Chat Index

Contents

This notebook goes over how to set up a chain to chat with an index. The only difference between this chain and the RetrievalQAChain is that this allows for passing in of a chat history which can be used to allow for follow up questions.

```
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import Chroma
from langchain.text_splitter import CharacterTextSplitter
from langchain.llms import OpenAI
from langchain.chains import ConversationalRetrievalChain
```

Load in documents. You can replace this with a loader for whatever type of data you want

```
from langchain.document_loaders import TextLoader
loader = TextLoader("../../state_of_the_union.txt")
documents = loader.load()
```

If you had multiple loaders that you wanted to combine, you do something like:

```
# loaders = [....]
# docs = []
# for loader in loaders:
#     docs.extend(loader.load())
```

Skip to main content

We now split the documents, create embeddings for them, and put them in a vectorstore. This allows us to do semantic search over them.

```
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
documents = text_splitter.split_documents(documents)

embeddings = OpenAIEmbeddings()
vectorstore = Chroma.from_documents(documents, embeddings)
```

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

We now initialize the ConversationalRetrievalChain

```
qa = ConversationalRetrievalChain.from_llm(OpenAI(temperature=0),
vectorstore.as_retriever())
```

Here's an example of asking a question with no chat history

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

```
result["answer"]
```

```
" The president said that Ketanji Brown Jackson is one of the nation's top legal
minds, a former top litigator in private practice, a former federal public
defender, and from a family of public school educators and police officers. He
also said that she is a consensus builder and has received a broad range of
support from the Fraternal Order of Police to former judges appointed by Democrats
and Republicans."
```

Here's an example of asking a question with some chat history

```
chat_history = [(query, result["answer"])]
query = "Did he mention who she suceeded"
result = qa({"question": query, "chat_history": chat_history})
```

Skip to main content

```
result['answer']
```

```
' Justice Stephen Breyer'
```

Return Source Documents

You can also easily return source documents from the ConversationalRetrievalChain. This is useful for when you want to inspect what documents were returned.

```
qa = ConversationalRetrievalChain.from_llm(OpenAI(temperature=0),
vectorstore.as_retriever(), return_source_documents=True)
```

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

```
result['source_documents'][0]
```

```
Document(page_content='Tonight. I call on the Senate to: Pass the Freedom to Vote
Act. Pass the John Lewis Voting Rights Act. And while you're at it, pass the
Disclose Act so Americans can know who is funding our elections. \n\nTonight, I'd
like to honor someone who has dedicated his life to serve this country: Justice
Stephen Breyer—an Army veteran, Constitutional scholar, and retiring Justice of
the United States Supreme Court. Justice Breyer, thank you for your service.
\n\nOne of the most serious constitutional responsibilities a President has is
nominating someone to serve on the United States Supreme Court. \n\nAnd I did that
4 days ago, when I nominated Circuit Court of Appeals Judge Ketanji Brown Jackson.
One of our nation's top legal minds, who will continue Justice Breyer's legacy of
excellence.', lookup_str='', metadata={'source': '../../state_of_the_union.txt'},
lookup_index=0)
```

ConversationalRetrievalChain with `search_distance`

If you are using a vector store that supports filtering by search distance, you can add a

Skip to main content

```
vectordbkwargs = {"search_distance": 0.9}
```

```
qa = ConversationalRetrievalChain.from_llm(OpenAI(temperature=0),
vectorstore.as_retriever(), return_source_documents=True)
chat_history = []
query = "What did the president say about Ketanji Brown Jackson"
result = qa({"question": query, "chat_history": chat_history, "vectordbkwargs":
vectordbkwargs})
```

ConversationalRetrievalChain with `map_reduce`

We can also use different types of combine document chains with the ConversationalRetrievalChain chain.

```
from langchain.chains import LLMChain
from langchain.chains.question_answering import load_qa_chain
from langchain.chains.chat_index.prompts import CONDENSE_QUESTION_PROMPT
```

```
llm = OpenAI(temperature=0)
question_generator = LLMChain(llm=llm, prompt=CONDENSE_QUESTION_PROMPT)
doc_chain = load_qa_chain(llm, chain_type="map_reduce")

chain = ConversationalRetrievalChain(
    retriever=vectorstore.as_retriever(),
    question_generator=question_generator,
    combine_docs_chain=doc_chain,
)
```

```
chat_history = []
query = "What did the president say about Ketanji Brown Jackson"
result = chain({"question": query, "chat_history": chat_history})
```

```
result['answer']
```

```
" The president said that Ketanji Brown Jackson is one of the nation's top legal
minds, a former top litigator in private practice, a former federal public
defender, from a family of public school educators and police officers, a
consensus builder, and has received a broad range of support from the Fraternal
```

Skip to main content

ConversationalRetrievalChain with Question Answering with sources

You can also use this chain with the question answering with sources chain.

```
from langchain.chains.qa_with_sources import load_qa_with_sources_chain
```

```
llm = OpenAI(temperature=0)
question_generator = LLMChain(llm=llm, prompt=CONDENSE_QUESTION_PROMPT)
doc_chain = load_qa_with_sources_chain(llm, chain_type="map_reduce")

chain = ConversationalRetrievalChain(
    retriever=vectorstore.as_retriever(),
    question_generator=question_generator,
    combine_docs_chain=doc_chain,
)
```

```
chat_history = []
query = "What did the president say about Ketanji Brown Jackson"
result = chain({"question": query, "chat_history": chat_history})
```

```
result['answer']
```

```
" The president said that Ketanji Brown Jackson is one of the nation's top legal
minds, a former top litigator in private practice, a former federal public
defender, from a family of public school educators and police officers, a
consensus builder, and has received a broad range of support from the Fraternal
Order of Police to former judges appointed by Democrats and Republicans.
\nSOURCES: ../../state_of_the_union.txt"
```

ConversationalRetrievalChain with streaming to `stdout`

Output from the chain will be streamed to `stdout` token by token in this example.

Skip to main content

```
from langchain.chains.llm import LLMChain
from langchain.callbacks.base import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
from langchain.chains.chat_index.prompts import CONDENSE_QUESTION_PROMPT, QA_PROMPT
from langchain.chains.question_answering import load_qa_chain

# Construct a ChatVectorDBChain with a streaming llm for combine docs
# and a separate, non-streaming llm for question generation
llm = OpenAI(temperature=0)
streaming_llm = OpenAI(streaming=True,
callback_manager=CallbackManager([StreamingStdOutCallbackHandler()]),
verbose=True, temperature=0)

question_generator = LLMChain(llm=llm, prompt=CONDENSE_QUESTION_PROMPT)
doc_chain = load_qa_chain(streaming_llm, chain_type="stuff", prompt=QA_PROMPT)

qa = ConversationalRetrievalChain(
    retriever=vectorstore.as_retriever(), combine_docs_chain=doc_chain,
question_generator=question_generator)
```

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

```
 The president said that Ketanji Brown Jackson is one of the nation's top legal
minds, a former top litigator in private practice, a former federal public
defender, and from a family of public school educators and police officers. He
also said that she is a consensus builder and has received a broad range of
support from the Fraternal Order of Police to former judges appointed by Democrats
and Republicans.
```

```
chat_history = [(query, result["answer"])]
query = "Did he mention who she suceeded"
result = qa({"question": query, "chat_history": chat_history})
```

```
 Justice Stephen Breyer
```

get_chat_history Function

You can also specify a `get_chat_history` function, which can be used to format the chat_history string.

Skip to main content

```
def get_chat_history(inputs) -> str:
    res = []
    for human, ai in inputs:
        res.append(f"Human:{human}\nAI:{ai}")
    return "\n".join(res)
qa = ConversationalRetrievalChain.from_llm(OpenAI(temperature=0), vectorstore,
get_chat_history=get_chat_history)
```

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

```
result['answer']
```

```
" The president said that Ketanji Brown Jackson is one of the nation's top legal
minds, a former top litigator in private practice, a former federal public
defender, and from a family of public school educators and police officers. He
also said that she is a consensus builder and has received a broad range of
support from the Fraternal Order of Police to former judges appointed by Democrats
and Republicans."
```