Retrievers

A retriever is an interface that returns documents given an unstructured query. It is more general than a vector store. A retriever does not need to be able to store documents, only to return (or retrieve) them. Vector stores can be used as the backbone of a retriever, but there are other types of retrievers as well.

Retrievers accept a string query as input and return a list of `Document`'s as output.

Advanced Retrieval Types

LangChain provides several advanced retrieval types. A full list is below, along with the following information:

Name: Name of the retrieval algorithm.

Index Type: Which index type (if any) this relies on.

Uses an LLM: Whether this retrieval method uses an LLM.

When to Use: Our commentary on when you should considering using this retrieval method.

Description: Description of what this retrieval algorithm is doing.

Name	Index Type	Uses an LLM	When to Use
Vectorstore	Vectorstore	No	If you are just gettin started anc looking for something quick and easy.
ParentDocument	Vectorstore + Document Store	No	If your pages have lots of smaller

Name	Index Type	Uses an LLM	When to Use
			pieces of distinct informatio that are best indexed by themselve but best retrieved a together.
Multi Vector	Vectorstore + Document Store	Sometimes during indexing	If you are able to extract informatio

Name	Index Type	Uses an LLM	When to Use
			from documents that you think is more relevant to index than the text itself.
Self Query	Vectorstore	Yes	If users are asking questions that are better answered by fetching documents

Name	Index Type	Uses an LLM	When to Use
			based on metadata rather thar similarity with the text.
Contextual Compression	Any	Sometimes	If you are finding tha your retrieved document:

Name	Index Type	Uses an LLM	When to Use
			contain too much irrelevant informatio and are distracting the LLM.
Time-Weighted Vectorstore	Vectorstore	No	If you have timestamp associated with your documents and you want to retrieve th most recer ones

Name	Index Type	Uses an LLM	When to Use
Multi-Query Retriever	Any	Yes	If users are asking questions that are complex and requir multiple pieces of distinct informatio to respond

Name	Index Type	Uses an LLM	When to Use
Ensemble	Any	No	If you have multiple retrieval methods and want t try combining them.
Long-Context Reorder	Any	No	If you are working with a long context model and

Name	Index Type	Uses an LLM	When to Use
			noticing that it's no paying attention t informatio in the middle of retrieved documents

Name	Index Type	Uses an LLM	When to Use

Third Party Integrations

LangChain also integrates with many third-party retrieval services. For a full list of these, check out this list of all integrations.

Using Retrievers in LCEL

Since retrievers are `Runnable`'s, we can easily compose them with other `Runnable` objects:

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain.schema import StrOutputParser
from langchain_core.runnables import RunnablePassthrough

template = """Answer the question based only
```

```
on the following context:

{context}

Question: {question}
"""
prompt =
ChatPromptTemplate.from_template(template)
model = ChatOpenAI()


def format_docs(docs):
    return "\n\n".join([d.page_content for d
in docs])


chain = (
    {"context": retriever | format_docs,
"question": RunnablePassthrough()}
    | prompt
    | model
    | StrOutputParser()
)

chain.invoke("What did the president say
about technology?")
```

Custom Retriever

Since the retriever interface is so simple, it's pretty easy to write a custom one.

```
from langchain_core.retrievers import
BaseRetriever
from langchain_core.callbacks import
CallbackManagerForRetrieverRun
from langchain_core.documents import Document
from typing import List


class CustomRetriever(BaseRetriever):

    def _get_relevant_documents(
        self, query: str, *, run_manager:
CallbackManagerForRetrieverRun
    ) -> List[Document]:
        return [Document(page_content=query)]

retriever = CustomRetriever()

retriever.get_relevant_documents("bar")
```