



Building More Accurate Chatbots

A technical guide

Hi, I'm Adam 🖐️

Developer Advocate at Neo4j

Hi, I'm Adam 🖐️

Developer Advocate at Neo4j

2004

Started building websites and mobile apps

2015

Started using Neo4j

2017

Joined Neo4j as a Professional Services Consultant

2020

Moved to the Developer Relations Team

Looking after Neo4j GraphAcademy

 [/in/adamcowley](https://www.linkedin.com/in/adamcowley)



Generative AI & LLMs

LLMs are great

LLMs have a problem



LLMs make things up

LLMs make things up



Hallucination

Hallucination (artificial intelligence)

Article [Talk](#)

From Wikipedia, the free encyclopedia

In the field of [artificial intelligence](#) (AI), a **hallucination** or **artificial hallucination** (also called **confabulation**^[1] or **delusion**^[2]) is a response generated by an AI which contains false or misleading information presented as fact.^{[3][4][5]} For example, a hallucinating [chatbot](#) might, when asked to generate a [financial report](#) for a company, falsely state that the company's revenue was \$13.6 billion (or some other number apparently "plucked from thin air").^[6] Such phenomena are termed "hallucinations", in loose analogy with the phenomenon of [hallucination in human psychology](#). However, one key difference is that human hallucination is usually associated with false *percepts*, but an AI hallucination is associated with the category of unjustified responses or beliefs.^[5] Some researchers believe the specific term "AI hallucination" unreasonably anthropomorphizes computers.^[1]

AI hallucination gained prominence around 2022 alongside the rollout of certain [large language models](#) (LLMs) such as [ChatGPT](#).^[7] Users complained that such bots often seemed to pointlessly embed plausible-sounding random falsehoods within their generated content.^[8] By 2023, analysts considered frequent hallucination to be a major problem in LLM technology, with some estimating chatbots hallucinate as much as 27% of the time.^{[9][10]}



Message ChatGPT...



ChatGPT can make mistakes. Consider checking important information.

Lawyers used ChatGPT to help with a case - it backfired massively

OpenAI's ChatGPT has wowed users with its ability to write essays and speeches, complete homework, crack computer code and pen poems - but it's also prone to completely making things up.

Two New York lawyers have been fined after submitting a legal brief with fake case citations generated by ChatGPT.

Steven Schwartz, of law firm Levidow, Levidow & Oberman, admitted using the chatbot to research the brief in a client's personal injury case against airline Avianca.

He had used it to find legal precedents supporting the case, but lawyers representing the Colombian carrier told the court they could not find some examples cited - understandable, given they were almost entirely fictitious.

Several of them were completely fake, while others misidentified judges or involved airlines that did not exist.

Company

Location

Product

Person





LLMs are designed to
generate human-like text

Why do LLMs Hallucinate?

Why do LLMs hallucinate?

1. Temperature

Corresponds to the randomness the model uses when generating a response.

2. Missing Information

Training LLMs is intricate and time-intensive. As such, these models might lack the most recent information.

3. Model Training & Complexity

Complex, untraceable decisions can be made on potentially flawed or biased training data.



You

How do I use the vector search index in Neo4j?



ChatGPT

To use a vector search index in Neo4j, you typically follow these steps:

1. **Install and Setup Neo4j:** Ensure you have Neo4j installed. Vector search in Neo4j is generally available through plugins or additional libraries, so you might need to install those as well.
2. **Prepare Your Data:** Your data should include vectors. These can be generated using various machine learning models depending on your use case, such as word embeddings for text data or image embeddings for visual data.
3. **Create a Vector Search Index:** Once your data is ready, you can create a vector search index. This is done using Cypher, Neo4j's query language. The syntax for creating an index may vary based on the plugin or library you're using.

For example, using the `'bloom'` plugin, the command might look like:

```
cypher
```

[Copy code](#)

```
CREATE INDEX my_vector_index FOR (n:Node) ON (n.embedding) OPTIONS {indexProvider: 'bloom'}
```

Here, `'Node'` is your node label, and `'embedding'` is the property holding your vector.

4. **Query Using the Vector Index:** With the index created, you can perform vector similarity searches. The exact query syntax will depend on the plugin and the type of search you're conducting (e.g., nearest neighbors, range search).

A simple example might be:

```
cypher
```

[Copy code](#)

```
MATCH (n:Node)
WHERE bloom.similarity(n.embedding, $queryVector) > 0.7
RETURN n
```

Here, `'$queryVector'` is the vector you're comparing against, and `'0.7'` is the similarity threshold.

✓ Coherent

☐ Confident

✗ Correct



You

What was your training data cut-off date?



ChatGPT

My training data includes information up to April 2023. Any events, developments, or data published after that date are not included in my current knowledge base. If you have questions about recent events or the latest information in a specific field, I can assist with online searches where possible.



[\(Neo4j Blog\)←\[:BACK\]](#)

Neo4j's Vector Search: Unlocking Deeper Insights for AI-Powered Applications



Sudhir Hasbe, Chief Product Officer, Neo4j

Aug 22 · 4 mins read

for AI-Powered Applications





You

How do I use the vector search index in



ChatGPT

To use a vector search index in Neo4j, y

1. **Install and Setup Neo4j:** Ensure you have Neo4j installed and running. You may need to install additional plugins or libraries through plugins or additional libraries.
2. **Prepare Your Data:** Your data should be in a format compatible with the learning models depending on your use case. For example, you might have embeddings for visual data.

3. **Create a Vector Search Index:** Once your data is ready, you can create a vector search index. This is done using Cypher, Neo4j's query language. The syntax for creating an index may vary based on the plugin or library you're using.

For example, using the 'bloom' plugin,

```
cypher
```

```
CREATE INDEX my_vector_index F
```

Here, 'Node' is your node label, and 'e

4. **Query Using the Vector Index:** With the index created, you can perform vector similarity searches. The exact query syntax will depend on the plugin and the type of search you're conducting (e.g., nearest neighbors, range search).

A simple example might be:

```
cypher
```

```
MATCH (n:Node)
```

```
WHERE bloom.similarity(n.embedding, $queryVector) > 0.7
```

```
RETURN n
```

Here, '\$queryVector' is the vector you're comparing against, and '0.7' is the similarity threshold.

3. **Create a Vector Search Index:** Once your data is ready, you can create a vector search index. This is done using Cypher, Neo4j's query language. The syntax for creating an index may vary based on the plugin or library you're using.

For example, using the 'bloom' plugin, the command might look like:

```
cypher
```

[Copy code](#)

```
CREATE INDEX my_vector_index FOR (n:Node) ON (n.embedding) OPTIONS {indexProvider: 'blo
```

4. **Query Using the Vector Index:** With the index created, you can perform vector similarity searches. The exact query syntax will depend on the plugin and the type of search you're conducting (e.g., nearest neighbors, range search).

A simple example might be:

```
cypher
```

[Copy code](#)

```
MATCH (n:Node)
```

```
WHERE bloom.similarity(n.embedding, $queryVector) > 0.7
```

```
RETURN n
```

Here, '\$queryVector' is the vector you're comparing against, and '0.7' is the similarity threshold.

Missing Information

- **Intricate Training Processes**

LLMs undergo a complex, time-consuming training process with large datasets.

- **Data Timeliness Issue**

LLMs might lack up-to-date information due to reliance on historical data.

- **Training Cut-off**

An LLM last updated in September 2023 wouldn't know about events or advancements after that date.

- **Knowledge Gaps**

Models may miss out on recent events or niche topics not covered in their training set.

Knowledge Gaps

- **No Niche Data**

The information may be missing from the initial dataset.

- **Lack of information**

There may not be enough information available for the model to reach a conclusion on a topic.

- **No Knowledge Enterprise Data**

The training dataset will most likely contain no enterprise data.

What about Enterprise data?

Enterprise Data

- **Training your own LLM**

An expensive process that requires expertise within the company.

- **Training Cut-Off**

As soon as you generate the data to train the model, it will be out of date.

Improving Accuracy

Avoiding Hallucination

1. Prompt Engineering

Iteratively refining instructions to achieve more consistent results.

2. In-Context Learning

Provide examples to guide AI for accurate, task-specific responses.

Also known as Few-Shot Prompting

3. Fine-tuning

Providing additional training to an LLM after its primary training phase.

4. Grounding

Providing the LLM with access to external reference data and data sources.

Grounding

Providing the LLM with access to external reference data and data sources.

Computer Science > Computation and Language

[Submitted on 22 May 2020 (v1), last revised 12 Apr 2021 (this version, v4)]

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela

Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable access mechanism to explicit non-parametric memory can overcome this issue, but have so far been only investigated for extractive downstream tasks. We explore a general-purpose fine-tuning recipe for retrieval-augmented generation (RAG) -- models which combine pre-trained parametric and non-parametric memory for language generation. We introduce RAG models where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We compare two RAG formulations, one which conditions on the same retrieved passages across the whole generated sequence, the other can use different passages per token. We fine-tune and evaluate our models on a wide range of knowledge-intensive NLP tasks and set the state-of-the-art on three open domain QA tasks, outperforming parametric seq2seq models and task-specific retrieve-and-extract architectures. For language generation tasks, we find that RAG models generate more specific, diverse and factual language than a state-of-the-art parametric-only seq2seq baseline.

Comments: Accepted at NeurIPS 2020

Subjects: **Computation and Language** (cs.CL); Machine Learning (cs.LG)

Cite as: arXiv:2005.11401 [cs.CL]

(or arXiv:2005.11401v4 [cs.CL] for this version)

<https://doi.org/10.48550/arXiv.2005.11401> 

Submission history

From: Patrick Lewis [[view email](#)]

[v1] Fri, 22 May 2020 21:34:34 UTC (698 KB)

[v2] Mon, 7 Dec 2020 16:23:06 UTC (767 KB)

Access Paper:

- [Download PDF](#)
- [PostScript](#)
- [Other Formats](#)

([view license](#))

Current browse context:

cs.CL

< prev | next >

new | recent | 2005

Change to browse by:

cs

cs.LG

References & Citations

- [NASA ADS](#)
- [Google Scholar](#)
- [Semantic Scholar](#)

6 blog links ([what is this?](#))

DBLP – CS Bibliography

[listing](#) | [bibtex](#)

Ethan Perez
Aleksandra Piktus
Fabio Petroni
Vladimir Karpukhin
Naman Goyal

Export BibTeX Citation

Bookmark



Retrieval Augmented Generation

R

A

G

Retrieval

Augmented

Generation

Retrieval Augmented Generation

G A R

Retrieval Augmented Generation

G A R

Instruct an LLM
to generate text

Retrieval Augmented Generation

G

Instruct an LLM
to generate text

A

that has been
augmented *or*
improved

R

Retrieval Augmented Generation

G

Instruct an LLM
to generate text

A

that has been
augmented *or*
improved

R

by the retrieval
of additional
information

A tongue-in-cheek example



You

Context:

The sky is red

Question:

What colour is the sky?



ChatGPT

The sky is red.



RAG & Graphs

The two sides of RAG

**Semantic
Search**

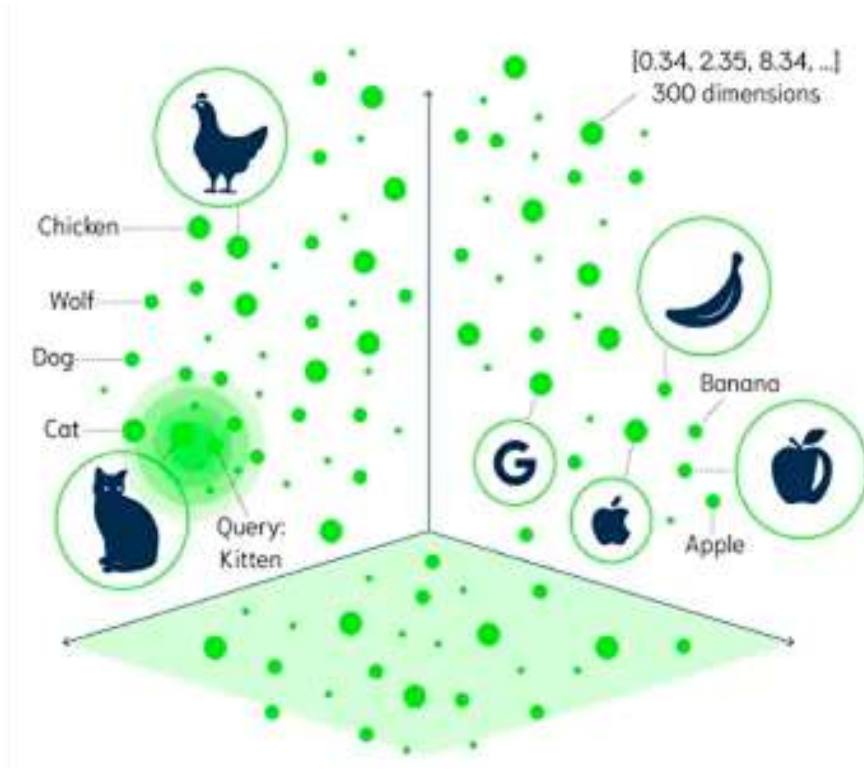
**Cypher
Generation**

The two sides of RAG

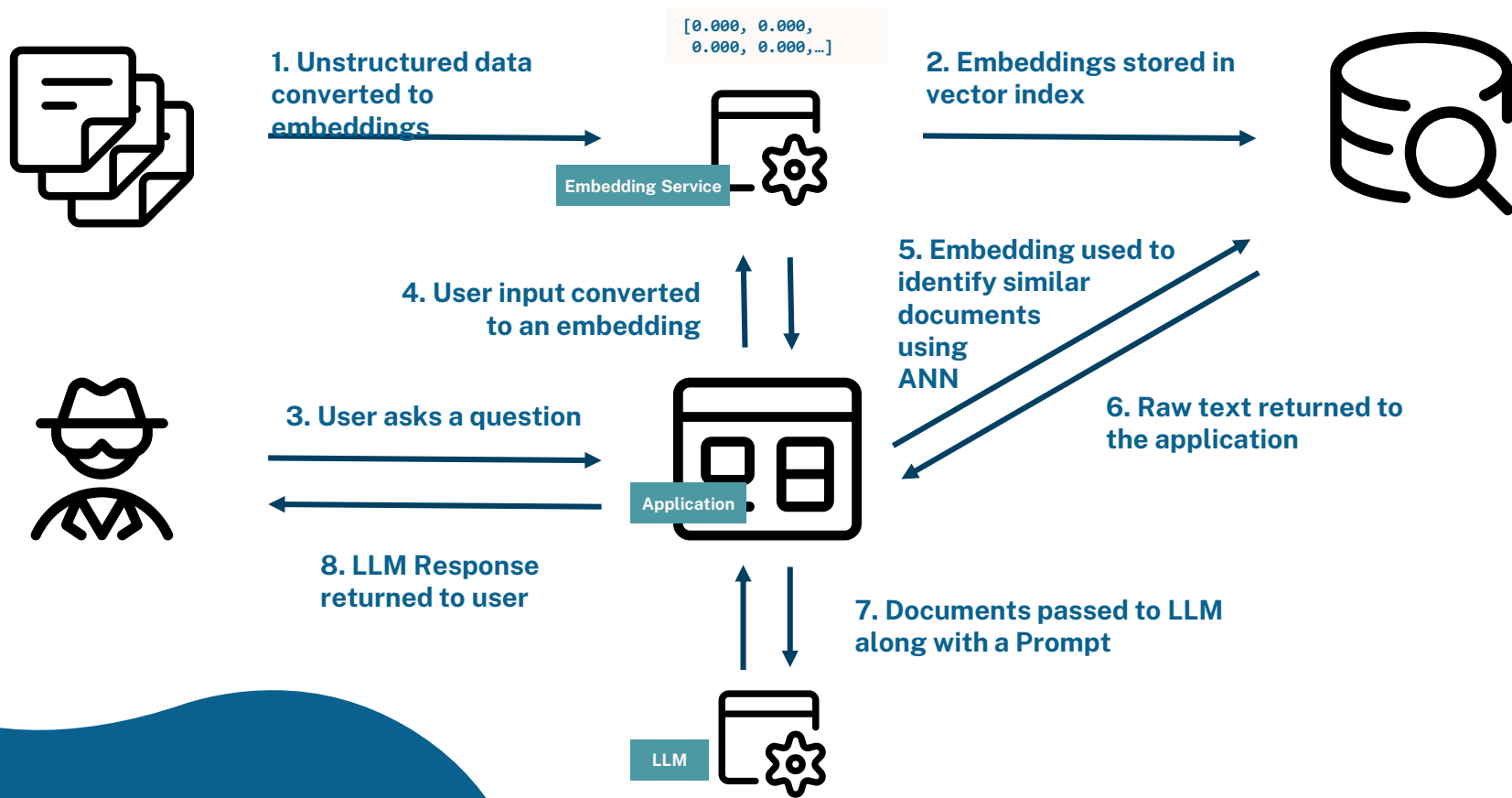
**Semantic
Search**

**Cypher
Generation**

Semantic Search with Vectors



How Vector Searches Work



Creating a Vector Search Index

```
CALL db.index.vector.createNodeIndex(  
    'moviePlots',           // The name of the index  
    'Movie',               // The node label on which to index  
    'embedding',           // The property that holds the embedding  
    1536,                  // The dimensions of the embedding - text-  
embedding-ada-002  
    'cosine'               // The similarity function to use - cosine or  
euclidean  
)
```

Querying a Vector Search Index

```
CALL db.index.vector.queryNodes(  
    'moviePlots',           // The name of the index  
    10,                     // The number of approximate nearest  
    neighbours to return  
    $embedding              // A list of floats representing an embedding  
) YIELD node, score
```

An example query

```
MATCH (m:Movie {title: 'Toy Story'})
```

```
WITH m LIMIT 1
```

```
CALL db.index.vector.queryNodes('moviePlots', 6, m.embedding)
```

```
YIELD node, score
```

```
RETURN node.title AS title, node.plot AS plot, score
```


Evaluating Vector Semantic Search

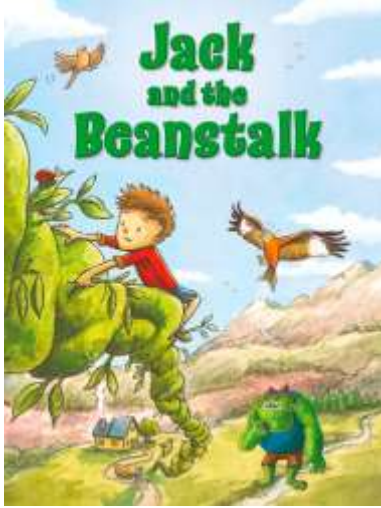
Positives:

- Relatively easy to implement
- Capable of disambiguating terms based on context

Negatives:

- Generating vectors can be cost and computationally expensive
- Vectors need to be constantly generated
- Lack of explainability in results

Rainy Day Movie Recommendations



Protagonist: Jack

Locations: Isolated lodging; castle



Protagonist: Jack

Locations: Isolated lodging; hotel

The two sides of RAG

**Semantic
Search**

**Cypher
Generation**

The two sides of RAG

Semantic
Search

Cypher
Generation

Lawyers used ChatGPT to help with a case - it backfired massively

OpenAI's ChatGPT has wowed users with its ability to write essays and speeches, complete homework, crack computer code and pen poems - but it's also prone to completely making things up.

Two New York lawyers have been fined after submitting a legal brief with fake case citations generated by ChatGPT.

Steven Schwartz, of law firm Levidow, Levidow & Oberman, admitted using the chatbot to research the brief in a client's personal injury case against airline Avianca.

He had used it to find legal precedents supporting the case, but lawyers representing the Colombian carrier told the court they could not find some examples cited - understandable, given they were almost entirely fictitious.

Several of them were completely fake, while others misidentified judges or involved airlines that did not exist.

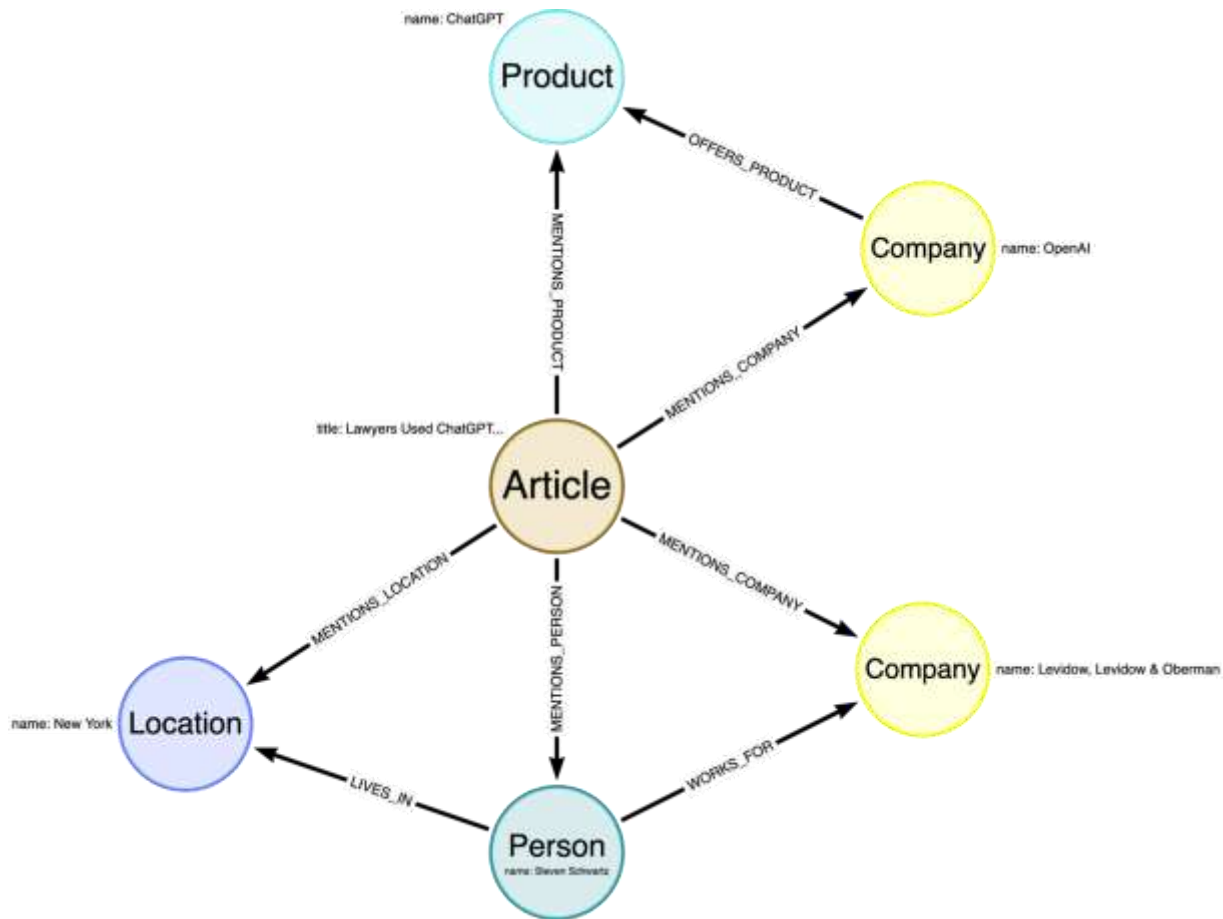
Company

Location

Product

Person

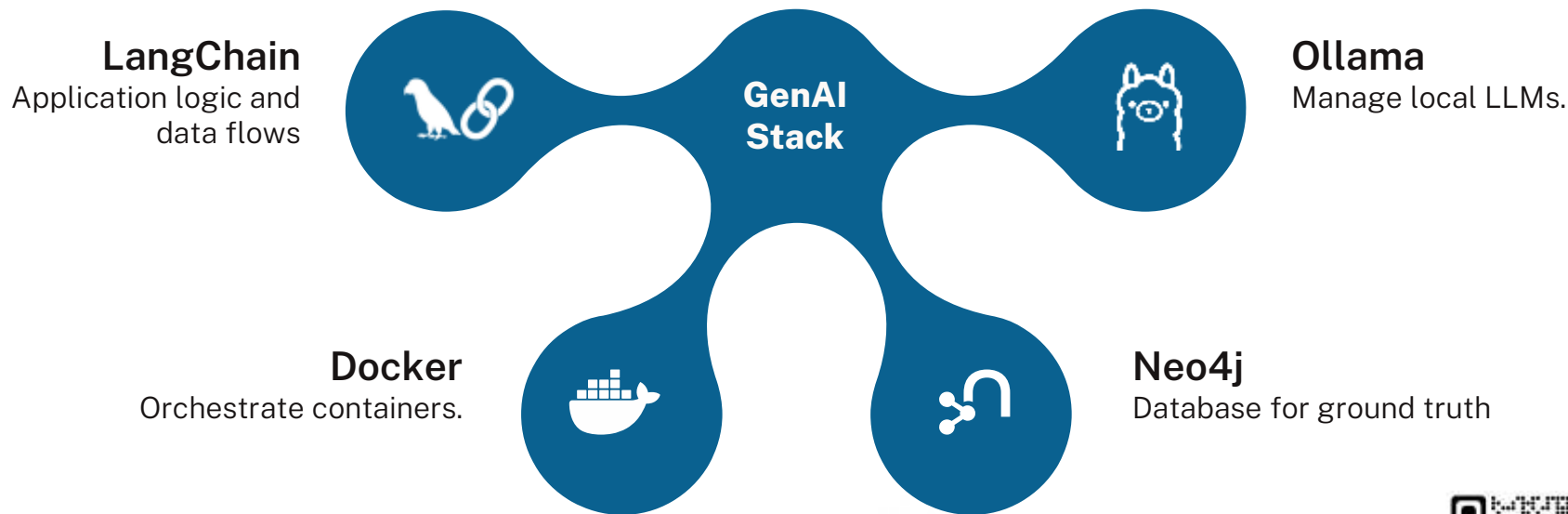




Which articles mention ChatGPT?

```
MATCH (a:Article)-[:MENTIONS_PRODUCT]→(p:Product {name: "ChatGPT"})
RETURN a.body AS text,
  a {
    .url,
    .title,
    mentions: [ (a)-[r:MENTIONS_PRODUCT|MENTIONS_COMPANY|MENTIONS_LOCATION]→(n) | n.name ]
  } AS metadata
```

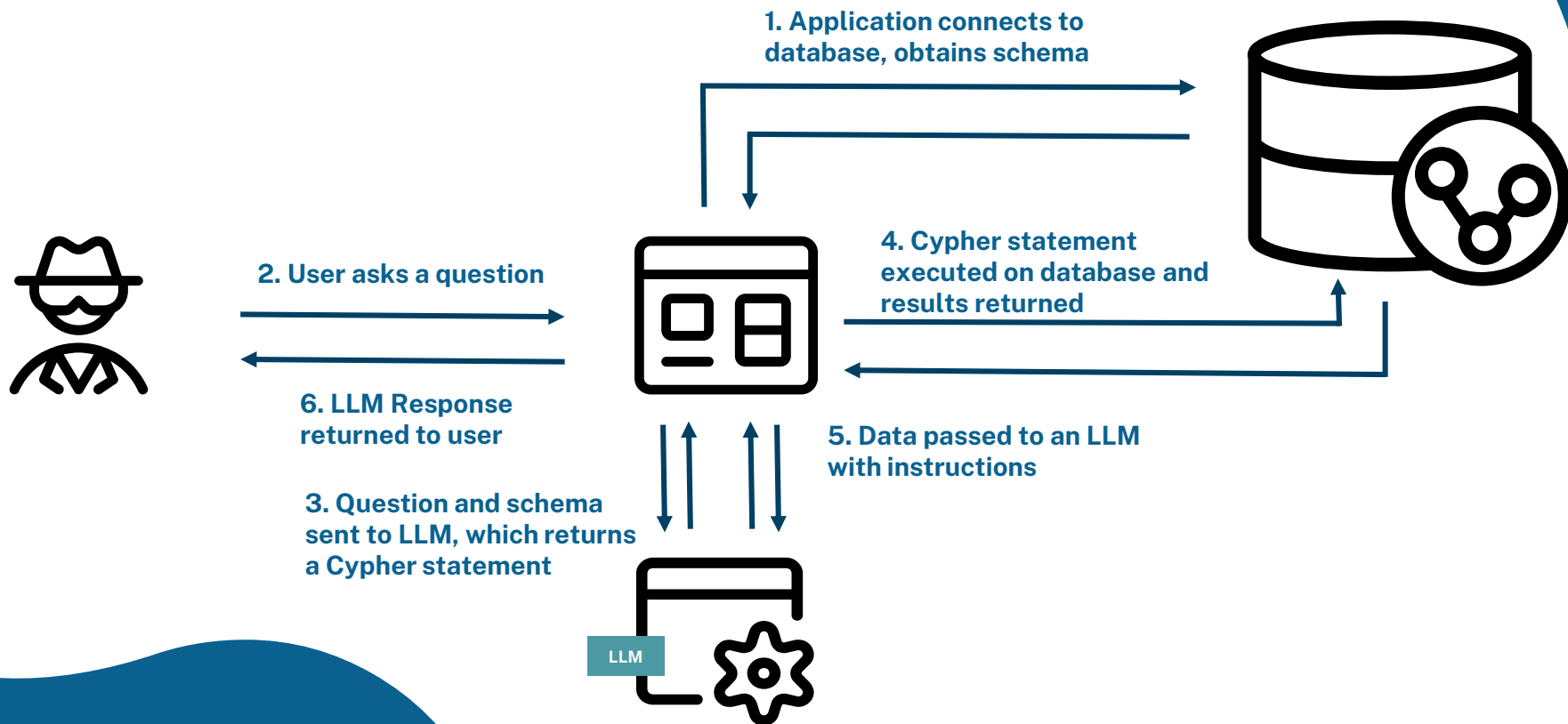
The GenAI Stack



LangChain

- Open-source framework for developing AI applications
- Build applications with Python or JavaScript (with TypeScript support)
- Lift and replace integrations for LLMs
- Hundreds of tools for third-party services and APIs
- Frequent releases and new features

How Cypher Generation Works



Initiate an LLM

Import an LLM

```
from langchain.llms import Ollama
```

Or OpenAI, etc...

```
llm = Ollama(  
    model="mistral",  
    temperature=0.2,  
)
```

Ask a question

```
response = llm("What is Neo4j?")
```

Creating Embeddings with Langchain

```
# Create an embedding model
```

```
from langchain.embeddings import OllamaEmbeddings # Or OpenAIEmbeddings etc...
```

```
embeddings = OllamaEmbeddings(model="mistral")
```

```
# Load Documents
```

```
from langchain.document_loaders import PyPDFLoader
```

```
loader = PyPDFLoader("building-knowledge-graphs.pdf")
```

```
pages = loader.load_and_split()
```

Create a Vector Index from Documents

```
from langchain.vectorstores.neo4j_vector import Neo4jVector
```

```
neo4jvector = Neo4jVector.from_documents(  
    pages,  
    embeddings,  
    url=os.getenv("NEO4J_URI"),  
    username=os.getenv("NEO4J_USERNAME"),  
    password=os.getenv("NEO4J_PASSWORD"),  
    index_name="pages",  
    node_label="Page",  
    text_node_property="embedding",  
    text_node_property="text",  
)
```

Use an Existing Vector Index

```
neo4jvector = Neo4jVector.from_existing_index(  
    embeddings,  
    url=os.getenv("NEO4J_URI"),  
    username=os.getenv("NEO4J_USERNAME"),  
    password=os.getenv("NEO4J_PASSWORD"),  
    index_name="pages"  
)
```

Retriever

```
retriever = neo4jvector.get_retriever()
```

Stuff QA Chain

```
from langchain.chains import RetrievalQA
```

```
# Create a Chain
```

```
qa_chain = RetrievalQA.from_chain_type(  
    llm,  
    chain_type="stuff",  
    retriever=retriever,  
)
```

```
# Ask a Question
```

```
qa_chain("Recommend me a movie about labradors")
```

Graph Cypher QA Chain

```
from langchain.graphs import Neo4jGraph
from langchain.chains import GraphCypherQAChain

graph = Neo4jGraph(url="neo4j://localhost:7687", username="neo4j", password="pleaseletmein")

# Create a Chain
cypher_chain = GraphCypherQAChain.from_llm(
    llm,                                     # Or cypher_llm and qa_llm for different LLMS &
models                                     models
    graph=graph
)

# Ask a Question
cypher_chain("What role did Tom Hanks play in Toy Story?")
```


Agents & Tools

```
# Register tools for the agent to use
```

```
tools = [  
    Tool.from_function(name="Cypher QA", description="...", func = cypher_qa),  
    Tool.from_function(name="Vector Search Index", description="...", func = kg_qa)  
]
```

```
# Create an agent with a list of tools and an LLM
```

```
agent = initialize_agent(  
    tools,  
    llm,  
    memory=memory,  
    verbose=True  
)
```

Advanced Features

- Provide custom Cypher generation prompt with additional instructions
- Few-shot examples for difficult Cypher statements
- Control the data that is returned from the Cypher statement
- Store the responses in the graph
- Hybrid Search - Search vectors and full-text indexes

The background is a solid blue color with large, flowing, organic shapes in a slightly darker shade of blue, creating a sense of movement and depth. The shapes are reminiscent of waves or liquid forms.

How do I learn more?

GraphAcademy — Free, Self-Paced, Hands-on Online Training

We're here to guide you on a fun and engaging journey to mastering Neo4j with free, hands-on courses.

Start with the Fundamentals

Are you a beginner or just getting started? Build your Neo4j foundation with these select courses.

Neo4j Fundamentals



Learn the basics of Neo4j and the property graph model.

Cypher Fundamentals



Learn Cypher in 1 hour

Graph Data Modeling Fundamentals



Learn how to design a Neo4j graph using best practices

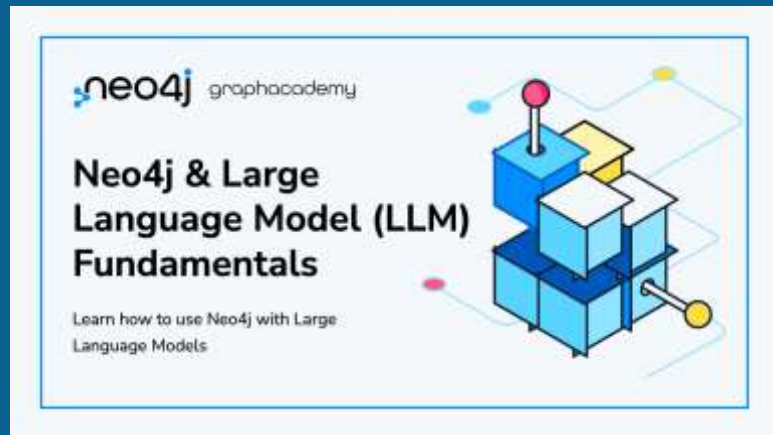
Importing CSV data into Neo4j



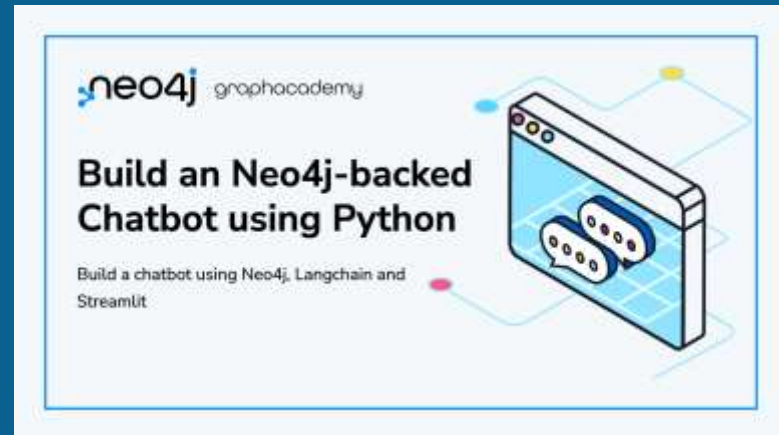
Learn how to import CSV data into Neo4j using Cypher



LLM Courses on Neo4j GraphAcademy



r.neo4j.com/ga-llm-fun



r.neo4j.com/ga-llm-py



Q&A?

r.neo4j.com/ga-llm-fun



r.neo4j.com/ga-llm-py

