LangChain retriever

May 13, 2025

1 Develop a Retriever to Fetch Document Segments Based on Queries

Estimated time needed: **40** minutes

1.1 Overview

Imagine you are working on a project that involves processing a large collection of text documents, such as research papers, legal documents, or customer service logs. Your task is to develop a system that can quickly retrieve the most relevant segments of text based on a user's query. Traditional keyword-based search methods might not be sufficient, as they often fail to capture the nuanced meanings and contexts within the documents. To address this challenge, you can use different types of retrievers based on LangChain.

Using retrievers is crucial for several reasons:

- Efficiency: Retrievers enable fast and efficient retrieval of relevant information from large datasets, saving time and computational resources.
- Accuracy: By leveraging advanced retrieval techniques, these tools can provide more accurate and contextually relevant results compared to traditional search methods.
- Versatility: Different retrievers can be tailored to specific use cases, making them adaptable to various types of text data and query requirements.
- Context awareness: Some retrievers, like the Parent Document Retriever, can consider the broader context of the document, enhancing the relevance of the retrieved segments.

In this lab, you will learn how to use various retrievers to efficiently extract relevant document segments from text using LangChain. You will learn about four types of retrievers: `Vector Store-backed Retriever`, `Multi-Query Retriever`, `Self-Querying Retriever`, and `Parent Document Retriever`. You will also learn the differences between these retrievers and understand the appropriate situations in which to use each one. By the end of this lab, you will be equipped with the skills to implement and utilize these retrievers in your projects.

1.2 Table of Contents

Build LLM

Text splitter

Embedding model

```
        </li>
    </ol>
</li>
<li>
    <a href="#Retrievers">Retrievers</a>
    <ol>
        <li><a href="#Vector-Store-Backed-Retriever">Vector Store-Backed Retriever</a></li>
        <li><a href="#Multi-Query-Retriever">Multi-Query Retriever</a></li>
        <li><a href="#Self-Querying-Retriever">Self-Querying Retriever</a></li>
        <li><a href="#Parent-Document-Retriever">Parent Document Retriever</a></li>
    </ol>
```

Exercises

Retrieve top 2 results using vector store-backed retriever

Self querying retriever for a query

1.3 Objectives

After completing this lab, you will be able to:

- Use various types of retrievers to efficiently extract relevant document segments from text, leveraging LangChain's capabilities.
- Apply the Vector Store-backed Retriever to solve problems involving semantic similarity and relevance in large text datasets.
- Utilize the Multi-Query Retriever to address situations where multiple query variations are needed to capture comprehensive results.
- Implement the Self-Querying Retriever to automatically generate and refine queries, enhancing the accuracy of information retrieval.
- Employ the Parent Document Retriever to maintain context and relevance by considering the broader context of the parent document.

1.4 Setup

For this lab, you will use the following libraries:

- `ibm-watson-ai` for using LLMs from IBM's watsonx.ai.
- `langchain`, `langchain-ibm`, `langchain-community` for using relevant features from LangChain.
- `pypdf`is an open-source pure Python PDF library capable of splitting, merging, cropping, and transforming the pages of PDF files.
- `chromadb` is an open-source vector database used to store embeddings.
- `lark` is a general-purpose parsing library for Python. It is necessary for a Self-Querying Retriever.

1.4.1 Installing required libraries

The following required libraries are **not** preinstalled in the Skills Network Labs environment. **You must run the following cell** to install them:

Note: The version is being pinned here to specify the version. It's recommended that you do this as well. Even if the library is updated in the future, the installed library could still support this lab work.

This might take approximately 1-2 minutes.

[1]:
```
# Please restart the kernel and run all cells after executing this cell.
!pip install --user "ibm-watsonx-ai==1.1.2" | tail -n 1
!pip install --user "langchain==0.2.1" | tail -n 1
!pip install --user "langchain-ibm==0.1.11" | tail -n 1
!pip install --user "langchain-community==0.2.1" | tail -n 1
!pip install --user "chromadb==0.4.24" | tail -n 1
!pip install --user "pypdf==4.3.1" | tail -n 1
!pip install --user "lark==1.1.9" | tail -n 1
```

```
Successfully installed ibm-cos-sdk-2.13.6 ibm-cos-sdk-core-2.13.6 ibm-cos-
sdk-s3transfer-2.13.6 ibm-watsonx-ai-1.1.2 jmespath-1.0.1 lomond-0.3.3
numpy-1.26.4 pandas-2.1.4 requests-2.32.2 tabulate-0.9.0 tzdata-2025.2
Successfully installed langchain-0.2.1 langchain-core-0.2.43 langchain-text-
splitters-0.2.4 langsmith-0.1.147 orjson-3.10.18 requests-toolbelt-1.0.0
tenacity-8.5.0
Successfully installed langchain-ibm-0.1.11
Successfully installed dataclasses-json-0.6.7 langchain-community-0.2.1
marshmallow-3.26.1 mypy-extensions-1.1.0 typing-inspect-0.9.0
Successfully installed asgiref-3.8.1 backoff-2.2.1 bcrypt-4.3.0
build-1.2.2.post1 cachetools-5.5.2 chroma-hnswlib-0.7.3 chromadb-0.4.24
click-8.2.0 coloredlogs-15.0.1 deprecated-1.2.18 durationpy-0.9 fastapi-0.115.12
filelock-3.18.0 flatbuffers-25.2.10 fsspec-2025.3.2 google-auth-2.40.1
googleapis-common-protos-1.70.0 grpcio-1.71.0 httptools-0.6.4 huggingface-
hub-0.31.2 humanfriendly-10.0 kubernetes-32.0.1 markdown-it-py-3.0.0 mdurl-0.1.2
mmh3-5.1.0 mpmath-1.3.0 onnxruntime-1.22.0 opentelemetry-api-1.33.0
opentelemetry-exporter-otlp-proto-common-1.33.0 opentelemetry-exporter-otlp-
proto-grpc-1.33.0 opentelemetry-instrumentation-0.54b0 opentelemetry-
instrumentation-asgi-0.54b0 opentelemetry-instrumentation-fastapi-0.54b0
opentelemetry-proto-1.33.0 opentelemetry-sdk-1.33.0 opentelemetry-semantic-
conventions-0.54b0 opentelemetry-util-http-0.54b0 posthog-4.0.1 protobuf-5.29.4
pulsar-client-3.6.1 pyasn1-0.6.1 pyasn1-modules-0.4.2 pypika-0.48.9
pyproject_hooks-1.2.0 python-dotenv-1.1.0 requests-oauthlib-2.0.0 rich-14.0.0
rsa-4.9.1 shellingham-1.5.4 starlette-0.46.2 sympy-1.14.0 tokenizers-0.21.1
typer-0.15.3 uvicorn-0.34.2 uvloop-0.21.0 watchfiles-1.0.5 websockets-15.0.1
wrapt-1.17.2
Successfully installed pypdf-4.3.1
Successfully installed lark-1.1.9
```

After you install the libraries, restart your kernel. You can do that by clicking the **Restart the**

kernel icon.

1.4.2 Defining helper functions

Use the following code to define some helper functions to reduce the repeat work in the notebook:

[1]:
```
# You can use this section to suppress warnings generated by your code:
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
warnings.filterwarnings('ignore')
```

The following functions are prerequisite knowledge for understanding the topic of this project—retrievers. These functions include:

- Building LLMs
- Splitting documents into chunks
- Building an embedding model

The relevant knowledge and details of these functions have been covered in previous lessons.

Build LLM

[2]:
```
from ibm_watsonx_ai.foundation_models import ModelInference
from ibm_watsonx_ai.metanames import GenTextParamsMetaNames as GenParams
from ibm_watsonx_ai import Credentials
from ibm_watsonx_ai.foundation_models.extensions.langchain import WatsonxLLM
```

`mixtral-8x7b-instruct-v01` is used as the base foundational LLM.

[3]:
```
def llm():
    model_id = 'mistralai/mixtral-8x7b-instruct-v01'

    parameters = {
        GenParams.MAX_NEW_TOKENS: 256,  # this controls the maximum number of␣
 ↪tokens in the generated output
        GenParams.TEMPERATURE: 0.5, # this randomness or creativity of the␣
 ↪model's responses
    }

    credentials = {
        "url": "https://us-south.ml.cloud.ibm.com"
    }


    project_id = "skills-network"

    model = ModelInference(
        model_id=model_id,
        params=parameters,
```

```
        credentials=credentials,
        project_id=project_id
    )

    mixtral_llm = WatsonxLLM(model = model)
    return mixtral_llm
```

Text splitter

[4]:
```
from langchain.text_splitter import RecursiveCharacterTextSplitter
```

[5]:
```
def text_splitter(data, chunk_size, chunk_overlap):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size,
        chunk_overlap=chunk_overlap,
        length_function=len,
    )
    chunks = text_splitter.split_documents(data)
    return chunks
```

Embedding model The following code demonstrates how to build an embedding model using the `watsonx.ai` package.

For this project, the `ibm/slate-125m-english-rtrvr` embedding model is used.

[6]:
```
from ibm_watsonx_ai.metanames import EmbedTextParamsMetaNames
from langchain_ibm import WatsonxEmbeddings
```

[7]:
```
def watsonx_embedding():
    embed_params = {
        EmbedTextParamsMetaNames.TRUNCATE_INPUT_TOKENS: 3,
        EmbedTextParamsMetaNames.RETURN_OPTIONS: {"input_text": True},
    }

    watsonx_embedding = WatsonxEmbeddings(
        model_id="ibm/slate-125m-english-rtrvr",
        url="https://us-south.ml.cloud.ibm.com",
        project_id="skills-network",
        params=embed_params,
    )
    return watsonx_embedding
```

1.5 Retrievers

A retriever is an interface designed to return documents based on an unstructured query. Unlike a vector store, which stores and retrieves documents, a retriever's primary function is to find and return relevant documents. While vector stores can serve as the backbone of a retriever, there are various other types of retrievers that can be used as well.

Retrievers take a string `query` as input and output a list of `Documents`.

1.5.1 Vector Store-Backed Retriever

A vector store retriever is a type of retriever that utilizes a vector store to fetch documents. It acts as a lightweight wrapper around the vector store class, enabling it to conform to the retriever interface. This retriever leverages the search methods implemented by the vector store, such as similarity search and Maximum Marginal Relevance (MMR), to query texts stored within it.

Before demonstrating this retriever, you need to load some example text. A `.txt` document has been prepared for you.

[8]:
```
!wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪MZ9z1lm-Ui3YBp3SYWLTAQ/companypolicies.txt"
```

```
--2025-05-13 15:37:19--  https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/MZ9z1lm-Ui3YBp3SYWLTAQ/companypolicies.txt
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-
courses-data.s3.us.cloud-object-storage.appdomain.cloud)… 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-
courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|:443…
connected.
200 OKequest sent, awaiting response…
Length: 15660 (15K) [text/plain]
Saving to: ‘companypolicies.txt.2’

companypolicies.txt 100%[===================>]  15.29K  --.-KB/s    in 0s

2025-05-13 15:37:19 (67.5 MB/s) - ‘companypolicies.txt.2’ saved [15660/15660]
```

Use `TextLoader` to load the document.

[9]:
```
from langchain_community.document_loaders import TextLoader
```

[10]:
```
loader = TextLoader("companypolicies.txt")
txt_data = loader.load()
```

Let's take a look at this document. This is a document about different policies in a company.

[11]:
```
txt_data
```

[11]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content="1.\tCode of
Conduct\n\nOur Code of Conduct outlines the fundamental principles and ethical
standards that guide every member of our organization. We are committed to
maintaining a workplace that is built on integrity, respect, and
accountability.\nIntegrity: We hold ourselves to the highest ethical standards.
This means acting honestly and transparently in all our interactions, whether
with colleagues, clients, or the broader community. We respect and protect
sensitive information, and we avoid conflicts of interest.\nRespect: We embrace
```

diversity and value each individual's contributions. Discrimination, harassment, or any form of disrespectful behavior is unacceptable. We create an inclusive environment where differences are celebrated and everyone is treated with dignity and courtesy.\nAccountability: We take responsibility for our actions and decisions. We follow all relevant laws and regulations, and we strive to continuously improve our practices. We report any potential violations of this code and support the investigation of such matters.\nSafety: We prioritize the safety of our employees, clients, and the communities we serve. We maintain a culture of safety, including reporting any unsafe conditions or practices.\nEnvironmental Responsibility: We are committed to minimizing our environmental footprint and promoting sustainable practices.\nOur Code of Conduct is not just a set of rules; it is the foundation of our organization's culture. We expect all employees to uphold these principles and serve as role models for others, ensuring we maintain our reputation for ethical conduct, integrity, and social responsibility.\n\n2.\tRecruitment Policy\n\nOur Recruitment Policy reflects our commitment to attracting, selecting, and onboarding the most qualified and diverse candidates to join our organization. We believe that the success of our company relies on the talents, skills, and dedication of our employees.\nEqual Opportunity: We are an equal opportunity employer and do not discriminate on the basis of race, color, religion, sex, sexual orientation, gender identity, national origin, age, disability, or any other protected status. We actively promote diversity and inclusion.\nTransparency: We maintain transparency in our recruitment processes. All job vacancies are advertised internally and externally when appropriate. Job descriptions and requirements are clear and accurately represent the role.\nSelection Criteria: Our selection process is based on the qualifications, experience, and skills necessary for the position. Interviews and assessments are conducted objectively, and decisions are made without bias.\nData Privacy: We are committed to protecting the privacy of candidates' personal information and adhere to all relevant data protection laws and regulations.\nFeedback: Candidates will receive timely and constructive feedback on their application and interview performance.\nOnboarding: New employees receive comprehensive onboarding to help them integrate into the organization effectively. This includes information on our culture, policies, and expectations.\nEmployee Referrals: We encourage and appreciate employee referrals as they contribute to building a strong and engaged team.\nOur Recruitment Policy is a foundation for creating a diverse, inclusive, and talented workforce. It ensures that we attract and hire the best candidates who align with our company values and contribute to our continued success. We continuously review and update this policy to reflect evolving best practices in recruitment.\n\n3.\tInternet and Email Policy\n\nOur Internet and Email Policy is established to guide the responsible and secure use of these essential tools within our organization. We recognize their significance in daily business operations and the importance of adhering to principles that maintain security, productivity, and legal compliance.\nAcceptable Use: Company-provided internet and email services are primarily meant for job-related tasks. Limited personal use is allowed during non-work hours, provided it doesn't interfere with work

responsibilities.\nSecurity: Safeguard your login credentials, avoiding the sharing of passwords. Exercise caution with email attachments and links from unknown sources. Promptly report any unusual online activity or potential security breaches.\nConfidentiality: Reserve email for the transmission of confidential information, trade secrets, and sensitive customer data only when encryption is applied. Exercise discretion when discussing company matters on public forums or social media.\nHarassment and Inappropriate Content: Internet and email usage must not involve harassment, discrimination, or the distribution of offensive or inappropriate content. Show respect and sensitivity to others in all online communications.\nCompliance: Ensure compliance with all relevant laws and regulations regarding internet and email usage, including those related to copyright and data protection.\nMonitoring: The company retains the right to monitor internet and email usage for security and compliance purposes.\nConsequences: Policy violations may lead to disciplinary measures, including potential termination.\nOur Internet and Email Policy aims to promote safe, responsible usage of digital communication tools that align with our values and legal obligations. Each employee is expected to understand and follow this policy. Regular reviews ensure its alignment with evolving technology and security standards.\n\n4.\tMobile Phone Policy\n\nThe Mobile Phone Policy sets forth the standards and expectations governing the appropriate and responsible usage of mobile devices in the organization. The purpose of this policy is to ensure that employees utilize mobile phones in a manner consistent with company values and legal compliance.\nAcceptable Use: Mobile devices are primarily intended for work-related tasks. Limited personal usage is allowed, provided it does not disrupt work obligations.\nSecurity: Safeguard your mobile device and access credentials. Exercise caution when downloading apps or clicking links from unfamiliar sources. Promptly report security concerns or suspicious activities related to your mobile device.\nConfidentiality: Avoid transmitting sensitive company information via unsecured messaging apps or emails. Be discreet when discussing company matters in public spaces.\nCost Management: Keep personal phone usage separate from company accounts and reimburse the company for any personal charges on company-issued phones.\nCompliance: Adhere to all pertinent laws and regulations concerning mobile phone usage, including those related to data protection and privacy.\nLost or Stolen Devices: Immediately report any lost or stolen mobile devices to the IT department or your supervisor.\nConsequences: Non-compliance with this policy may lead to disciplinary actions, including the potential loss of mobile phone privileges.\nThe Mobile Phone Policy is aimed at promoting the responsible and secure use of mobile devices in line with legal and ethical standards. Every employee is expected to comprehend and abide by these guidelines. Regular reviews of the policy ensure its ongoing alignment with evolving technology and security best practices.\n\n5.\tSmoking Policy\n\nPolicy Purpose: The Smoking Policy has been established to provide clear guidance and expectations concerning smoking on company premises. This policy is in place to ensure a safe and healthy environment for all employees, visitors, and the general public.\nDesignated Smoking Areas: Smoking is only permitted in designated smoking areas, as marked by appropriate signage. These areas have been chosen to

minimize exposure to secondhand smoke and to maintain the overall cleanliness of the premises.\nSmoking Restrictions: Smoking inside company buildings, offices, meeting rooms, and other enclosed spaces is strictly prohibited. This includes electronic cigarettes and vaping devices.\nCompliance with Applicable Laws: All employees and visitors must adhere to relevant federal, state, and local smoking laws and regulations.\nDisposal of Smoking Materials: Properly dispose of cigarette butts and related materials in designated receptacles. Littering on company premises is prohibited.\nNo Smoking in Company Vehicles: Smoking is not permitted in company vehicles, whether they are owned or leased, to maintain the condition and cleanliness of these vehicles.\nEnforcement and Consequences: All employees and visitors are expected to adhere to this policy. Non-compliance may lead to appropriate disciplinary action, which could include fines, or, in the case of employees, possible termination of employment.\nReview of Policy: This policy will be reviewed periodically to ensure its alignment with evolving legal requirements and best practices for maintaining a healthy and safe workplace.\nWe appreciate your cooperation in maintaining a smoke-free and safe environment for all.\n\n6.\tDrug and Alcohol Policy\n\nPolicy Objective: The Drug and Alcohol Policy is established to establish clear expectations and guidelines for the responsible use of drugs and alcohol within the organization. This policy aims to maintain a safe, healthy, and productive workplace.\nProhibited Substances: The use, possession, distribution, or sale of illegal drugs or unauthorized controlled substances is strictly prohibited on company premises or during work-related activities. This includes the misuse of prescription drugs.\nAlcohol Consumption: The consumption of alcoholic beverages is not allowed during work hours, on company property, or while performing company-related duties. Exception may be made for company-sanctioned events.\nImpairment: Employees are expected to perform their job duties without impairment from drugs or alcohol. The use of substances that could impair job performance or pose a safety risk is prohibited.\nTesting and Searches: The organization reserves the right to conduct drug and alcohol testing as per applicable laws and regulations. Employees may be subject to testing in cases of reasonable suspicion, post-accident, or as part of routine workplace safety measures.\nReporting: Employees should report any concerns related to drug or alcohol misuse by themselves or their colleagues, as well as safety concerns arising from such misuse.\nTreatment and Assistance: Employees with substance abuse issues are encouraged to seek help. The organization is committed to providing support, resources, and information to assist those seeking treatment.\nConsequences: Violation of this policy may result in disciplinary actions, up to and including termination of employment. Legal action may also be pursued when necessary.\nPolicy Review: This policy will undergo periodic review to ensure its continued relevance and compliance with evolving legal requirements and best practices for a safe and productive work environment.\nYour adherence to this policy is appreciated as it helps to maintain a safe and drug-free workplace for all.\n\n7.\tHealth and Safety Policy\n\nOur commitment to health and safety is paramount. We prioritize the well-being of our employees, customers, and the public. We diligently comply with all relevant health and safety laws and regulations. Our objective is to

maintain a workplace free from hazards, preventing accidents, injuries, and illnesses. Every individual within our organization is responsible for upholding these standards. We regularly assess and improve our safety measures, provide adequate training, and encourage open communication regarding safety concerns. Through collective dedication, we aim to ensure a safe, healthy, and secure environment for all. Your cooperation is essential in achieving this common goal.\n\n8.\tAnti-discrimination and Harassment Policy\n\nThe Anti-Discrimination and Harassment Policy is a testament to the commitment of this organization in fostering a workplace that is free from discrimination, harassment, and any form of unlawful bias. This policy applies to every individual within the organization, including employees, contractors, visitors, and clients.\nNon-Discrimination: This organization strictly prohibits discrimination based on race, color, religion, gender, national origin, age, disability, sexual orientation, or any other legally protected characteristic in all aspects of employment, including recruitment, hiring, compensation, benefits, promotions, and terminations.\nHarassment: Harassment in any form, whether based on the aforementioned characteristics or any other protected status, is unacceptable. This encompasses unwelcome advances, offensive jokes, slurs, and other verbal or physical conduct that creates a hostile or intimidating work environment.\nReporting: Individuals who experience or witness any form of discrimination or harassment are encouraged to promptly report the incident to their supervisor, manager, or the designated HR representative. The organization is committed to a timely and confidential investigation of such complaints.\nConsequences: Violation of this policy may result in disciplinary action, including termination of employment. The organization is committed to taking appropriate action against any individual found to be in violation of this policy.\nReview and Update: This policy is subject to regular review and update to remain aligned with evolving legal requirements and best practices in preventing discrimination and harassment. This organization considers it a collective responsibility to ensure a workplace free from discrimination and harassment, and it is essential that every individual within the organization plays their part in upholding these principles.\n\n9.\tDiscipline and Termination Policy\n\nThe Discipline and Termination Policy underscores the organization's commitment to maintaining a productive, ethical, and respectful work environment. This policy applies to all personnel, including employees, contractors, and temporary staff.\nPerformance and Conduct Expectations: Employees are expected to meet performance standards and adhere to conduct guidelines. The organization will provide clear expectations, feedback, and opportunities for improvement when performance or conduct issues arise.\nDisciplinary Actions: When necessary, disciplinary actions will be taken, which may include verbal warnings, written warnings, suspension, or other appropriate measures. Disciplinary actions are designed to address issues constructively and maintain performance standards.\nTermination: In situations where an employee's performance or conduct issues persist, the organization may resort to termination. Termination may also occur for reasons such as redundancy, violation of policies, or restructuring.\nTermination Procedure: The organization will follow appropriate procedures, ensuring fairness and adherence

```
to legal requirements during the termination process. Employees may be eligible
for notice periods, severance pay, or other benefits as per employment
agreements and applicable laws.\nExit Process: The organization will conduct an
exit process to ensure a smooth transition for departing employees, including
the return of company property, final pay, and cancellation of access and
benefits.\nThis policy serves as a framework for handling discipline and
termination. The organization recognizes the importance of fairness and
consistency in these processes, and decisions will be made after careful
consideration. Every employee is expected to understand and adhere to this
policy, contributing to a respectful and productive workplace. Regular reviews
will ensure its alignment with evolving legal requirements and best
practices.\n")]
```

Split `txt_data` into chunks. `chunk_size = 200`, `chunk_overlap = 20` has been set.

[12]:
```
chunks_txt = text_splitter(txt_data, 200, 20)
```

Store the embeddings into a `ChromaDB`.

[13]:
```
from langchain.vectorstores import Chroma
```

[14]:
```
vectordb = Chroma.from_documents(chunks_txt, watsonx_embedding())
```

Simple similarity search Here is an example of a simple similarity search based on the vector database.

For this demonstration, the query has been set to "email policy".

[15]:
```
query = "email policy"
retriever = vectordb.as_retriever()
```

[16]:
```
docs = retriever.invoke(query)
```

By default, the number of retrieval results is four, and they are ranked by similarity level.

[17]:
```
docs
```

[17]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content='This policy
serves as a framework for handling discipline and termination. The organization
recognizes the importance of fairness and consistency in these processes, and
decisions will be made after'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='This policy
aims to maintain a safe, healthy, and productive workplace.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='will ensure
its alignment with evolving legal requirements and best practices.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='Review of
Policy: This policy will be reviewed periodically to ensure its alignment with
evolving legal requirements and best practices for maintaining a healthy and
safe workplace.')]
```

You can also specify `search kwargs` like `k` to limit the retrieval results.

[18]:
```
retriever = vectordb.as_retriever(search_kwargs={"k": 1})
docs = retriever.invoke(query)
docs
```

[18]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content='This policy
serves as a framework for handling discipline and termination. The organization
recognizes the importance of fairness and consistency in these processes, and
decisions will be made after')]
```

MMR retrieval MMR in vector stores is a technique used to balance the relevance and diversity of retrieved results. It selects documents that are both highly relevant to the query and minimally similar to previously selected documents. This approach helps to avoid redundancy and ensures a more comprehensive coverage of different aspects of the query.

The following code is showing how to conduct an MMR search in a vector database. You just need to sepecify `search_type="mmr"`.

[19]:
```
retriever = vectordb.as_retriever(search_type="mmr")
docs = retriever.invoke(query)
docs
```

[19]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content='This policy
serves as a framework for handling discipline and termination. The organization
recognizes the importance of fairness and consistency in these processes, and
decisions will be made after'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='Employee
Referrals: We encourage and appreciate employee referrals as they contribute to
building a strong and engaged team.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content="Data
Privacy: We are committed to protecting the privacy of candidates' personal
information and adhere to all relevant data protection laws and regulations."),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='harassment.
This organization considers it a collective responsibility to ensure a workplace
free from discrimination and harassment, and it is essential that every
individual within the organization')]
```

Similarity score threshold retrieval You can also set a retrieval method that defines a similarity score threshold, returning only documents with a score above that threshold.

[20]:
```
retriever = vectordb.as_retriever(
    search_type="similarity_score_threshold", search_kwargs={"score_threshold":␣
 ↪0.4}
)
docs = retriever.invoke(query)
docs
```

[20]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content='This policy
serves as a framework for handling discipline and termination. The organization
recognizes the importance of fairness and consistency in these processes, and
decisions will be made after'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='This policy
aims to maintain a safe, healthy, and productive workplace.')]
```

1.5.2 Multi-Query Retriever

Distance-based vector database retrieval represents queries in high-dimensional space and finds similar embedded documents based on "distance". However, retrieval results may vary with subtle changes in query wording or if the embeddings do not accurately capture the data's semantics.

The `MultiQueryRetriever` addresses this by using an LLM to generate multiple queries from different perspectives for a given user input query. For each query, it retrieves a set of relevant documents and then takes the unique union of these results to form a larger set of potentially relevant documents. By generating multiple perspectives on the same question, the `MultiQueryRetriever` can potentially overcome some limitations of distance-based retrieval, resulting in a richer and more diverse set of results.

The following picture shows the difference between retrievers solely based on distance and the Multi-Query Retriever.

Let's consider the query sentence, `"I like cats"`.

On the upper side of the picture, you can see a retriever that relies solely on distance. This retriever calculates the distance between the query and the documents in the vector store, returning the document with the closest match.

On the lower side, you can see a multi-query retriever. It first uses an LLM to generate multiple queries from different perspectives based on the user's input query. For each generated query, it retrieves relevant documents and then returns the union of these results.

A PDF document has been prepared to demonstrate this Multi-Query Retriever.

[21]:
```
from langchain_community.document_loaders import PyPDFLoader
```

[22]:
```
loader = PyPDFLoader("https://cf-courses-data.s3.us.cloud-object-storage.
↪appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf")
pdf_data = loader.load()
```

Let's take a look at the first page of this paper. This paper is talking about the framework LangChain.

[23]:
```
pdf_data[1]
```

[23]:
```
Document(metadata={'source': 'https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf', 'page': 1},
page_content='LangChain helps us to unlock the ability to harness the \nLLM's
immense potential in tasks such as document analysis, \nchatbot development,
code analysis, and countless other \napplications. Whether your desire is to
```

unlock deeper natural \nlanguage understanding , enhance data, or circumvent \nlanguage barriers through translation, LangChain is ready to \nprovide the tools and programming support you need to do \nwithout it that it is not only difficult but also fresh for you . Its \ncore functionalities encompass: \n1. Context -Aware Capabilities: LangChain facilitates the \ndevelopment of applications that are inherently \ncontext -aware. This means that these applications can \nconnect to a language model and draw from various \nsources of context, such as prompt instructions, a few-\nshot examples, or existing content, to ground their \nresponses effectively. \n2. Reasoning Abilities: LangChain equips applications \nwith the capacity to reason effectively. By relying on a \nlanguage model, these applications can make informed \ndecisions about how to respond based on the provided \ncontext and determine the appropriate acti ons to take. \nLangChain offers several key value propositions: \nModular Components: It provides abstractions that \nsimplify working with language models, along with a \ncomprehensive collection of implementations for each \nabstraction. These components are designed to be modular \nand user -friendly, making them useful whethe r you are \nutilizing the entire LangChain framework or not. \nOff-the-Shelf Chains: LangChain offers pre -configured \nchains, which are structured assemblies of components \ntailored to accomplish specific high -level tasks. These pre -\ndefined chains streamline the initial setup process and serve as \nan ideal starting point for your projects. The MindGuide Bot \nuses below components from LangChain . \nA. ChatModel \nWithin LangChain, a ChatModel is a specific kind of \nlanguage model crafted to manage conversational \ninteractions. Unlike traditional language models that take one \nstring as input and generate a single string as output, \nChatModels operate with a list of mes sages as input, \ngenerating a message as output. \nEach message in the list has two parts: the content and the \nrole. The content is the actual text or substance of the message, \nwhile the role denotes the role or source of the message (such \nas "User," "Assistant," "System," etc.). \nThis approach with ChatModels opens the door to more \ndynamic and interactive conversations with the language \nmodel. It empowers the creation of chatbot applications, \ncustomer support systems, or any other application involving \nmulti -turn conversations. We utilized the ChatOpenAI \nChatModel to create MindGuide chatbots specifically \ndesigned to function as mental health therapists. In our \ninteraction with OpenAI, we opted for an OpenAI API key to \nengage with the ChatGpt3 turbo model and utilized a \ntemperature value of 0.5. The steps to create an OpenAI API \nkey are outlined [9]. B. Message \nIn the context of LangChain, messages [10] refer to a list of \nmessages that are used as input when interacting with a \nChatModel. Each message in the list represents a specific turn \nor exchange in a conversation. Each message in the messages \nlist typically consists of two components: \n• content: This represents the actual text or content of \nthe message. It can be a user query, a system \ninstruction, or any other relevant information. \n• role: This represents the role or source of the \nmessage. It defines who is speaking or generating \nthe message. Common roles include "User", \n"Assistant", "System", or any other custom role you \ndefine . \nThe chat model interface is based around messages rather \nthan raw text. The types of

```
messages supported in LangChain \nare Systen Message, HumanMessage, and
AIMessage . \nSystemMessage  is the ChatMessage coming from the system  \nin its
LangChain template  as illustrated in Figure 1. Human \nMessage  is a
ChatMessage coming from a human/user.  \nAIMessage is a ChatMessage  coming from
an AI/assistant as \nillustrated in Figure 2 .  \n \n                    Figure
1. A System Message illustration   You are a compassionate and experienced
mental \nhealth therapist with a proven track record of \nhelping patients
overcome anxiety and other mental \nhealth challenges. Your primary objective is
to \nsupport the patient in addressing their concerns \nand guiding th em
towards positive change. In this \ninteractive therapy session, you will engage
with \nthe patient by asking open -ended questions, \nactively listening to
their responses, and providing \nempathetic feedback. Your approach is
\ncollaborative, and you strive to cr eate a safe and \nnon-judgmental space for
the patient to share their \nthoughts and feelings.  \nAs the patient shares
their struggles, you will \nprovide insightful guidance and evidence -based
\nstrategies tailored to their unique needs. You may \nalso offer practical
exercises or resources to help \nthem manage their symptoms and improve their
\nmental wellbe ing. When necessary, you will gently \nredirect the conversation
back to the patient\'s \nprimary concerns related to anxiety, mental health,
\nor family issues. This ensures that each session is \nproductive and focused
on addressing the most \npressing issues. Thro ughout the session, you \nremain
mindful of the patient\'s emotional state and \nadjust your approach
accordingly.  \nYou recognize that everyone\'s journey is \ndifferent, and that
progress can be incremental.  \nBy building trust and fostering a strong
\ntherapeutic relationship, you empower the patient \nto take ownership of their
growth and development. \nAt the end of the session, you will summarize key
\npoints from your discussion, highlighting the \npatient\'s strength s and
areas for improvement.  \nTogether, you will set achievable goals for future
\nsessions, reinforcing a sense of hope and \nmotivation. Your ultimate goal is
to equip the \npatient with the tools and skills needed to navigate \nlife\'s
challenges with confidence and resilience . ')
```

Split document and store the embeddings into a vector database.

[24]:
```
# Split
chunks_pdf = text_splitter(pdf_data, 500, 20)

# VectorDB
ids = vectordb.get()["ids"]
vectordb.delete(ids) # We need to delete existing embeddings from previous␣
 ↪documents and then store current document embeddings in.
vectordb = Chroma.from_documents(documents=chunks_pdf,␣
 ↪embedding=watsonx_embedding())
```

The `MultiQueryRetriever` function from LangChain is used.

[31]:
```
from langchain.retrievers.multi_query import MultiQueryRetriever
```

```
query = "What does the paper say about langchain?"

retriever = MultiQueryRetriever.from_llm(
    retriever=vectordb.as_retriever(), llm=llm()
)
```

Set logging for the queries.

[32]:
```
import logging

logging.basicConfig()
logging.getLogger("langchain.retrievers.multi_query").setLevel(logging.INFO)
```

[33]:
```
docs = retriever.invoke(query)
docs
```

```
INFO:langchain.retrievers.multi_query:Generated queries: ['1. Can you provide a
summary of the information discussed about langchain in the document?', '2. What
is the stance of the paper towards langchain?', '3. In the context of the paper,
what is the role or significance of langchain?']
```

[33]:
```
[Document(metadata={'page': 3, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='Step 8.  User Response Delivery:  Present the model -\ngenerated
response to the user, thereby delivering the \nmental health advice or
information they sought .\n \nFigure 3. MindGuide Chatbot Architecture'),
 Document(metadata={'page': 0, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='II. LANGCHAIN  \nLangChain , with its open -source essence,
emerges as a \npromising solution, aiming to simplify the complex process of
\ndeveloping applications powered by large language models \n(LLMs) . This
framework though the rapid delivery of building \nblocks and pre -built chains
for building large language model \napplications shows the easy way developers
can do it .'),
 Document(metadata={'page': 2, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='simple task. What is less obvious are the data \nstructures and
algorithms built on top of chat \nconversations to provide the most usable view
of \nthose chats . \nA simple memory system may only return the most \nrecent
messages on each iteration. A slightly more \ncomplicated memory system may
return a brief summary of \nthe last K messages. A more complex system might
extract \nentities from stored messages and only retur n information \nabout
entities that have been referenced in the current run.'),
 Document(metadata={'page': 5, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='[3] S. Ji, C. P. Yu, S. F. Fung, S. Pan, and G. Long, "Supervised
learning \nfor suicidal ideation detection in online user content," Complex,
2018.  \n[4] LangChain, https://www.langchain.com/  (accessed Nov. 29, 202 3).
```

```
\n[5] LangChain ChatModels, https://blog.langchain.dev/chat -models/ \n(accessed
Nov. 29, 2023).  \n[6] LangChain with OpenAI Chat Model ,
\nhttps://python.langchain.com/docs/integrations/chat/openai / (accessed'),
 Document(metadata={'page': 2, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='Figure 2. An AIMessage illustration  \nC. Prompt Template
\nPrompt templates  [10] allow you to structure  input for LLMs. \nThey provide
a convenient way to format user inputs and \nprovide instructions to generate
responses. Prompt templates \nhelp ensure that the LLM understands the  desired
context and \nproduces relevant outputs.  \nThe prompt template classes in
LangChain  are built to \nmake constructing prompts with dynamic inputs easier.
Of'),
 Document(metadata={'page': 1, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='• content: This represents the actual text or content of \nthe
message. It can be a user query, a system \ninstruction, or any other relevant
information.  \n• role: This represents the role or source of the \nmessage. It
defines who is speaking or generating \nthe message. Common roles include
"User", \n"Assistant", "System", or any other custom role you \ndefine . \nThe
chat model interface is based around messages rather \nthan raw text. The types
of messages supported in LangChain'),
 Document(metadata={'page': 0, 'source': 'https://cf-courses-data.s3.us.cloud-
object-storage.appdomain.cloud/ioch1wsxkfqgfLLgmd-6Rw/langchain-paper.pdf'},
page_content='Additionally, the paper discusses the implementation of
\nStreamlit to enhance the user ex perience and interaction with \nthe chatbot.
Th is novel approach holds great promise for \nproactive mental health
intervention and assistance . \nKeywords  -Large Language models , LangChain,
Chatbot , \nPretrained models , Mental health , Mental health support . \nI.
INTRODUCTION  \nThe issue of mental health is an international situation,
\naffecting people in each particularly developed nations and')]
```

From the log results, you can see that the LLM generated three additional queries from different perspectives based on the given query.

The returned results are the union of the results from each query.

1.5.3 Self-Querying Retriever

A Self-Querying Retriever, as the name suggests, has the ability to query itself. Specifically, given a natural language query, the retriever uses a query-constructing LLM chain to generate a structured query. It then applies this structured query to its underlying vector store. This enables the retriever to not only use the user-input query for semantic similarity comparison with the contents of stored documents but also to extract and apply filters based on the metadata of those documents.

The following code demonstrates how to use a Self-Querying Retriever.

[34]:
```
from langchain_core.documents import Document
from langchain.chains.query_constructor.base import AttributeInfo
from langchain.retrievers.self_query.base import SelfQueryRetriever
```

```
from lark import lark
```

A couple of document pieces have been prepared where the `page_content` contains descriptions of movies, and the `meta_data` includes different attributes for each movie, such as `year`, `rating`, `genre`, and `director`. These attributes are crucial in the Self-Querying Retriever, as the LLM will use the metadata information to apply filters during the retrieval process.

[35]:
```
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,
            "director": "Andrei Tarkovsky",
            "genre": "thriller",
            "rating": 9.9,
        },
    ),
]
```

Now you can instantiate your retriever. To do this, you'll need to provide some upfront information about the metadata fields that your documents support, as well as a brief description of the document contents.

[36]:
```
metadata_field_info = [
    AttributeInfo(
        name="genre",
        description="The genre of the movie. One of ['science fiction',␣
 ↪'comedy', 'drama', 'thriller', 'romance', 'action', 'animated']",
        type="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"
    ),
]
```

Store the document's embeddings into a vector database.

[37]:
```
vectordb = Chroma.from_documents(docs, watsonx_embedding())
```

Use the `SelfQueryRetriever`.

[38]:
```
document_content_description = "Brief summary of a movie."

retriever = SelfQueryRetriever.from_llm(
    llm(),
    vectordb,
    document_content_description,
    metadata_field_info,
)
```

Now you can actually try using your retriever.

[39]:
```
# This example only specifies a filter
retriever.invoke("I want to watch a movie rated higher than 8.5")
```

[39]:
```
[Document(metadata={'director': 'Andrei Tarkovsky', 'genre': 'thriller',
'rating': 9.9, 'year': 1979}, page_content='Three men walk into the Zone, three
men walk out of the Zone'),
 Document(metadata={'director': 'Satoshi Kon', 'rating': 8.6, 'year': 2006},
page_content='A psychologist / detective gets lost in a series of dreams within
dreams within dreams and Inception reused the idea')]
```

[40]:
```
# This example specifies a query and a filter
retriever.invoke("Has Greta Gerwig directed any movies about women")
```

[40]:
```
[Document(metadata={'director': 'Greta Gerwig', 'rating': 8.3, 'year': 2019},
page_content='A bunch of normal-sized women are supremely wholesome and some men
pine after them')]
```

When running the following cell, you might encounter some errors or blank content. This is because the LLM cannot get the answer at first. Don't worry; if you re-run it several times, you will get the answer.

[41]:
```
# This example specifies a composite filter
retriever.invoke("What's a highly rated (above 8.5) science fiction film?")
```

[41]:
```
[Document(metadata={'director': 'Satoshi Kon', 'rating': 8.6, 'year': 2006},
page_content='A psychologist / detective gets lost in a series of dreams within
dreams within dreams and Inception reused the idea'),
 Document(metadata={'director': 'Andrei Tarkovsky', 'genre': 'thriller',
'rating': 9.9, 'year': 1979}, page_content='Three men walk into the Zone, three
men walk out of the Zone')]
```

1.5.4 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 the documents are too long, the embeddings can lose meaning.
2. You want to have long enough documents so 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.

[42]:
```
from langchain.retrievers import ParentDocumentRetriever
from langchain_text_splitters import CharacterTextSplitter
from langchain.storage import InMemoryStore
```

[43]:
```
# Set two splitters. One is with big chunk size (parent) and one is with small␣
 ↪chunk size (child)
parent_splitter = CharacterTextSplitter(chunk_size=2000, chunk_overlap=20,␣
 ↪separator='\n')
child_splitter = CharacterTextSplitter(chunk_size=400, chunk_overlap=20,␣
 ↪separator='\n')
```

[44]:
```
vectordb = Chroma(
    collection_name="split_parents", embedding_function=watsonx_embedding()
)

# The storage layer for the parent documents
```

```
store = InMemoryStore()
```

[45]:
```
retriever = ParentDocumentRetriever(
    vectorstore=vectordb,
    docstore=store,
    child_splitter=child_splitter,
    parent_splitter=parent_splitter,
)
```

[46]:
```
retriever.add_documents(chunks_txt)
```

These are the number of large chunks.

[47]:
```
len(list(store.yield_keys()))
```

[47]: 122

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

[55]:
```
sub_docs = vectordb.similarity_search("smoking policy")
```

[56]:
```
print(sub_docs[0].page_content)
```

```
Smoking Restrictions: Smoking inside company buildings, offices, meeting rooms,
and other enclosed spaces is strictly prohibited. This includes electronic
cigarettes and vaping devices.
```

Then, retrieve the relevant large chunk.

[57]:
```
retrieved_docs = retriever.invoke("smoking policy")
print(retrieved_docs[0].page_content)
```

```
Smoking Restrictions: Smoking inside company buildings, offices, meeting rooms,
and other enclosed spaces is strictly prohibited. This includes electronic
cigarettes and vaping devices.
```

2 Exercises

2.0.1 Exercise 1

2.0.2 Retrieve top 2 results using vector store-backed retriever

Can you retrieve the top two results for the company policy document for the query "smoking policy" using the Vector Store-Backed Retriever?

[70]:
```
# Your code here
vectordb = Chroma.from_documents(documents=chunks_txt,␣
 ↪embedding=watsonx_embedding())
retriever = vectordb.as_retriever(search_kwargs={"k": 4})
query = "smoking policy"
docs = retriever.invoke(query)
```

```
docs
```

[70]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content='Smoking
Restrictions: Smoking inside company buildings, offices, meeting rooms, and
other enclosed spaces is strictly prohibited. This includes electronic
cigarettes and vaping devices.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='Smoking
Restrictions: Smoking inside company buildings, offices, meeting rooms, and
other enclosed spaces is strictly prohibited. This includes electronic
cigarettes and vaping devices.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='Smoking
Restrictions: Smoking inside company buildings, offices, meeting rooms, and
other enclosed spaces is strictly prohibited. This includes electronic
cigarettes and vaping devices.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='Smoking
Restrictions: Smoking inside company buildings, offices, meeting rooms, and
other enclosed spaces is strictly prohibited. This includes electronic
cigarettes and vaping devices.')]
```

Click here for Solution

```
vectordb = Chroma.from_documents(documents=chunks_txt, embedding=watsonx_embedding())
retriever = vectordb.as_retriever(search_kwargs={"k": 2})
query = "smoking policy"
docs = retriever.invoke(query)
docs
```

2.0.3 Exercise 2

2.0.4 Self-Querying Retriever for a query

Can you use the Self Querying Retriever to invoke a query with a filter?

[71]:
```
# Your code here
vectordb = Chroma.from_documents(docs, watsonx_embedding())

retriever = SelfQueryRetriever.from_llm(
    llm(),
    vectordb,
    document_content_description,
    metadata_field_info,
)

# This example specifies a query with filter
retriever.invoke(
    "I want to watch a movie directed by Christopher Nolan"
)
```

[71]:
```
[Document(metadata={'director': 'Christopher Nolan', 'rating': 8.2, 'year':
2010}, page_content='Leo DiCaprio gets lost in a dream within a dream within a
dream within a …')]
```

[73]:
```
retriever.invoke(
    "Can you recommend me a movie with Humphrey Bogart?"
)
```

[73]:
```
[Document(metadata={'source': 'companypolicies.txt'}, page_content='the
designated HR representative. The organization is committed to a timely and
confidential investigation of such complaints.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='the
designated HR representative. The organization is committed to a timely and
confidential investigation of such complaints.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='the
designated HR representative. The organization is committed to a timely and
confidential investigation of such complaints.'),
 Document(metadata={'source': 'companypolicies.txt'}, page_content='the
designated HR representative. The organization is committed to a timely and
confidential investigation of such complaints.')]
```

Click here for a Solution

```
# You might encouter some errors or blank content when run the following code.
# It is becasue LLM cannot get the answer at first. Don't worry, re-run it several times you w

vectordb = Chroma.from_documents(docs, watsonx_embedding())

retriever = SelfQueryRetriever.from_llm(
    llm(),
    vectordb,
    document_content_description,
    metadata_field_info,
)

# This example specifies a query with filter
retriever.invoke(
    "I want to watch a movie directed by Christopher Nolan"
)
```

2.1 Authors

Kang Wang

Kang Wang is a Data Scientist in IBM. He is also a PhD Candidate in the University of Waterloo.

2.1.1 Other Contributors

Joseph Santarcangelo

Joseph has a Ph.D. in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

```
{## Change Log}
```

```
{|Date (YYYY-MM-DD)|Version|Changed By|Change Description||-|-|-|-||2024-07-29|0.1|Kang Wang|Create the lab|}
```