= Graph-Enhanced Vector Retriever
:order: 4
:type: lesson
:branch: main

To take advantage of the relationships in the graph, you can create a retriever that uses both vector search and graph traversal to find relevant data.

The `VectorCypherRetriever` allows you to perform vector searches and then traverse the graph to find related nodes or entities.

Open the `genai_fundamentals/vector_cypher_rag.py` file and review the code:

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

The program includes all the code to connect to Neo4j, create the `embedder`, `llm`, and `GraphRAG` pipeline.

Your task is to:

. Configure the Cypher retrieval query that will traverse the graph
. Create the `VectorCypherRetriever` retriever.

== Retrieval Query

The retrieval query is a Cypher query that will be used to get data from the graph after the nodes are returned by the vector search.

The query receives the `node` and `score` variables yielded by the vector search.

Add this retrieval query to the code:

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

The query traverses the graph to find related nodes for genres and actors, as well as sorting the results by the user rating.

== Retriever

You can now use the `VectorCypherRetriever` class to create a retriever that will perform the vector search and then traverse the graph:

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

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

The retriever requires the vector index name (`moviePlots`), the retrieval query, and the `embedder` to encode the query.

[%collapsible]
.Click to view the complete code
====
[source,python]
----
include::{repository-raw}/{branch}/genai-fundamentals/solutions/vector_cypher_rag.py[tag=**]
----
====

== Context

When you run the code, it will complete a vector search for the provided query and then traverse the graph to find related nodes.

The additional context allows the LLM to generate more accurate responses based on the additional data in the graph.

[TIP]
.Transparency
====
The context is returned after the response, allowing you to see what data was used to generate the response.

This transparency is important for understanding how the LLM arrived at its response and for debugging purposes.
====

When sent the query _"Find the highest rated action movie about travelling to other planets"_, the GraphRAG pipeline will follow these steps:

. Perform a vector search for movie plots related to _travelling to other planets_.
. Run the retrieval query to find related actors, genres, and user ratings.
. Pass the retrieved data to the LLM to generate a response.

You can expect a response to be based on:

* Travelling to other planets.
```

* The comedy genre.
* With the highest user rating (not the vector similarity score).

A typical response might be _"The highest rated action movie about traveling to other planets is "Aliens," with a user rating of 3.92"_

Test the code with different queries relating to movies, actors, and genres, such as:

* _Find a comedy movie about vampires_
* _Who acts in drama movies about romance and love?_
* _What genres are represented about movies where the hero fails his mission?_

[TIP]
====
If you are having unpredictable responses try modifying the temperature of the LLM:

[source,python]
----
llm = OpenAILLM(
    model_name="gpt-4o",
    model_params={"temperature": 0.5}
)

====

== Optional challenge

Modify the retrieval query to include the directors of the movies in the context.

Directors can be found using the pattern `(node)<-[:DIRECTED]-(director)`.

Try queries relating to directors, such as _"Who has directed movies about weddings?"_

== Continue

When you are ready, continue to the next lesson.

read::Continue[]

[.summary]
== Lesson Summary

In this lesson, you learned how to create a retriever that uses both vector search and graph traversal to find relevant data

In the next lesson, you will create a text to cypher retriever to find relevant data based on natural language queries.