Weaviate self-querying

Creating a Weaviate vectorstore

First we'll want to create a Weaviate VectorStore and seed it with some data. We've created a small demo set of documents that contain summaries of movies.

NOTE: The self-query retriever requires you to have `lark` installed (`pip install lark`). We also need the `weaviate-client` package.

```
#!pip install lark weaviate-client
```

```
from langchain.schema import Document
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import Weaviate
import os

embeddings = OpenAIEmbeddings()
```

API Reference:

- Document from `langchain.schema`
- OpenAIEmbeddings from `langchain.embeddings.openai`
- Weaviate from `langchain.vectorstores`

```
docs = [
    Document(
        page_content="A bunch of scientists bring back dinosaurs and
mayhem breaks loose",
        metadata={"year": 1993, "rating": 7.7, "genre": "science
fiction"},
    ),
    Document(
        page_content="Leo DiCaprio gets lost in a dream within a dream
within a dream within a ...",
        metadata={"year": 2010, "director": "Christopher Nolan",
```

```
"rating": 8.2},
    ),
    Document(
        page_content="A psychologist / detective gets lost in a series
of dreams within dreams within dreams and Inception reused the idea",
        metadata={"year": 2006, "director": "Satoshi Kon", "rating":
8.6},
    ),
    Document(
        page_content="A bunch of normal-sized women are supremely
wholesome and some men pine after them",
        metadata={"year": 2019, "director": "Greta Gerwig", "rating":
8.3},
    ),
    Document(
        page_content="Toys come alive and have a blast doing so",
        metadata={"year": 1995, "genre": "animated"},
    ),
    Document(
        page_content="Three men walk into the Zone, three men walk out
of the Zone",
        metadata={
            "year": 1979,
            "rating": 9.9,
            "director": "Andrei Tarkovsky",
            "genre": "science fiction",
            "rating": 9.9,
        },
    ),
]
vectorstore = Weaviate.from_documents(
    docs, embeddings, weaviate_url="http://127.0.0.1:8080"
)
```

Creating our self-querying retriever

Now we can instantiate our retriever. To do this we'll need to provide some information upfront about the metadata fields that our documents support and a short description of the document contents.

```
from langchain.llms import OpenAI
from langchain.retrievers.self_query.base import SelfQueryRetriever
from langchain.chains.query_constructor.base import AttributeInfo
```

```
metadata_field_info = [
    AttributeInfo(
        name="genre",
        description="The genre of the movie",
        type="string or list[string]",
    ),
    AttributeInfo(
        name="year",
        description="The year the movie was released",
        type="integer",
    ),
    AttributeInfo(
        name="director",
        description="The name of the movie director",
        type="string",
    ),
    AttributeInfo(
        name="rating", description="A 1-10 rating for the movie",
type="float"
    ),
]
document_content_description = "Brief summary of a movie"
llm = OpenAI(temperature=0)
retriever = SelfQueryRetriever.from_llm(
    llm, vectorstore, document_content_description,
metadata_field_info, verbose=True
)
```

API Reference:

- OpenAI from `langchain.llms`
- SelfQueryRetriever from `langchain.retrievers.self_query.base`
- AttributeInfo from `langchain.chains.query_constructor.base`

Testing it out

And now we can try actually using our retriever!

```
# This example only specifies a relevant query
retriever.get_relevant_documents("What are some movies about
dinosaurs")
```

```
    query='dinosaur' filter=None limit=None
```

```
    [Document(page_content='A bunch of scientists bring back dinosaurs
and mayhem breaks loose', metadata={'genre': 'science fiction',
'rating': 7.7, 'year': 1993}),
     Document(page_content='Toys come alive and have a blast doing so',
metadata={'genre': 'animated', 'rating': None, 'year': 1995}),
     Document(page_content='Three men walk into the Zone, three men
walk out of the Zone', metadata={'genre': 'science fiction', 'rating':
9.9, 'year': 1979}),
     Document(page_content='A psychologist / detective gets lost in a
series of dreams within dreams within dreams and Inception reused the
idea', metadata={'genre': None, 'rating': 8.6, 'year': 2006})]
```

```
# This example specifies a query and a filter
retriever.get_relevant_documents("Has Greta Gerwig directed any movies
about women")
```

```
    query='women' filter=Comparison(comparator=<Comparator.EQ: 'eq'>,
attribute='director', value='Greta Gerwig') limit=None
```

```
    [Document(page_content='A bunch of normal-sized women are supremely
wholesome and some men pine after them', metadata={'genre': None,
'rating': 8.3, 'year': 2019})]
```

Filter k

We can also use the self query retriever to specify `k`: the number of documents to fetch.

We can do this by passing `enable_limit=True` to the constructor.

```
retriever = SelfQueryRetriever.from_llm(
    llm,
```

```
    vectorstore,
    document_content_description,
    metadata_field_info,
    enable_limit=True,
    verbose=True,
)
```

```
# This example only specifies a relevant query
retriever.get_relevant_documents("what are two movies about dinosaurs")
```

```
    query='dinosaur' filter=None limit=2
```

```
    [Document(page_content='A bunch of scientists bring back dinosaurs
and mayhem breaks loose', metadata={'genre': 'science fiction',
'rating': 7.7, 'year': 1993}),
     Document(page_content='Toys come alive and have a blast doing so',
metadata={'genre': 'animated', 'rating': None, 'year': 1995})]
```