Document QA

Here we walk through how to use LangChain for question answering over a list of documents. Under the hood we'll be using our Document chains.

Prepare Data

First we prepare the data. For this example we do similarity search over a vector database, but these documents could be fetched in any manner (the point of this notebook to highlight what to do AFTER you fetch the documents).

```
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import Chroma
from langchain.docstore.document import Document
from langchain.prompts import PromptTemplate
from langchain.indexes.vectorstore import VectorstoreIndexCreator
```

```
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": str(i)} for i in
range(len(texts))]).as_retriever()
```

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

```
query = "What did the president say about Justice Breyer"
docs = docsearch.get_relevant_documents(query)
```

```
from langchain.chains.question_answering import load_qa_chain
from langchain.llms import OpenAI
```

Quickstart

If you just want to get started as quickly as possible, this is the recommended way to do it:

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="stuff")
query = "What did the president say about Justice Breyer"
chain.run(input_documents=docs, question=query)
```

```
    ' The president said that Justice Breyer has dedicated his life to serve the country and thanked him for
his service.'
```

If you want more control and understanding over what is happening, please see the information below.

The `stuff` Chain

This sections shows results of using the `stuff` Chain to do question answering.

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="stuff")
```

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

```
    {'output_text': ' The president said that Justice Breyer has dedicated his life to serve the country and thanked him for his service.'}
```

Custom Prompts

You can also use your own prompts with this chain. In this example, we will respond in Italian.

```
prompt_template = """Use the following pieces of context to answer the question at the end. If you don't know the answer, just say that you don't know, don't try to make up an answer.

{context}

Question: {question}
Answer in Italian:"""
PROMPT = PromptTemplate(
    template=prompt_template, input_variables=["context", "question"]
)
chain = load_qa_chain(OpenAI(temperature=0), chain_type="stuff", prompt=PROMPT)
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'output_text': ' Il presidente ha detto che Justice Breyer ha dedicato la sua vita a servire questo paese e ha ricevuto una vasta gamma di supporto.'}
```

The `map_reduce` Chain

This sections shows results of using the `map_reduce` Chain to do question answering.

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="map_reduce")
```

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

```
    {'output_text': ' The president said that Justice Breyer is an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court, and thanked him for his service.'}
```

Intermediate Steps

We can also return the intermediate steps for `map_reduce` chains, should we want to inspect them. This is done with the `return_map_steps` variable.

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="map_reduce", return_map_steps=True)
```

```
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'intermediate_steps': [' "Tonight, 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."',
      ' A former top litigator in private practice. A former federal public defender. And from a family of public school educators and police officers. A consensus builder. Since she’s been nominated, she’s received a broad range of support—from the Fraternal Order of Police to former judges appointed by Democrats and Republicans.',
      ' None',
      ' None'],
     'output_text': ' The president said that Justice Breyer is an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court, and thanked him for his service.'}
```

Custom Prompts

You can also use your own prompts with this chain. In this example, we will respond in Italian.

```
question_prompt_template = """Use the following portion of a long document to see if any of the text is relevant to answer the question.
Return any relevant text translated into italian.
{context}
Question: {question}
Relevant text, if any, in Italian:"""
QUESTION_PROMPT = PromptTemplate(
    template=question_prompt_template, input_variables=["context", "question"]
)

combine_prompt_template = """Given the following extracted parts of a long document and a question, create a
final answer italian.
If you don't know the answer, just say that you don't know. Don't try to make up an answer.

QUESTION: {question}
=========
{summaries}
=========
Answer in Italian:"""
COMBINE_PROMPT = PromptTemplate(
    template=combine_prompt_template, input_variables=["summaries", "question"]
)
chain = load_qa_chain(OpenAI(temperature=0), chain_type="map_reduce", return_map_steps=True,
question_prompt=QUESTION_PROMPT, combine_prompt=COMBINE_PROMPT)
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'intermediate_steps': ["\nStasera vorrei onorare qualcuno che ha dedicato la sua vita a servire questo
paese: il giustizia Stephen Breyer - un veterano dell'esercito, uno studioso costituzionale e un giustizia in
uscita della Corte Suprema degli Stati Uniti. Giustizia Breyer, grazie per il tuo servizio.",
      '\nNessun testo pertinente.',
      ' Non ha detto nulla riguardo a Justice Breyer.',
      " Non c'è testo pertinente."],
     'output_text': ' Non ha detto nulla riguardo a Justice Breyer.'}
```

Batch Size

When using the `map_reduce` chain, one thing to keep in mind is the batch size you are using during the map step. If this is too high, it could cause rate limiting errors. You can control this by setting the batch size on the LLM used. Note that this only applies for LLMs with this parameter. Below is an example of doing so:

```
llm = OpenAI(batch_size=5, temperature=0)
```

The `refine` Chain

This sections shows results of using the `refine` Chain to do question answering.

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="refine")
```

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

```
    {'output_text': '\n\nThe president said that he wanted to honor Justice Breyer for his dedication to serving the country, his legacy of excellence, and his commitment to advancing liberty and justice, as well as for his support of the Equality Act and his commitment to protecting the rights of LGBTQ+ Americans. He also praised Justice Breyer for his role in helping to pass the Bipartisan Infrastructure Law, which he said would be the most sweeping investment to rebuild America in history and would help the country compete for the jobs of the 21st Century.'}
```

Intermediate Steps

We can also return the intermediate steps for `refine` chains, should we want to inspect them. This is done with the `return_refine_steps` variable.

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="refine", return_refine_steps=True)
```

```
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'intermediate_steps': ['\nThe president said that he wanted to honor Justice Breyer for his dedication to serving the country and his legacy of excellence.',
      '\nThe president said that he wanted to honor Justice Breyer for his dedication to serving the country, his legacy of excellence, and his commitment to advancing liberty and justice.',
      '\n\nThe president said that he wanted to honor Justice Breyer for his dedication to serving the country, his legacy of excellence, and his commitment to advancing liberty and justice, as well as for his support of the Equality Act and his commitment to protecting the rights of LGBTQ+ Americans.',
      '\n\nThe president said that he wanted to honor Justice Breyer for his dedication to serving the country, his legacy of excellence, and his commitment to advancing liberty and justice, as well as for his support of the Equality Act and his commitment to protecting the rights of LGBTQ+ Americans. He also praised Justice Breyer for his role in helping to pass the Bipartisan Infrastructure Law, which is the most sweeping investment to rebuild America in history.'],
     'output_text': '\n\nThe president said that he wanted to honor Justice Breyer for his dedication to serving the country, his legacy of excellence, and his commitment to advancing liberty and justice, as well as for his support of the Equality Act and his commitment to protecting the rights of LGBTQ+ Americans. He also praised Justice Breyer for his role in helping to pass the Bipartisan Infrastructure Law, which is the most sweeping investment to rebuild America in history.'}
```

Custom Prompts

You can also use your own prompts with this chain. In this example, we will respond in Italian.

```
refine_prompt_template = (
    "The original question is as follows: {question}\n"
    "We have provided an existing answer: {existing_answer}\n"
    "We have the opportunity to refine the existing answer"
    "(only if needed) with some more context below.\n"
    "------------\n"
    "{context_str}\n"
    "------------\n"
    "Given the new context, refine the original answer to better "
    "answer the question. "
    "If the context isn't useful, return the original answer. Reply in Italian."
)
refine_prompt = PromptTemplate(
    input_variables=["question", "existing_answer", "context_str"],
    template=refine_prompt_template,
)


initial_qa_template = (
    "Context information is below. \n"
    "---------------------\n"
    "{context_str}"
    "\n---------------------\n"
    "Given the context information and not prior knowledge, "
    "answer the question: {question}\nYour answer should be in Italian.\n"
)
initial_qa_prompt = PromptTemplate(
    input_variables=["context_str", "question"], template=initial_qa_template
)
chain = load_qa_chain(OpenAI(temperature=0), chain_type="refine", return_refine_steps=True,
                      question_prompt=initial_qa_prompt, refine_prompt=refine_prompt)
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'intermediate_steps': ['\nIl presidente ha detto che Justice Breyer ha dedicato la sua vita al servizio di questo paese e ha reso omaggio al suo servizio.',
      "\nIl presidente ha detto che Justice Breyer ha dedicato la sua vita al servizio di questo paese, ha reso omaggio al suo servizio e ha sostenuto la nomina di una top litigatrice in pratica privata, un ex difensore pubblico federale e una famiglia di insegnanti e agenti di polizia delle scuole pubbliche. Ha anche sottolineato l'importanza di avanzare la libertà e la giustizia attraverso la sicurezza delle frontiere e la risoluzione del sistema di immigrazione.",
      "\nIl presidente ha detto che Justice Breyer ha dedicato la sua vita al servizio di questo paese, ha reso omaggio al suo servizio e ha sostenuto la nomina di una top litigatrice in pratica privata, un ex difensore pubblico federale e una famiglia di insegnanti e agenti di polizia delle scuole pubbliche. Ha anche sottolineato l'importanza di avanzare la libertà e la giustizia attraverso la sicurezza delle frontiere, la risoluzione del sistema di immigrazione, la protezione degli americani LGBTQ+ e l'approvazione dell'Equality Act. Ha inoltre sottolineato l'importanza di lavorare insieme per sconfiggere l'epidemia di oppiacei.",
      "\n\nIl presidente ha detto che Justice Breyer ha dedicato la sua vita al servizio di questo paese, ha reso omaggio al suo servizio e ha sostenuto la nomina di una top litigatrice in pratica privata, un ex difensore pubblico federale e una famiglia di insegnanti e agenti di polizia delle scuole pubbliche. Ha anche sottolineato l'importanza di avanzare la libertà e la giustizia attraverso la sicurezza delle frontiere, la risoluzione del sistema di immigrazione, la protezione degli americani LGBTQ+ e l'approvazione dell'Equality Act. Ha inoltre sottolineato l'importanza di lavorare insieme per sconfiggere l'epidemia di oppiacei e per investire in America, educare gli americani, far crescere la forza lavoro e costruire l'economia dal"],
     'output_text': "\n\nIl presidente ha detto che Justice Breyer ha dedicato la sua vita al servizio di questo paese, ha reso omaggio al suo servizio e ha sostenuto la nomina di una top litigatrice in pratica privata, un ex difensore pubblico federale e una famiglia di insegnanti e agenti di polizia delle scuole pubbliche. Ha anche sottolineato l'importanza di avanzare la libertà e la giustizia attraverso la sicurezza delle frontiere, la risoluzione del sistema di immigrazione, la protezione degli americani LGBTQ+ e l'approvazione dell'Equality Act. Ha inoltre sottolineato l'importanza di lavorare insieme per sconfiggere l'epidemia di oppiacei e per investire in America, educare gli americani, far crescere la forza lavoro e costruire l'economia dal"}
```

The `map-rerank` Chain

This sections shows results of using the `map-rerank` Chain to do question answering with sources.

```
chain = load_qa_chain(OpenAI(temperature=0), chain_type="map_rerank", return_intermediate_steps=True)
```

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

```
results["output_text"]
```

```
    ' The President thanked Justice Breyer for his service and honored him for dedicating his life to serve
the country.'
```

```
results["intermediate_steps"]
```

```
    [{'answer': ' The President thanked Justice Breyer for his service and honored him for dedicating his
life to serve the country.',
      'score': '100'},
     {'answer': ' This document does not answer the question', 'score': '0'},
     {'answer': ' This document does not answer the question', 'score': '0'},
     {'answer': ' This document does not answer the question', 'score': '0'}]
```

Custom Prompts

You can also use your own prompts with this chain. In this example, we will respond in Italian.

```
from langchain.output_parsers import RegexParser

output_parser = RegexParser(
    regex=r"(.*?)\nScore: (.*)",
    output_keys=["answer", "score"],
)

prompt_template = """Use the following pieces of context to answer the question at the end. If you don't know
the answer, just say that you don't know, don't try to make up an answer.

In addition to giving an answer, also return a score of how fully it answered the user's question. This
should be in the following format:

Question: [question here]
Helpful Answer In Italian: [answer here]
Score: [score between 0 and 100]

Begin!

Context:
---------
{context}
---------
Question: {question}
Helpful Answer In Italian:"""
PROMPT = PromptTemplate(
    template=prompt_template,
    input_variables=["context", "question"],
    output_parser=output_parser,
)

chain = load_qa_chain(OpenAI(temperature=0), chain_type="map_rerank", return_intermediate_steps=True,
prompt=PROMPT)
query = "What did the president say about Justice Breyer"
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'intermediate_steps': [{'answer': ' Il presidente ha detto che Justice Breyer ha dedicato la sua vita a
servire questo paese.',
       'score': '100'},
      {'answer': ' Il presidente non ha detto nulla sulla Giustizia Breyer.',
       'score': '100'},
      {'answer': ' Non so.', 'score': '0'},
      {'answer': ' Non so.', 'score': '0'}],
     'output_text': ' Il presidente ha detto che Justice Breyer ha dedicato la sua vita a servire questo
paese.'}
```

Document QA with sources

We can also perform document QA and return the sources that were used to answer the question. To do this we'll just need to make sure each document has a "source" key in the metadata, and we'll use the `load_qa_with_sources` helper to construct our chain:

```
docsearch = Chroma.from_texts(texts, embeddings, metadatas=[{"source": str(i)} for i in range(len(texts))])
query = "What did the president say about Justice Breyer"
docs = docsearch.similarity_search(query)
```

```
from langchain.chains.qa_with_sources import load_qa_with_sources_chain

chain = load_qa_with_sources_chain(OpenAI(temperature=0), chain_type="stuff")
query = "What did the president say about Justice Breyer"
chain({"input_documents": docs, "question": query}, return_only_outputs=True)
```

```
    {'output_text': ' The president thanked Justice Breyer for his service.\nSOURCES: 30-pl'}
```