Retrieve Documents

We recommend LangChain's OpenAIEmbeddings class to embed queries and retrieve documents from Astra DB Serverless.

Basic retrieval

1. This retriever uses the OpenAIEmbeddings class to embed the query and retrieve the ten most similar documents from the Astra DB Serverless vector store. The `(search_kwargs={'k': 10})` parameter means that when a query is performed against the database, the retriever will return the top 10 closest matches based on the vector embeddings.

Python | Result

```
import os
from dotenv import load_dotenv
from langchain_astradb import AstraDBVectorStore
from langchain_openai import OpenAIEmbeddings
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import ChatOpenAI
from langchain.output_parsers import StrOutputParser

load_dotenv()

OPEN_AI_API_KEY = os.environ["OPENAI_API_KEY"]

vstore = AstraDBVectorStore(
    embedding=OpenAIEmbeddings(openai_api_key=OPEN_AI_API_KEY),
    collection_name="test",
    token=os.environ["ASTRA_DB_APPLICATION_TOKEN"],
    api_endpoint=os.environ["ASTRA_DB_API_ENDPOINT"],
)
retriever = vstore.as_retriever(search_kwargs={'k': 10})

prompt_template = """
Answer the question based only on the supplied context. If you don't k
Context: {context}
Question: {question}
Your answer:
"""
```

```
prompt = ChatPromptTemplate.from_template(prompt_template)
model = ChatOpenAI(openai_api_key=OPEN_AI_API_KEY, model_name="gpt-3.5

chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | model
    | StrOutputParser()
)

response = chain.invoke("Can you summarize the given context?")
print(response)
```

Retrieval with multiple questions

Build an iterative retriever to ask multiple questions.

1. Create a basic prompt template and a set of questions.

```
prompt_template = """
Answer the question based only on the supplied context. If you don't know th
Context: {context}
Question: {question}
Your answer:
"""

prompt = ChatPromptTemplate.from_template(prompt_template)

questions = [
    "What motivates the narrator, Montresor, to seek revenge against Fortuna
    "What are the major themes in this story?",
    "What is the significance of the story taking place during the carnival
    "How is vivid and descriptive language used in the story?",
    "Is there any foreshadowing in the story? If yes, how is it used in the
]
```

2. Create a helper method to iterate over the questions.

```python
def do_retrieval(chain):
    for i in range(len(questions)):
        print("-" * 40)
        print(f"Question: {questions[i]}\n")
        with get_openai_callback() as cb:
            pprint_result(chain.invoke(questions[i]))
            print(f'\nTotal Tokens: {cb.total_tokens}\n')
```

3. Run the retrieval against your vector store.

Python | Result

```python
base_retriever = vstore.as_retriever(search_kwargs={'k': 10})
model = ChatOpenAI(openai_api_key=OPEN_AI_API_KEY, model_name="gpt-3.5

base_chain = (
    {"context": base_retriever, "question": RunnablePassthrough()}
    | prompt
    | model
    | StrOutputParser()
)

do_retrieval(base_chain)
```