Retrieval Question Answering with Sources

Contents

This notebook goes over how to do question-answering with sources over an Index. It does this by using the `RetrievalQAWithSourcesChain`, which does the lookup of the documents from an Index.

```
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.embeddings.cohere import CohereEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores.elastic_vector_search import ElasticVectorSearch
from langchain.vectorstores import Chroma
```

```
with open("../../state_of_the_union.txt") as f:
    state_of_the_union = f.read()
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
texts = text_splitter.split_text(state_of_the_union)

embeddings = OpenAIEmbeddings()
```

```
docsearch = Chroma.from_texts(texts, embeddings, metadatas=[{"source": f"{i}-pl"}
for i in range(len(texts))])
```

```
Running Chroma using direct local API.
Using DuckDB in-memory for database. Data will be transient.
```

```
from langchain.chains import RetrievalQAWithSourcesChain
```

Skip to main content

```
from langchain import OpenAI

chain = RetrievalQAWithSourcesChain.from_chain_type(OpenAI(temperature=0),
chain_type="stuff", retriever=docsearch.as_retriever())
```

```
chain({"question": "What did the president say about Justice Breyer"},
return_only_outputs=True)
```

```
{'answer': ' The president honored Justice Breyer for his service and mentioned
his legacy of excellence.\n',
 'sources': '31-pl'}
```

Chain Type

You can easily specify different chain types to load and use in the RetrievalQAWithSourcesChain chain. For a more detailed walkthrough of these types, please see this notebook.

There are two ways to load different chain types. First, you can specify the chain type argument in the `from_chain_type` method. This allows you to pass in the name of the chain type you want to use. For example, in the below we change the chain type to `map_reduce`.

```
chain = RetrievalQAWithSourcesChain.from_chain_type(OpenAI(temperature=0),
chain_type="map_reduce", retriever=docsearch.as_retriever())
```

```
chain({"question": "What did the president say about Justice Breyer"},
return_only_outputs=True)
```

```
{'answer': ' The president said "Justice Breyer—an Army veteran, Constitutional
scholar, and retiring Justice of the United States Supreme Court. Justice Breyer,
thank you for your service."\n',
 'sources': '31-pl'}
```

The above way allows you to really simply change the chain_type, but it does provide a ton of flexibility over parameters to that chain type. If you want to control those parameters, you can load the chain directly (as you did in this notebook) and then pass that directly to the the RetrievalQAWithSourcesChain chain with the `combine_document_chain` parameter. For

Skip to main content

```
from langchain.chains.qa_with_sources import load_qa_with_sources_chain
qa_chain = load_qa_with_sources_chain(OpenAI(temperature=0), chain_type="stuff")
qa = RetrievalQAWithSourcesChain(combine_documents_chain=qa_chain,
retriever=docsearch.as_retriever())
```

```
qa({"question": "What did the president say about Justice Breyer"},
return_only_outputs=True)
```

```
{'answer': ' The president honored Justice Breyer for his service and mentioned
his legacy of excellence.\n',
 'sources': '31-pl'}
```