Hypothetical Document Embeddings

Contents

This notebook goes over how to use Hypothetical Document Embeddings (HyDE), as described in this paper.

At a high level, HyDE is an embedding technique that takes queries, generates a hypothetical answer, and then embeds that generated document and uses that as the final example.

In order to use HyDE, we therefore need to provide a base embedding model, as well as an LLMChain that can be used to generate those documents. By default, the HyDE class comes with some default prompts to use (see the paper for more details on them), but we can also create our own.

```
from langchain.llms import OpenAI
from langchain.embeddings import OpenAIEmbeddings
from langchain.chains import LLMChain, HypotheticalDocumentEmbedder
from langchain.prompts import PromptTemplate
```

```
base_embeddings = OpenAIEmbeddings()
llm = OpenAI()
```

```
# Load with `web_search` prompt
embeddings = HypotheticalDocumentEmbedder.from_llm(llm, base_embeddings,
"web_search")
```

Skip to main content

```
# Now we can use it as any embedding class!
result = embeddings.embed_query("Where is the Taj Mahal?")
```

Multiple generations

We can also generate multiple documents and then combine the embeddings for those. By default, we combine those by taking the average. We can do this by changing the LLM we use to generate documents to return multiple things.

```
multi_llm = OpenAI(n=4, best_of=4)
```

```
embeddings = HypotheticalDocumentEmbedder.from_llm(multi_llm, base_embeddings,
"web_search")
```

```
result = embeddings.embed_query("Where is the Taj Mahal?")
```

Using our own prompts

Besides using preconfigured prompts, we can also easily construct our own prompts and use those in the LLMChain that is generating the documents. This can be useful if we know the domain our queries will be in, as we can condition the prompt to generate text more similar to that.

In the example below, let's condition it to generate text about a state of the union address (because we will use that in the next example).

```
prompt_template = """Please answer the user's question about the most recent state
of the union address
Question: {question}
Answer:"""
prompt = PromptTemplate(input_variables=["question"], template=prompt_template)
llm_chain = LLMChain(llm=llm, prompt=prompt)
```

```
```

Skip to main content

```
base_embeddings=base_embeddings)
```

```
result = embeddings.embed_query("What did the president say about Ketanji Brown Jackson")
```

Using HyDE

Now that we have HyDE, we can use it as we would any other embedding class! Here is using it to find similar passages in the state of the union example.

```
from langchain.text_splitter import CharacterTextSplitter
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)
```

```
docsearch = Chroma.from_texts(texts, embeddings)

query = "What did the president say about Ketanji Brown Jackson"
docs = docsearch.similarity_search(query)
```

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

```
print(docs[0].page_content)
```

```
In state after state, new laws have been passed, not only to suppress the vote, but to subvert entire elections.

We cannot let this happen.

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.

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
```

Skip to main content

```
One of the most serious constitutional responsibilities a President has is
nominating someone to serve on the United States Supreme Court.

And 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.
```