```asciidoc
= Grounding LLMs
:order: 3
:type: lesson

In the previous lesson, you learned about the potential methods that you
could use to avoid _Hallucination_, including _Grounding_.

In this lesson, you will explore grounding and an approach known as
_Retrieval Augmented Generation_ (RAG).

[.video]
video::Le_j18w2oGI[youtube,width=560,height=315]

[.transcript]
== Grounding LLMs with RAG

Grounding is the process of providing context to an LLM to improve the
accuracy of its responses and reduce the likelihood of hallucinations.
For developers and data scientists, grounding usually offers the highest
impact for the lowest effort.

Training a Large Language Model has a high computational cost.
According to Wikipedia, OpenAI's GPT-3 model was
link:https://en.wikipedia.org/wiki/GPT-3[trained on 175 billion
parameters^], and the resulting trained model takes 800GB to store.

Retraining a model on new data would be expensive and time-consuming.
A model may take weeks or months to train.

For example, if you were building a chatbot for a news agency it would be
impractical to train an entirely new LLM and providing real-time updates
on breaking news would be near impossible.

=== Retrieval Augmented Generation

A news agency could train a model on static data and supplement the
pre-existing knowledge base with real-time data retrieved from up-to-date
sources.

This approach is known as **Retrieval Augmented Generation**, or **RAG**

RAG combines the strengths of large-scale language models with external
retrieval or search mechanisms, enabling relevant information from vast
datasets to be dynamically fed into the model during the generation
process, thereby enhancing its ability to provide detailed and
contextually accurate responses.

In summary, by adding content from additional data sources, you can
improve the responses generated by an LLM.

The main benefit of RAG is its **enhanced accuracy**.
By dynamically pulling information from external, domain-specific sources,
a response grounded by RAG can provide more detailed and contextually
```

```asciidoc
accurate answers than a standalone LLM.

RAG provides additional benefits of:

* Increased transparency, as the sources of the information can be stored
and examined.
* Security, as the data sources can be secured and access controlled.
* Accuracy and timeliness, as the data sources can be updated in
real-time.
* Access to private or proprietary data

[WARNING]
.Bad data in, bad data out
====
When prompting the LLM to respond based on the context provided, the
answer will always be as good as the provided context.

If your prompt suggests pineapple as a pizza topping, don't be surprised
if it suggests that you order a Hawaiian Pizza.
====

=== Access to Real-Time Data

RAG could support the news agency chatbot by:

. Accessing real-time news feeds
. Pulling recent headlines or news articles from a database,
. Giving this additional context to the LLM

News articles stored in a knowledge graph would be ideal for this use
case. A knowledge graph could pass the LLM detail about the relationship
between the entities involved and the article's metadata.

For example, when asking about the results of a recent election, the
knowledge could provide additional context about the candidates, news
stories relating to them, or interesting articles from the same author.

image::images/llm-news-agency-knowledge-graph.svg[A news agency chatbot
using a knowledge graph to provide context to an LLM. The knowledge graph
takes data from a news API and previous articles.]

During this course, you will explore methods for implementing RAG with
Neo4j, including:

* Semantic Search
* Zero-shot and few-shot learning and prompts
* Text embedding and vector indexes with unstructured data
* Cypher generation to gather structured data from a knowledge graph


== Check Your Understanding

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

[.summary]
== Lesson Summary

In this lesson, you learned about the importance of grounding to enhance LLM accuracy, the computational challenges of constantly training large models, and how Retrieval Augmented Generation (RAG) combines LLMs with external data for improved responses with increased transparency.

In the next module, you will learn how to implement semantic search and use vector indexes in Neo4j.