Retrieval QA using OpenAI functions

OpenAI functions allows for structuring of response output. This is often useful in question answering when you want to not only get the final answer but also supporting evidence, citations, etc.

In this notebook we show how to use an LLM chain which uses OpenAI functions as part of an overall retrieval pipeline.

```
from langchain.chains import RetrievalQA
from langchain.document_loaders import TextLoader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import Chroma
```

```
loader = TextLoader("../../state_of_the_union.txt", encoding="utf-8")
documents = loader.load()
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
texts = text_splitter.split_documents(documents)
for i, text in enumerate(texts):
    text.metadata['source'] = f"{i}-pl"
embeddings = OpenAIEmbeddings()
docsearch = Chroma.from_documents(texts, embeddings)
```

```
from langchain.chat_models import ChatOpenAI
from langchain.chains.combine_documents.stuff import StuffDocumentsChain
from langchain.prompts import PromptTemplate
from langchain.chains import create_qa_with_sources_chain
```

```
llm = ChatOpenAI(temperature=0, model="gpt-3.5-turbo-0613")
```

```
qa_chain = create_qa_with_sources_chain(llm)
```

```
doc_prompt = PromptTemplate(
    template="Content: {page_content}\nSource: {source}",
    input_variables=["page_content", "source"],
)
```

```
final_qa_chain = StuffDocumentsChain(
    llm_chain=qa_chain,
    document_variable_name='context',
    document_prompt=doc_prompt,
)
```

```
retrieval_qa = RetrievalQA(
    retriever=docsearch.as_retriever(),
    combine_documents_chain=final_qa_chain
)
```

```
query = "What did the president say about russia"
```

```
retrieval_qa.run(query)
```

```
    '{\n  "answer": "The President expressed strong condemnation of Russia\'s actions in Ukraine and announced measures to isolate Russia and provide support to Ukraine. He stated that Russia\'s invasion of Ukraine will have long-term consequences for Russia and emphasized the commitment to defend NATO countries. The President also mentioned taking robust action through sanctions and releasing oil reserves to mitigate gas prices. Overall, the President conveyed a message of solidarity with Ukraine and determination to protect American interests.",\n  "sources": ["0-pl", "4-pl", "5-pl", "6-pl"]\n}'
```

Using Pydantic

If we want to, we can set the chain to return in Pydantic. Note that if downstream chains consume the output of this chain - including memory - they will generally expect it to be in string format, so you should only use this chain when it is the final chain.

```
qa_chain_pydantic = create_qa_with_sources_chain(llm, output_parser="pydantic")
```

```
final_qa_chain_pydantic = StuffDocumentsChain(
    llm_chain=qa_chain_pydantic,
    document_variable_name='context',
    document_prompt=doc_prompt,
)
```

```
retrieval_qa_pydantic = RetrievalQA(
    retriever=docsearch.as_retriever(),
    combine_documents_chain=final_qa_chain_pydantic
)
```

```
retrieval_qa_pydantic.run(query)
```

```
    AnswerWithSources(answer="The President expressed strong condemnation of Russia's actions in Ukraine and announced measures to isolate Russia and provide support to Ukraine. He stated that Russia's invasion of Ukraine will have long-term consequences for Russia and emphasized the commitment to defend NATO countries. The President also mentioned taking robust action through sanctions and releasing oil reserves to mitigate gas prices. Overall, the President conveyed a message of solidarity with Ukraine and determination to protect American interests.", sources=['0-pl', '4-pl', '5-pl', '6-pl'])
```

Using in ConversationalRetrievalChain

We can also show what it's like to use this in the ConversationalRetrievalChain. Note that because this chain involves memory, we will NOT use the Pydantic return type.

```
from langchain.chains import ConversationalRetrievalChain
from langchain.memory import ConversationBufferMemory
from langchain.chains import LLMChain
memory = ConversationBufferMemory(memory_key="chat_history", return_messages=True)
_template = """Given the following conversation and a follow up question, rephrase the follow up question to
be a standalone question, in its original language.\
Make sure to avoid using any unclear pronouns.

Chat History:
{chat_history}
```

```
Follow Up Input: {question}
Standalone question:"""
CONDENSE_QUESTION_PROMPT = PromptTemplate.from_template(_template)
condense_question_chain = LLMChain(
    llm=llm,
    prompt=CONDENSE_QUESTION_PROMPT,
)
```

```
qa = ConversationalRetrievalChain(
    question_generator=condense_question_chain,
    retriever=docsearch.as_retriever(),
    memory=memory,
    combine_docs_chain=final_qa_chain
)
```

```
query = "What did the president say about Ketanji Brown Jackson"
result = qa({"question": query})
```

```
result
```

```
    {'question': 'What did the president say about Ketanji Brown Jackson',
     'chat_history': [HumanMessage(content='What did the president say about Ketanji Brown Jackson',
additional_kwargs={}, example=False),
      AIMessage(content='{\n  "answer": "The President nominated Ketanji Brown Jackson as a Circuit Court of
Appeals Judge and praised her as one of the nation\'s top legal minds who will continue Justice Breyer\'s
legacy of excellence.",\n  "sources": ["31-pl"]\n}', additional_kwargs={}, example=False)],
     'answer': '{\n  "answer": "The President nominated Ketanji Brown Jackson as a Circuit Court of Appeals
Judge and praised her as one of the nation\'s top legal minds who will continue Justice Breyer\'s legacy of
excellence.",\n  "sources": ["31-pl"]\n}'}
```

```
query = "what did he say about her predecessor?"
result = qa({"question": query})
```

```
result
```

```
    {'question': 'what did he say about her predecessor?',
     'chat_history': [HumanMessage(content='What did the president say about Ketanji Brown Jackson',
additional_kwargs={}, example=False),
      AIMessage(content='{\n  "answer": "The President nominated Ketanji Brown Jackson as a Circuit Court of
Appeals Judge and praised her as one of the nation\'s top legal minds who will continue Justice Breyer\'s
legacy of excellence.",\n  "sources": ["31-pl"]\n}', additional_kwargs={}, example=False),
      HumanMessage(content='what did he say about her predecessor?', additional_kwargs={}, example=False),
      AIMessage(content='{\n  "answer": "The President honored Justice Stephen Breyer for his service as an
Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court.",\n
"sources": ["31-pl"]\n}', additional_kwargs={}, example=False)],
     'answer': '{\n  "answer": "The President honored Justice Stephen Breyer for his service as an Army
veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court.",\n  "sources":
["31-pl"]\n}'}
```

Using your own output schema

We can change the outputs of our chain by passing in our own schema. The values and descriptions of this schema will inform the function we pass to the OpenAI API, meaning it won't just affect how we parse outputs but will also change the OpenAI output itself.

For example we can add a `countries_referenced` parameter to our schema and describe what we want this parameter to mean, and that'll cause the OpenAI output to include a description of a speaker in the response.

In addition to the previous example, we can also add a custom prompt to the chain. This will allow you to add additional context to the response, which can be useful for question answering.

```
from typing import List

from pydantic import BaseModel, Field

from langchain.chains.openai_functions import create_qa_with_structure_chain

from langchain.prompts.chat import ChatPromptTemplate, HumanMessagePromptTemplate
from langchain.schema import SystemMessage, HumanMessage
```

```
class CustomResponseSchema(BaseModel):
    """An answer to the question being asked, with sources."""

    answer: str = Field(..., description="Answer to the question that was asked")
    countries_referenced: List[str] = Field(..., description="All of the countries mentioned in the sources")
    sources: List[str] = Field(
        ..., description="List of sources used to answer the question"
    )


prompt_messages = [
    SystemMessage(
        content=(
            "You are a world class algorithm to answer "
            "questions in a specific format."
        )
    ),
    HumanMessage(content="Answer question using the following context"),
    HumanMessagePromptTemplate.from_template("{context}"),
    HumanMessagePromptTemplate.from_template("Question: {question}"),
    HumanMessage(content="Tips: Make sure to answer in the correct format. Return all of the countries
mentioned in the sources in uppercase characters."),
]

chain_prompt = ChatPromptTemplate(messages=prompt_messages)

qa_chain_pydantic = create_qa_with_structure_chain(llm, CustomResponseSchema, output_parser="pydantic",
prompt=chain_prompt)
final_qa_chain_pydantic = StuffDocumentsChain(
    llm_chain=qa_chain_pydantic,
    document_variable_name='context',
    document_prompt=doc_prompt,
)
retrieval_qa_pydantic = RetrievalQA(
    retriever=docsearch.as_retriever(),
    combine_documents_chain=final_qa_chain_pydantic
)
query = "What did he say about russia"
retrieval_qa_pydantic.run(query)
```

```
    CustomResponseSchema(answer="He announced that American airspace will be closed off to all Russian
flights, further isolating Russia and adding an additional squeeze on their economy. The Ruble has lost 30%
of its value and the Russian stock market has lost 40% of its value. He also mentioned that Putin alone is to
blame for Russia's reeling economy. The United States and its allies are providing support to Ukraine in
their fight for freedom, including military, economic, and humanitarian assistance. The United States is
giving more than $1 billion in direct assistance to Ukraine. He made it clear that American forces are not
engaged and will not engage in conflict with Russian forces in Ukraine, but they are deployed to defend NATO
allies in case Putin decides to keep moving west. He also mentioned that Putin's attack on Ukraine was
premeditated and unprovoked, and that the West and NATO responded by building a coalition of freedom-loving
nations to confront Putin. The free world is holding Putin accountable through powerful economic sanctions,
cutting off Russia's largest banks from the international financial system, and preventing Russia's central
bank from defending the Russian Ruble. The U.S. Department of Justice is also assembling a task force to go
after the crimes of Russian oligarchs.", countries_referenced=['AMERICA', 'RUSSIA', 'UKRAINE'], sources=['4-
pl', '5-pl', '2-pl', '3-pl'])
```