method which takes in a string (assumed to be the response from a language model) and parses it into some structure.

Pydantic

- Data validation
- Data parsing
- Automatic docs

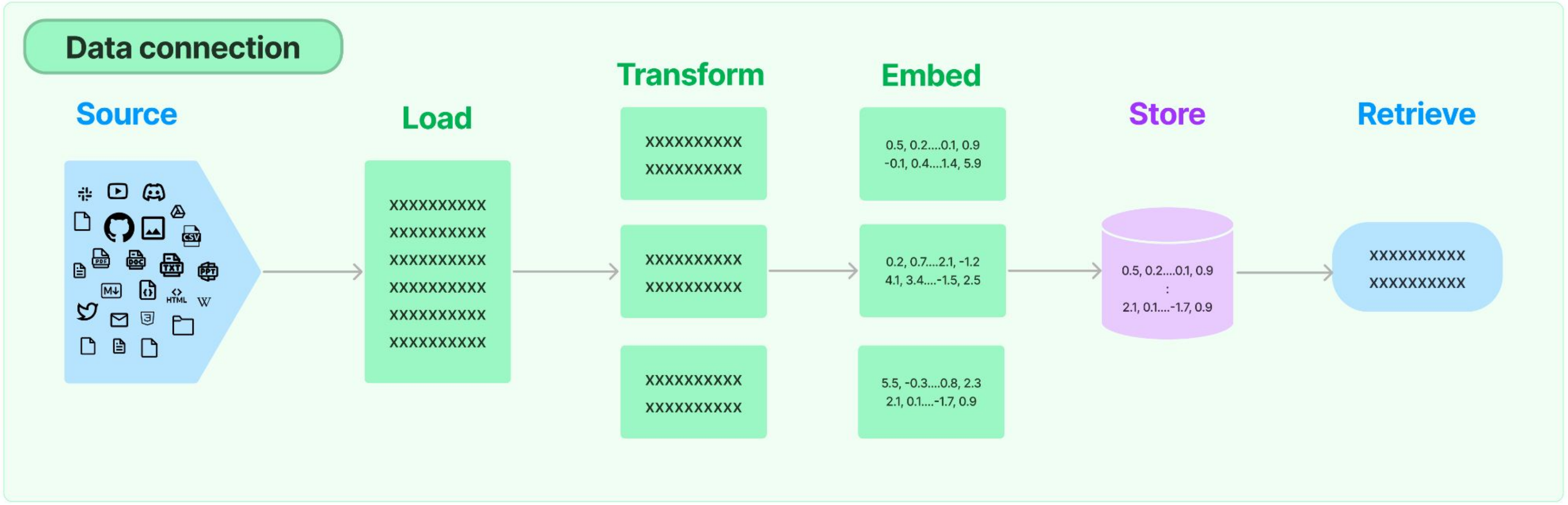
```python
from datetime import datetime
from pydantic import BaseModel, PositiveInt

class User(BaseModel):
  id: int
  name: str = 'John Doe'
  signup_ts: datetime | None
  tastes: dict[str, int]

external_data = {
  'id': 123,
  'signup_ts': '2019-06-01 12:22',
  'tastes': {
    'wine': 9,
    'cheese': 7
  }
}

user = User(**external_data)
```

检索


Data connection
Source
Load
Transform
Embed
Store
Retrieve
xxxxxxxxxx
0.5, 0.2....0.1, 0.9
-0.1, 0.4....1.4, 5.9
0.2, 0.7....2.1, -1.2
4.1, 3.4....-1.5, 2.5
5.5, -0.3....0.8, 2.3
2.1, 0.1....-1.7, 0.9
0.5, 0.2....0.1, 0.9
:
2.1, 0.1....-1.7, 0.9

TECHLENT

文档加载

- CSV
- File Directory
- HTML
- JSON
- Markdown
- PDF

```python
from langchain.document_loaders import TextLoader

loader = DirectoryLoader(
  '../',
  glob="**/*.md",
  loader_cls=TextLoader
)

docs = loader.load()
```

文档转换: Text splitters

- RecursiveCharacterTextSplitter
- It tries to split on them in order until the chunks are small enough. The default list is ["\n\n", "\n", " ", ""]

```
documents = TextLoader("./state_of_the_union.txt").load()

text_splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
  chunk_size=500,
  chunk_overlap=10
)

texts = text_splitter.split_documents(documents)
```

Embedding模型

- Embedding providers: OpenAI, Voyage, Hugging Face, etc.

```python
from langchain.embeddings import OpenAIEmbeddings

embeddings_model = OpenAIEmbeddings()

embeddings = embeddings_model.embed_documents([
  "Hi there!",
  "Oh, hello!",
  "What's your name?",
])

embedded_query = embeddings_model.embed_query("What is your name?")
```

向量数据库

- Local stores: Chroma, FAISS, etc.
- On-premis stores: Qdrant, Milvus, PGVector, etc.
- Cloud stores: Pinecone, etc.

```
raw_documents = TextLoader('./state_of_the_union.txt').load()
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
documents = text_splitter.split_documents(raw_documents)
db = Chroma.from_documents(documents, OpenAIEmbeddings())

query = "What did the president say about Ketanji Brown Jackson"
docs = db.similarity_search(query)
```

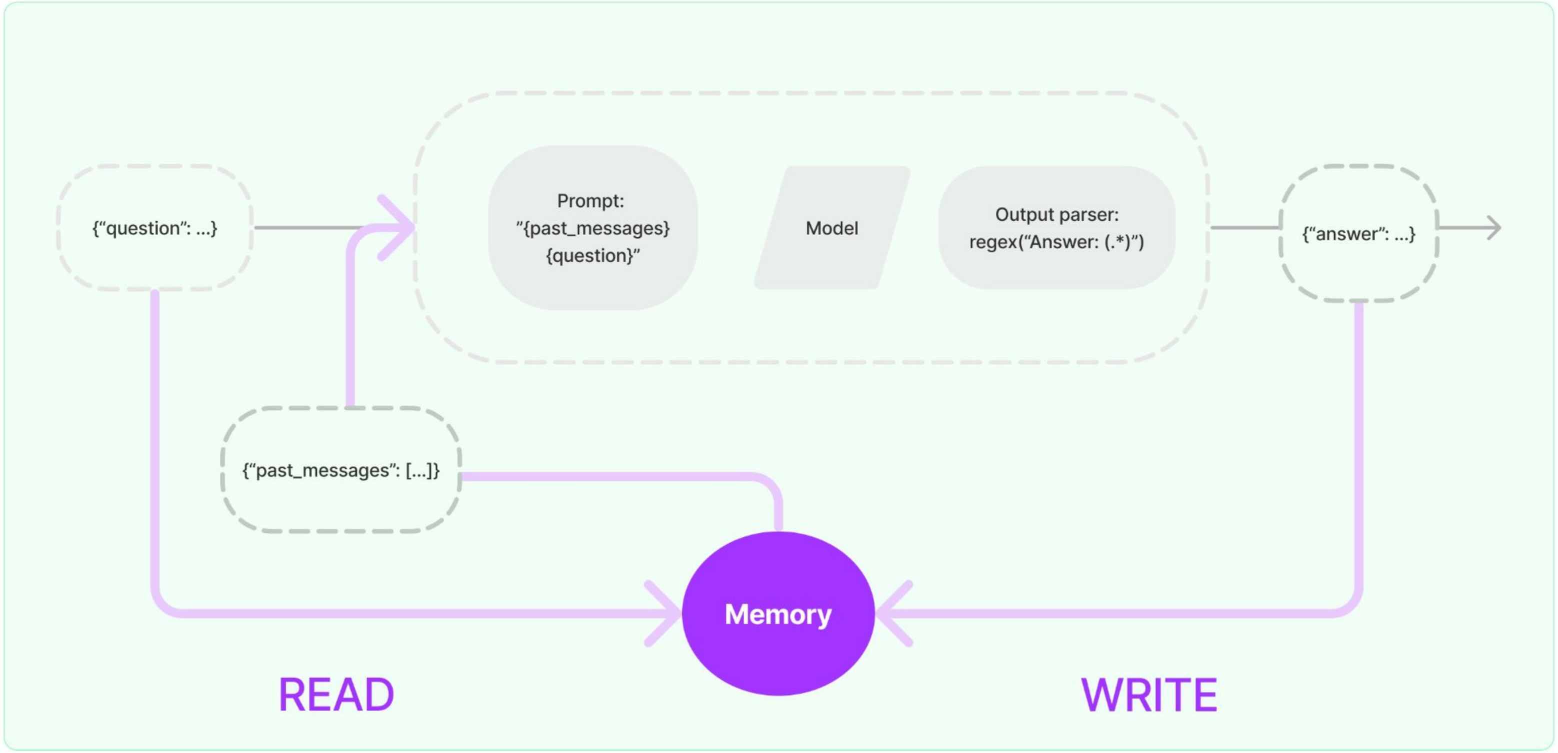
Retriever

- Retriever class returns Documents given a text query.
- It is more general than a vector store. A retriever does not need to be able to store documents, only to return (or retrieve) it. Vector stores can be used as the backbone of a retriever, but there are other types of retrievers as well.

```
db = FAISS.from_documents(texts, embeddings)

retriever = db.as_retriever()
docs = retriever.get_relevant_documents("what did he say about jackson")
```

Memory


{"question": ...}
Prompt: "{past_messages} {question}"
Model
Output parser: regex("Answer: (.*)")
{"answer": ...}
{"past_messages": [...]}
Memory
READ
WRITE

给Chatbot增加Memory

Refer to test_06_chat_with_memory.py

一个最小的Chatbot

Refer to test_07_chat_with_memory_rag_continous.py

使用Pinecone作为向量数据库

- For handling large scale of dataset
- For a hosted solution without a lot of DevOps overhead

Refer to test_08_pinecone_add_doc.py and test_09_pinecone.py

优化RAG: 基本优化技巧

- Prompt Engineering
- Embeddings
  - MTEB leaderboard
- Chunk Sizes
- Hybrid Search
  - Embedding search plus keyword search
- Metadata Filters
  - Filtering based on metadata of documents

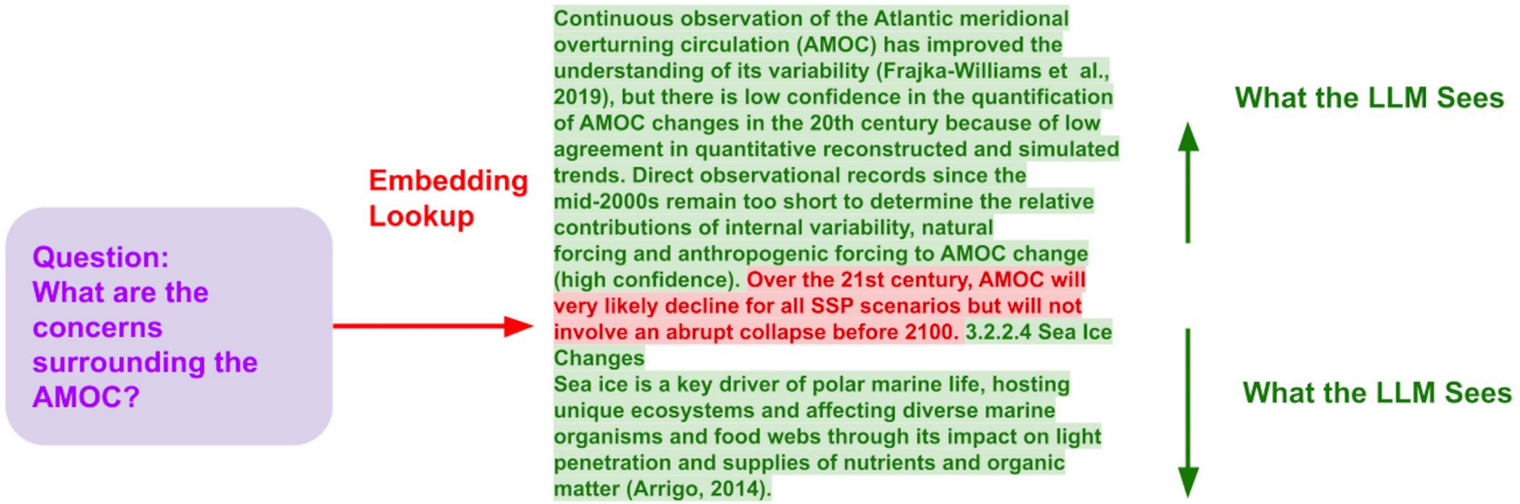
TECHLENT

优化RAG: 高级优化技巧

- Small-to-big retrieval
- Query transformation
- Reranking
- Recursive retrieval
- Embedded tables
- …

Small to Big Retrieval

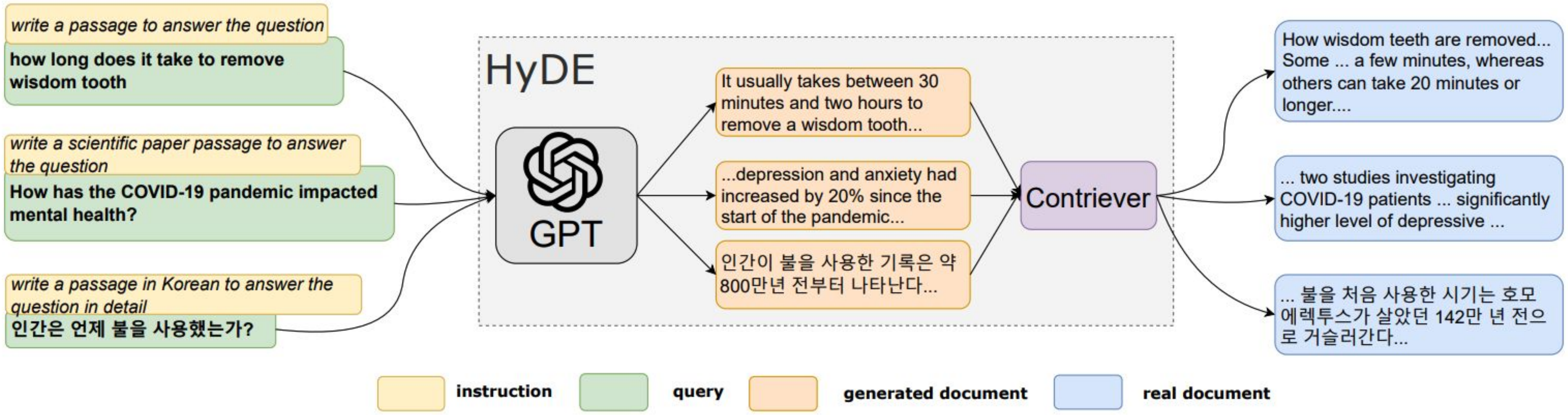
- Using smaller units for embedding but using expanded text for LLM inference
- Refer to test_10_pinecone_with_small_to_big.py



Continuous observation of the Atlantic meridional overturning circulation (AMOC) has improved the understanding of its variability (Frajka-Williams et al., 2019), but there is low confidence in the quantification of AMOC changes in the 20th century because of low agreement in quantitative reconstructed and simulated trends. Direct observational records since the mid-2000s remain too short to determine the relative contributions of internal variability, natural forcing and anthropogenic forcing to AMOC change (high confidence). Over the 21st century, AMOC will very likely decline for all SSP scenarios but will not involve an abrupt collapse before 2100. 3.2.2.4 Sea Ice Changes
Sea ice is a key driver of polar marine life, hosting unique ecosystems and affecting diverse marine organisms and food webs through its impact on light penetration and supplies of nutrients and organic matter (Arrigo, 2014).

Query Transformation - HyDE

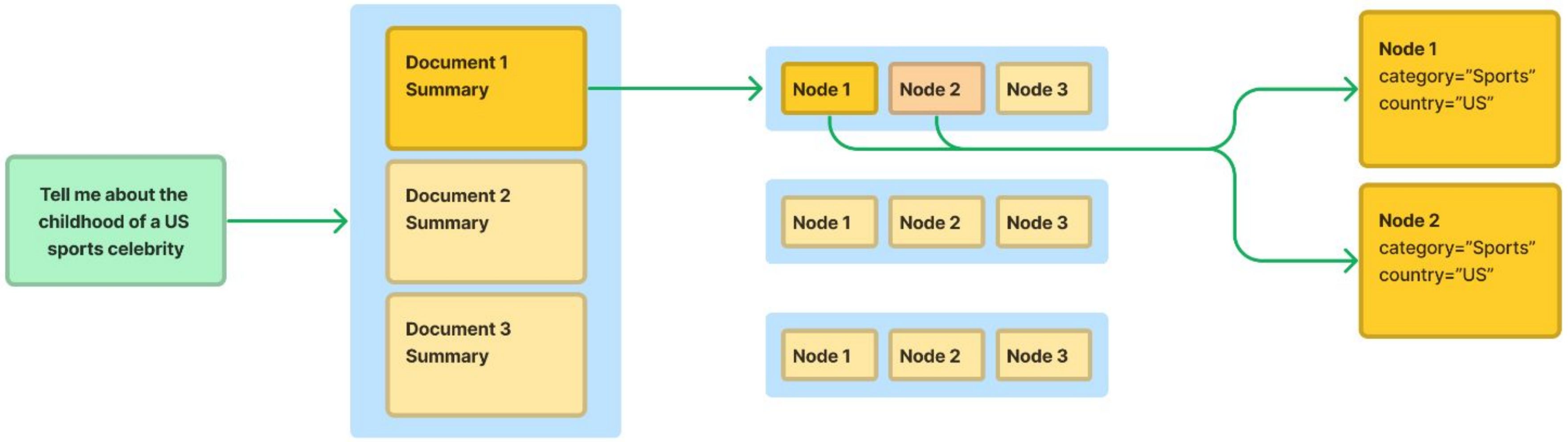
- HyDE: Hypothetical Document Embeddings
- Refer to test_11_pinecone_with_hyde.py



write a passage to answer the question
how long does it take to remove wisdom tooth
write a scientific paper passage to answer the question
How has the COVID-19 pandemic impacted mental health?
write a passage in Korean to answer the question in detail
인간은 언제 불을 사용했는가?
HyDE
GPT
It usually takes between 30 minutes and two hours to remove a wisdom tooth...
...depression and anxiety had increased by 20% since the start of the pandemic...
인간이 불을 사용한 기록은 약 800만년 전부터 나타난다...
Contriever
How wisdom teeth are removed... Some ... a few minutes, whereas others can take 20 minutes or longer....
... two studies investigating COVID-19 patients ... significantly higher level of depressive ...
... 불을 처음 사용한 시기는 호모 에렉투스가 살았던 142만 년 전으로 거슬러간다...
instruction
query
generated document
real document

Recursive Retrieval

Document Hierarchies (Summaries + Raw Chunks) + Recursive Retrieval


Tell me about the childhood of a US sports celebrity
Document 1 Summary
Document 2 Summary
Document 3 Summary
Node 1
Node 2
Node 3
Node 1
Node 2
Node 3
Node 1
Node 2
Node 3
Node 1 category="Sports" country="US"
Node 2 category="Sports" country="US"

Reranking

- A step to rerank the top n results returned from the embedding search
- Why reranking
  - Embedding model might be not strong enough
  - To use context information or user-specific information
- Choosing a reranking model
  - LLMRerank
  - Cohere Rerank
  - SentenceTransformerRerank
  - …

Homework

- Implement a RAG-based Chatbot using Langchain
  - A console program to continuously receive user input and respond
  - Try smaller dataset first and then larger dataset
  - Refer to the examples in https://github.com/hzeng-otterai/chatbot-example/tree/main/backend_langchain

TECHLENT

Questions?