# Parent Document Retriever

When splitting documents for retrieval, there are often conflicting desires:

1. You may want to have small documents, so that their embeddings can most accurately reflect their meaning. If too long, then the embeddings can lose meaning.
2. You want to have long enough documents that the context of each chunk is retained.

The ParentDocumentRetriever strikes that balance by splitting and storing small chunks of data. During retrieval, it first fetches the small chunks but then looks up the parent ids for those chunks and returns those larger documents.

Note that "parent document" refers to the document that a small chunk originated from. This can either be the whole raw document OR a larger chunk.

```
from langchain.retrievers import ParentDocumentRetriever
```

**API Reference:**

- ParentDocumentRetriever from `langchain.retrievers`

```
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.storage import InMemoryStore
from langchain.document_loaders import TextLoader
```

**API Reference:**

- Chroma from `langchain.vectorstores`
- OpenAIEmbeddings from `langchain.embeddings`
- RecursiveCharacterTextSplitter from `langchain.text_splitter`
- InMemoryStore from `langchain.storage`
- TextLoader from `langchain.document_loaders`

```
loaders = [
    TextLoader('../../paul_graham_essay.txt'),
```

```
    TextLoader('../../state_of_the_union.txt'),
]
docs = []
for l in loaders:
    docs.extend(l.load())
```

## Retrieving Full Documents

In this mode, we want to retrieve the full documents. Therefor, we only specify a child splitter.

```
# This text splitter is used to create the child documents

child_splitter = RecursiveCharacterTextSplitter(chunk_size=400)
# The vectorstore to use to index the child chunks
vectorstore = Chroma(
    collection_name="full_documents",
    embedding_function=OpenAIEmbeddings()
)
# The storage layer for the parent documents
store = InMemoryStore()
retriever = ParentDocumentRetriever(
    vectorstore=vectorstore,
    docstore=store,
    child_splitter=child_splitter,
)
```

```
retriever.add_documents(docs)
```

This should yield two keys, because we added two documents.

```
list(store.yield_keys())
```

```
    ['05fe8d8a-bf60-4f87-b576-4351b23df266',
     '571cc9e5-9ef7-4f6c-b800-835c83a1858b']
```

Let's now call the vectorstore search functionality - we should see that it returns small chunks (since we're storing the small chunks

```
sub_docs = vectorstore.similarity_search("justice breyer")
```

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

```
    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.

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

Let's now retrieve from the overall retriever. This should return large documents - since it returns the documents where the smaller chunks are located.

```
retrieved_docs = retriever.get_relevant_documents("justice breyer")
```

```
len(retrieved_docs[0].page_content)
```

```
    38539
```

## Retrieving Larger Chunks

Sometimes, the full documents can be too big to want to retrieve them as is. In that case, what we really want to do is to first split the raw documents into larger chunks, and then split it into smaller chunks. We then index the smaller chunks, but on retrieval we retrieve the larger chunks (but still not the full documents).

```
# This text splitter is used to create the parent documents
parent_splitter = RecursiveCharacterTextSplitter(chunk_size=2000)
# This text splitter is used to create the child documents
# It should create documents smaller than the parent
child_splitter = RecursiveCharacterTextSplitter(chunk_size=400)
# The vectorstore to use to index the child chunks
vectorstore = Chroma(collection_name="split_parents",
embedding_function=OpenAIEmbeddings())
```

```
# The storage layer for the parent documents
store = InMemoryStore()
```

```
retriever = ParentDocumentRetriever(
    vectorstore=vectorstore,
    docstore=store,
    child_splitter=child_splitter,
    parent_splitter=parent_splitter,
)
```

```
retriever.add_documents(docs)
```

We can see that there are much more than two documents now - these are the larger chunks

```
len(list(store.yield_keys()))
```

```
    66
```

Let's make sure the underlying vectorstore still retrieves the small chunks.

```
sub_docs = vectorstore.similarity_search("justice breyer")
```

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

```
    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.

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

```
retrieved_docs = retriever.get_relevant_documents("justice breyer")
```

```
len(retrieved_docs[0].page_content)
```

```
1849
```

```
print(retrieved_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 retiring Justice of the United States Supreme Court. Justice Breyer, thank you for your service.

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.

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.

And if we are to advance liberty and justice, we need to secure the Border and fix the immigration system.

We can do both. At our border, we've installed new technology like cutting-edge scanners to better detect drug smuggling.

We've set up joint patrols with Mexico and Guatemala to catch more human traffickers.
```

```
    We’re putting in place dedicated immigration judges so families
fleeing persecution and violence can have their cases heard faster.

    We’re securing commitments and supporting partners in South and
Central America to host more refugees and secure their own borders.
```