

```
= Improving Semantic Search
:order: 3
:type: lesson
:optional: true
```

In this module, you have learned about the benefits and drawbacks of Semantic Search using an underlying vector index.

You received an answer when querying the vector index for a similar movie, but it is difficult to tell whether that is the best movie in the database.

When using vector-backed semantic search, you are placing a lot of emphasis on the underlying model to provide a good similarity.

The context from which the similarity scores are provided is an important factor here.

The embeddings were trained on a generic dataset, and as such, may identify two movies with a main character called Jack who likes fruit as movies with a close similarity.

This is correct in one sense; they are both movies and very similar when compared to a cricket bat or a vanilla yoghurt.

On the other hand, if you are recommending a movie to pass away a rainy afternoon, a fairy tale for children is very different to a horror film.

== Incorporating Graph Features

Graph features can enrich vector-backed semantic search by providing structural and relational context to data.

By leveraging the connections in graphs, semantic search can draw upon relationships and hierarchies between entities, enhancing the depth and relevance of search results.

While vectors capture semantic nuances, adding graph structures allows for a more holistic understanding of data, considering its inherent meaning and position in a broader knowledge network.

//TODO - Do we want to embed this video? I think it will age quickly

// This topic is out of the scope of this module, but for more information, watch link:<https://www.youtube.com/watch?v=bRD09ndyJNs>[Going Meta - Ep 21: Vector-based Semantic Search and Graph-based Semantic Search^], in which Dr Jesus Barrasa and Alexander Erdl explore the differences between Vector-based Semantic Search and Graph-based Semantic Search.

// image::images/jesus-barrassa.png[Dr Jesus Barrasa]
// _Dr Jesus Barrasa_

== Using Feedback

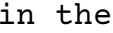
A more straightforward approach may be to use user feedback to gradually re-rank and curate the content returned by a vector index search.

GraphAcademy has a chatbot that uses this approach to improve the quality of its responses.

[NOTE]

.The GraphAcademy Chatbot

====

You can access the GraphAcademy chatbot by clicking on the chat icon  in the bottom right corner of the GraphAcademy website.

====

When a user asks a question, the server generates an embedding of the question and uses the vector index to identify relevant content.

This approach works well in most cases, but now and again, the index fails to return relevant content, in which case the server instructs the LLM to respond to the user with an apology.

GraphAcademy stores questions, responses, and the user's feedback in the database. Over time, a better picture of which documents do or do not answer specific queries emerges.

The chatbot can use this context to improve future responses. For example, when a user asks a question, the chatbot can compare it to previous questions and exclude any suggested documents where the answer was previously unhelpful.


[TIP]

.Want to know more?

====

The article

link:<https://medium.com/neo4j/building-an-educational-chatbot-for-graphacademy-with-neo4j-f707c4ce311b>[Building an Educational Chatbot for GraphAcademy with Neo4j Using LLMs and Vector Search^] describes in more detail how we ingested Neo4j Documentation and GraphAcademy lessons, using embeddings of the content to populate a vector index.

[The GraphAcademy Chatbot Data Model]

====

read::Continue[]

[.summary]

== Module Summary

In this module, you have learned how to implement vector search in Neo4j.

You have learned how to create a vector index using ``CREATE VECTOR INDEX``, set vector properties using the ``db.create.setVectorProperty()`` procedure, and query the vector index using the ``db.index.vector.queryNodes()`` procedure.

You also explored the benefits and potential drawbacks of Vector-based Semantic Search.

In the next module, you will get hands-on with Langchain, a framework designed to simplify the creation of applications using large language models.