= RAG Pipeline
:order: 3
:type: lesson
:branch: main

You can use a retriever as part of a RAG (Retrieval-Augmented Generation) pipeline to provide context to a LLM.

In this lesson, you will use the vector retriever you created to pass additional context to an LLM allowing it to generate more accurate and relevant responses.

Open the `genai-fundamentals/vector_rag.py` file and review the program:

[source,python]
.vector_rag.py
----
include::{repository-raw}/{branch}/genai-fundamentals/vector_rag.py[tag=**]

The program includes the code to connect to Neo4j and create the vector retriever.

You will add the code to:

. Create and configure the LLM
. Create the GraphRAG pipeline to use the vector retriever
. Submit a query to the RAG pipeline
. Parse the results

== LLM

You will need an LLM to generate the response based on the users query and the context provided by the vector retriever.

Create the LLM using the `OpenAILLM` class from the `neo4j_graphrag` package:

[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag.py[tag=import-llm]

include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag.py[tag=llm]

The LLM is configured to use `gpt-4o`.
You can change the configuration to another link:https://platform.openai.com/docs/models[OpenAI model^] by changing the `model` parameter.

You can also change the `temperature` to control the randomness of the generated response, by providing a value in the `model_params`.
A value of `0` will make responses more deterministic, while a value closer to `1` will make responses more random.

[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag.py[tag=llm-temp]

[NOTE]
The link:https://neo4j.com/docs/neo4j-graphrag-python/current/api.html#llm[`neo4j-graphrag` package supports multiple LLM models^] and the ability to create your own interface.

== GraphRAG pipeline

The `GraphRAG` class allows you to create a RAG pipeline including a retriever and an LLM.

Create the `GraphRAG` pipeline using the `retriever` and the `llm` you created:

[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag.py[tag=import-graphrag]

include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag.py[tag=graphrag]

The `GraphRAG` pipeline will:

. Use the retriever to find relevant context based on the user's query.
. Pass the user's query and the retrieve context to the LLM.

== Search

You can use the `search` method to submit a query.

[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag.py[tag=search]

The `search` method takes the user's query and returns the generated response from the LLM.
You can also specify addition `retriever_config`, such as the number of results to return.

```
[%collapsible]
.Click to view the complete code
====
[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag
.py[tags=**;!search_return_context;!llm-temp]
----
====
```

Run the program to search for similar movies based on a plot.

Return context

You can also return the context that was used to generate the response.
This can be useful in understanding how the LLM generated the response.

Add the `return_context=True` parameter to the `search` method:

```
[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_rag
.py[tag=search_return_context]
----
```

The `retriever_result` is returned along with the generated response.

Experiment with different queries and see how the LLM generates responses based on the context provided by the vector retriever.

```
[IMPORTANT]
.Randomness
====
Responses from the LLM will not be consistent and may vary each time you
run the program.
Experiment with the `temperature` parameter in the LLM configuration to
see how it affects the randomness of the generated response.
====
```

Check Your Understanding

```
include::questions/1-context.adoc[leveloffset=+2]
```

```
[.summary]
== Lesson Summary
```

In this lesson, you learned how to create a GraphRAG pipeline using a vector retriever and an LLM.

In the next lesson, you will build a more advanced retriever that combines vector search with graph traversal and relationships to enhance information retrieval.