RAG

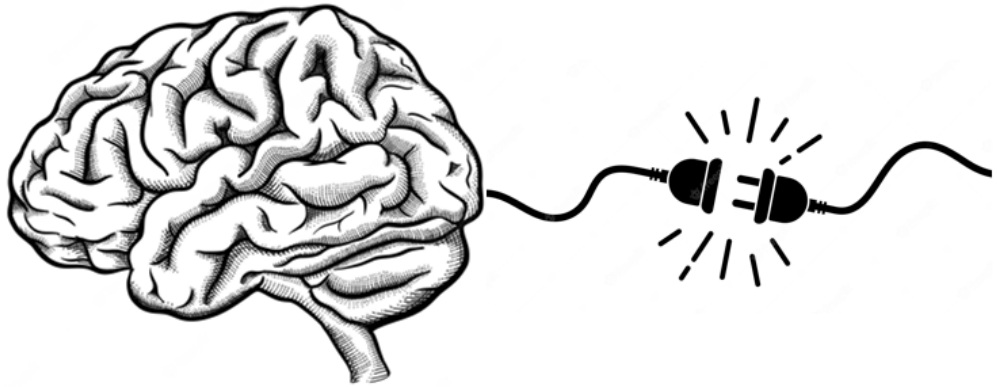
(Retrieval augmented generation)


Introducción

Retrieval Augmented Generation

RAG es una tecnología de vanguardia que mejora la eficacia de los LLMs.

A pesar de sus notables capacidades, los LLM están plagados de retos que los hacen inadecuados para tareas específicas, en particular las que requieren información actualizada o específica de un dominio.



LLMs

Datos propios

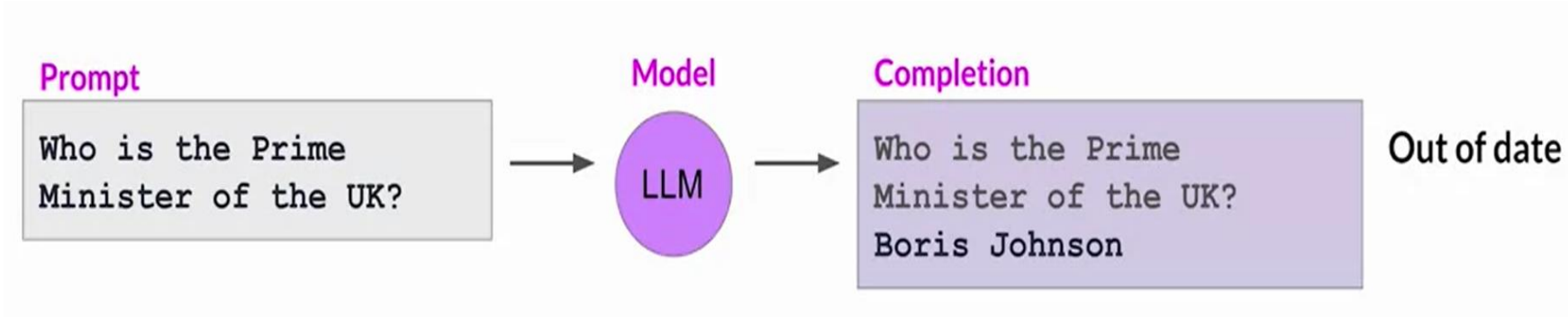
Large Language Models

Son estáticos: los LLM están "congelados en el tiempo" y carecen de información actualizada ya que fueron entrenados con datos de hasta X fecha.

Se quedan estancados en un momento concreto.

Por ejemplo, el "punto de corte" de los datos de formación para ChatGPT fue septiembre de 2021.

Esto significa que carece de información actualizada sobre acontecimientos o desarrollos ocurridos después de esta fecha. Por lo tanto, si se le pregunta a ChatGPT por algo que haya ocurrido recientemente, no sólo no dará una respuesta objetiva, sino que podría inventar una respuesta plausible pero incorrecta.



Large Language Models

Carecen de conocimientos específicos de un dominio: los LLM están capacitados para tareas generalizadas, lo que significa que no conocen los datos privados de las empresas, sus clientes o sus requisitos.

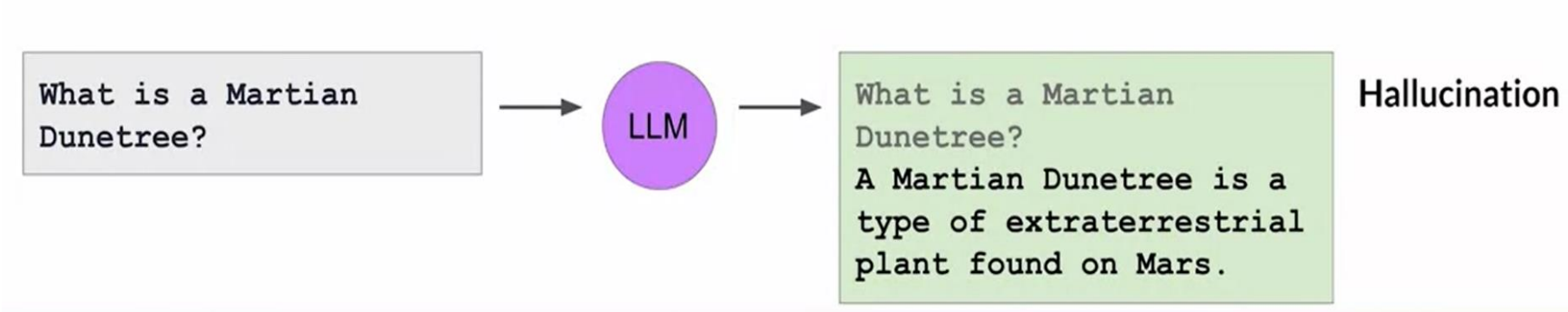
En consecuencia, tienden a alucinar cuando se les hacen preguntas específicas del dominio o de la empresa.

Large Language Models

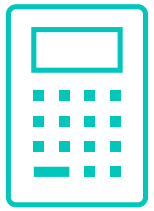
Funcionan como “cajas negras": no es fácil entender qué fuentes estaba considerando un LLM cuando llegó a sus conclusiones y produjo una respuesta.

Large Language Models

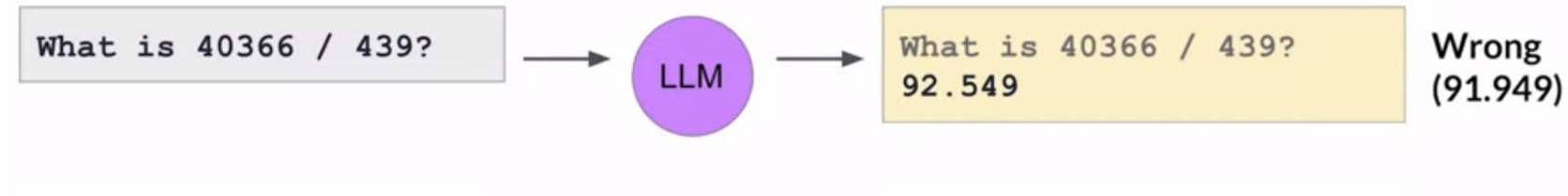
Pueden tener alucinaciones



Large Language Models



Resolución de problemas



Large Language Models

Producción costosa. Existen enfoques alternativos para aumentar el rendimiento de las aplicaciones GenAI, como la creación de un modelo de cimentación propio, el ajuste fino de un modelo existente o la realización de ingeniería rápida.

Sin embargo, RAG es el camino más rentable, fácil de implementar y de bajo riesgo para lograr un mayor rendimiento.

Solución

- Es necesario un mecanismo más eficaz y fiable = Generación Aumentada por Recuperación (RAG).

- RAG obtiene datos actualizados o específicos del contexto de una base de datos externa y los proporciona a un LLM durante la generación de la respuesta. Esto reduce la probabilidad de alucinaciones, lo que se traduce en una mejora significativa del rendimiento y la precisión de las aplicaciones.

RAG

01

Recuperación: En primer lugar, el modelo recupera información relevante o contexto de una base de datos utilizando un modelo de recuperación tradicional. La información recuperada sirve de contexto adicional para la etapa de generación.

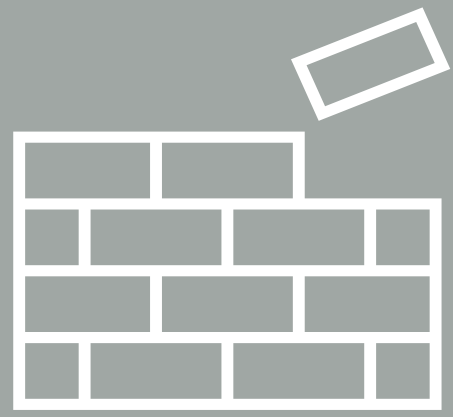
RAG

01

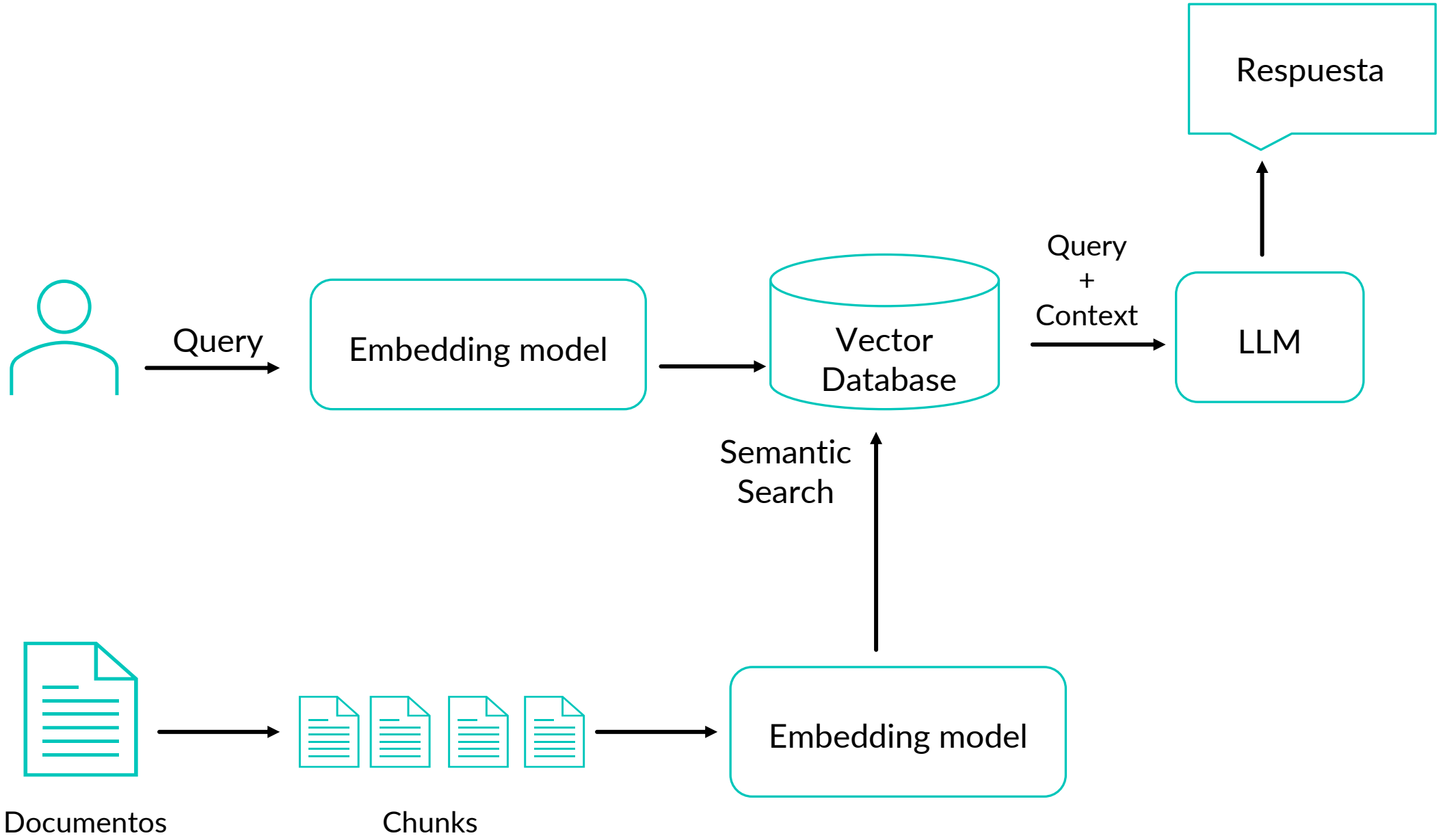
Recuperación: En primer lugar, el modelo recupera información relevante o contexto de una base de datos utilizando un modelo de recuperación tradicional (por ejemplo, BM25 o TF-IDF). La información recuperada sirve de contexto adicional para la etapa de generación.

02

Generación: Una vez recuperado el contexto relevante, se combina con la entrada original y se utiliza un modelo de lenguaje generativo (como GPT) para producir el resultado final. El modelo generativo puede ahora utilizar tanto la entrada original como el contexto recuperado para generar respuestas coherentes y contextualmente consistentes.



Arquitectura RAG

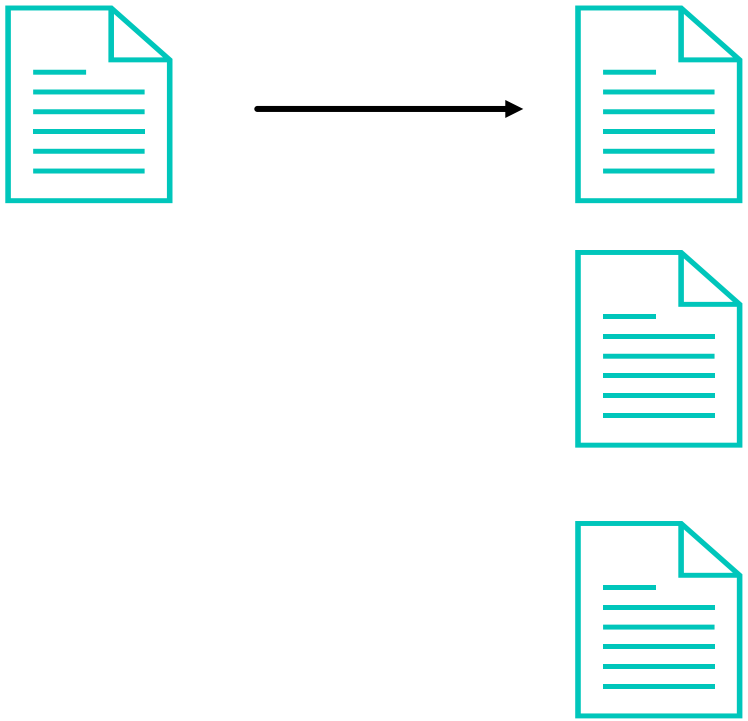

Respuesta
Query
Embedding model
Vector Database
Query + Context
LLM
Semantic Search
Embedding model
Documentos
Chunks

Documentos

¡Múltiples formatos!

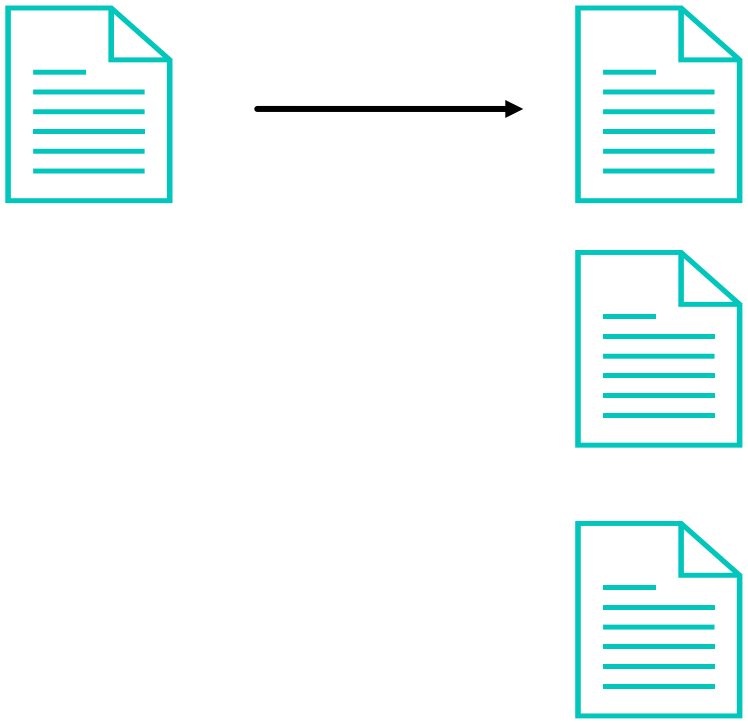
- PDFs
- Word
- Links a la web
- PPT
- Hojas de cálculo
- Txt
- Markdown
- Html
- Youtube
- Emails

Chunking



- En un LLM, la ventana de contexto es fija y limitada en tamaño.
- Debido a la limitación de la ventana de contexto en los LLM, es necesario dividir documentos largos en fragmentos más pequeños para que el modelo pueda procesarlos.
- Si un documento es demasiado extenso, el modelo no será capaz de considerar todo su contenido a la vez.
- Además, mejora la eficiencia en la búsqueda y en la recuperación de información relevante.

Chunking

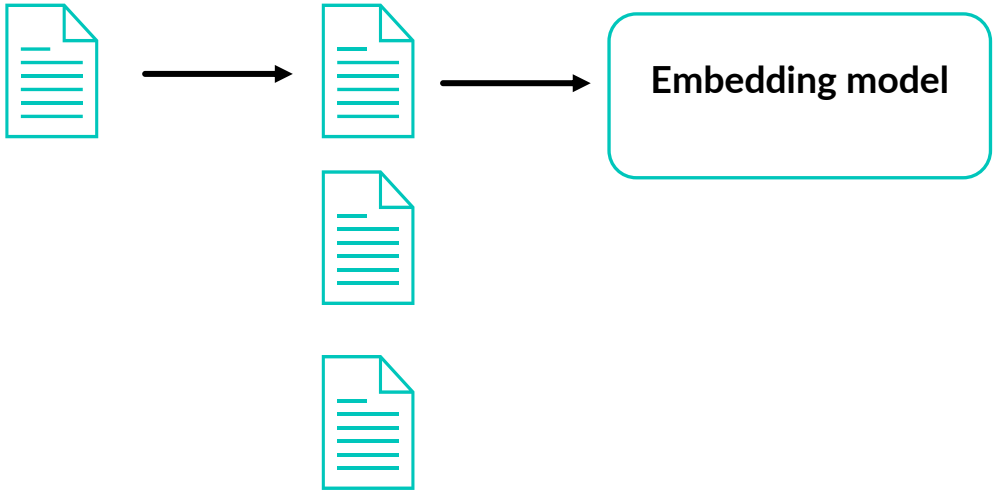


El objetivo del RAG es recuperar la información más relevante para generar respuestas basadas en la base de datos.

Problema: Los documentos aportados pueden ser muy extensos y pesados, lo cual dificulta la búsqueda de información.

Solución: Aplicamos *chunking* para dividir el gran conjunto de datos en fragmentos más pequeños, lo cual mejora la eficiencia en la búsqueda y la precisión en la recuperación de información relevante.

Embeddings

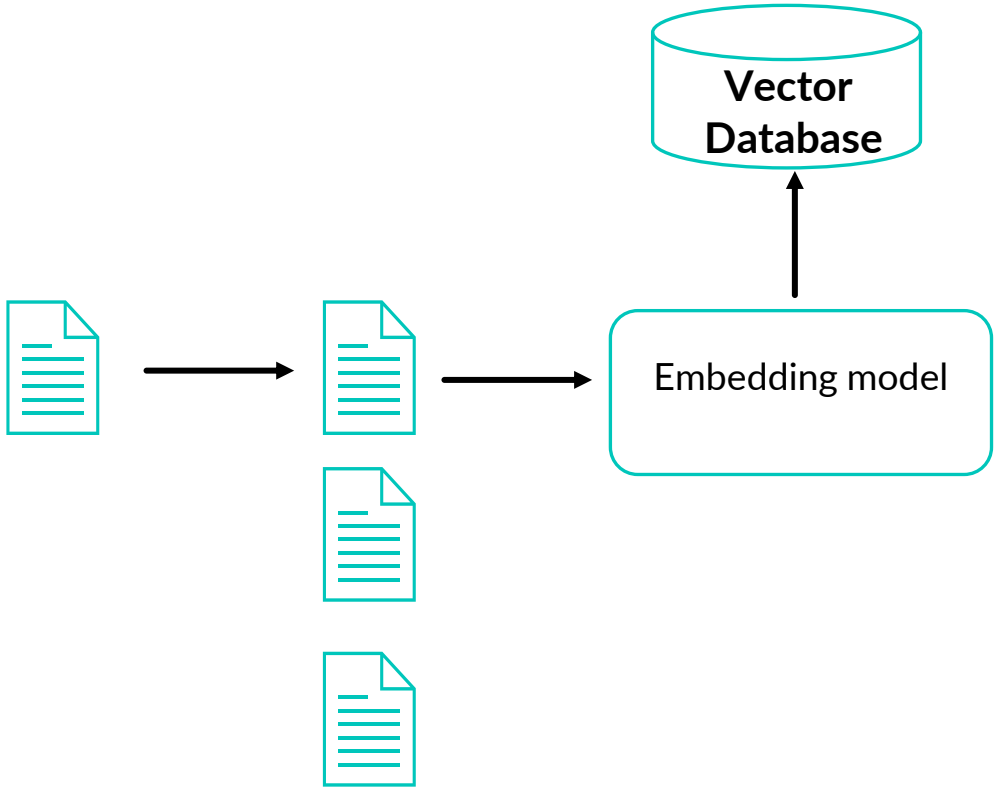


Los modelos no entienden palabras, sino números

Con los embeddings se transforman palabras o frases, en vectores numéricos que capturan el significado semántico.

- Text-embedding-ada-002
- BERT

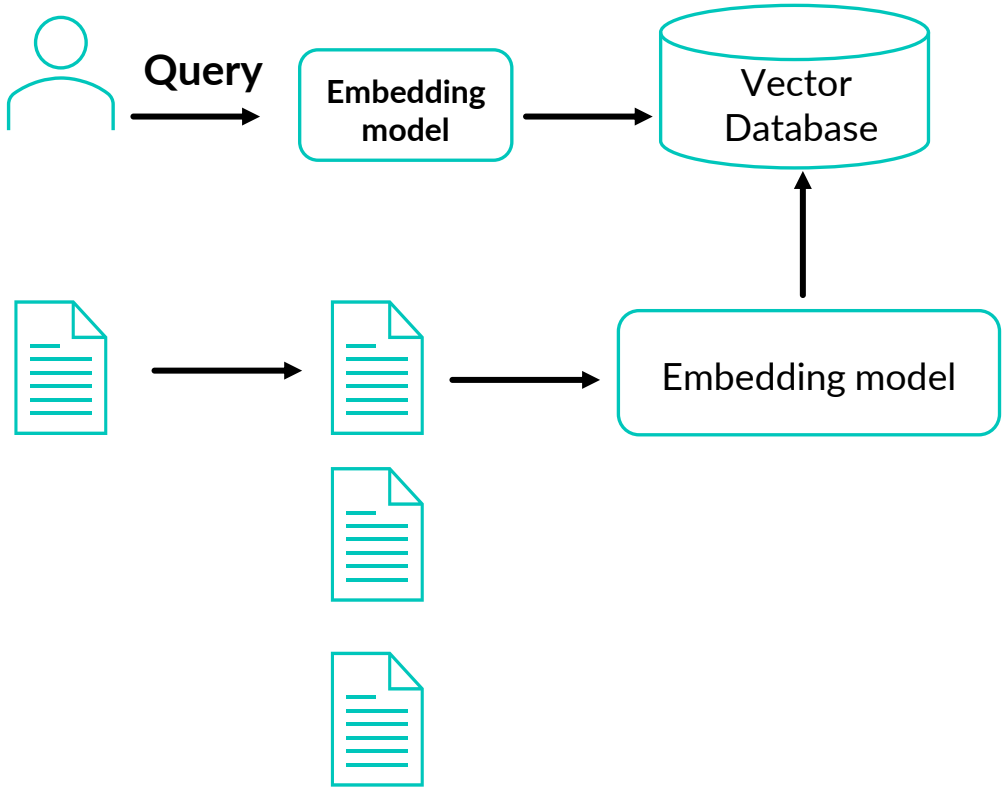
Vector Database



Tipos de vector database

- Faiss
- Chroma DB
- PineCone

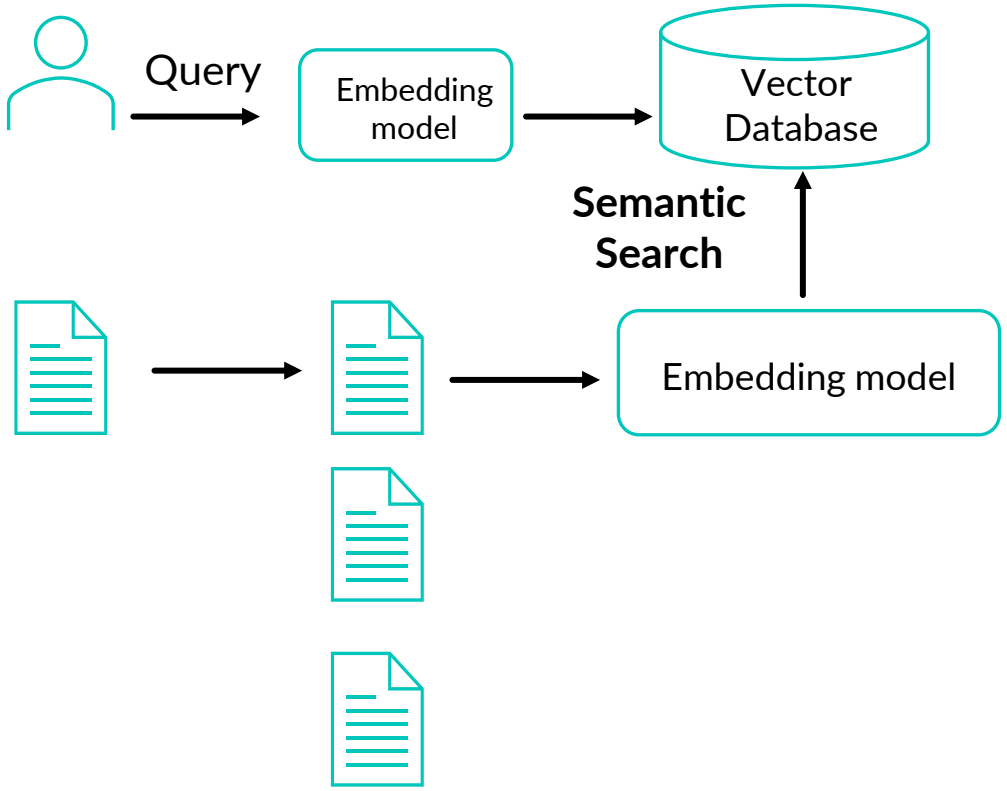
Query



Query: solicitud o pregunta que se hace a un modelo generativo de lenguaje mediante lenguaje natural.

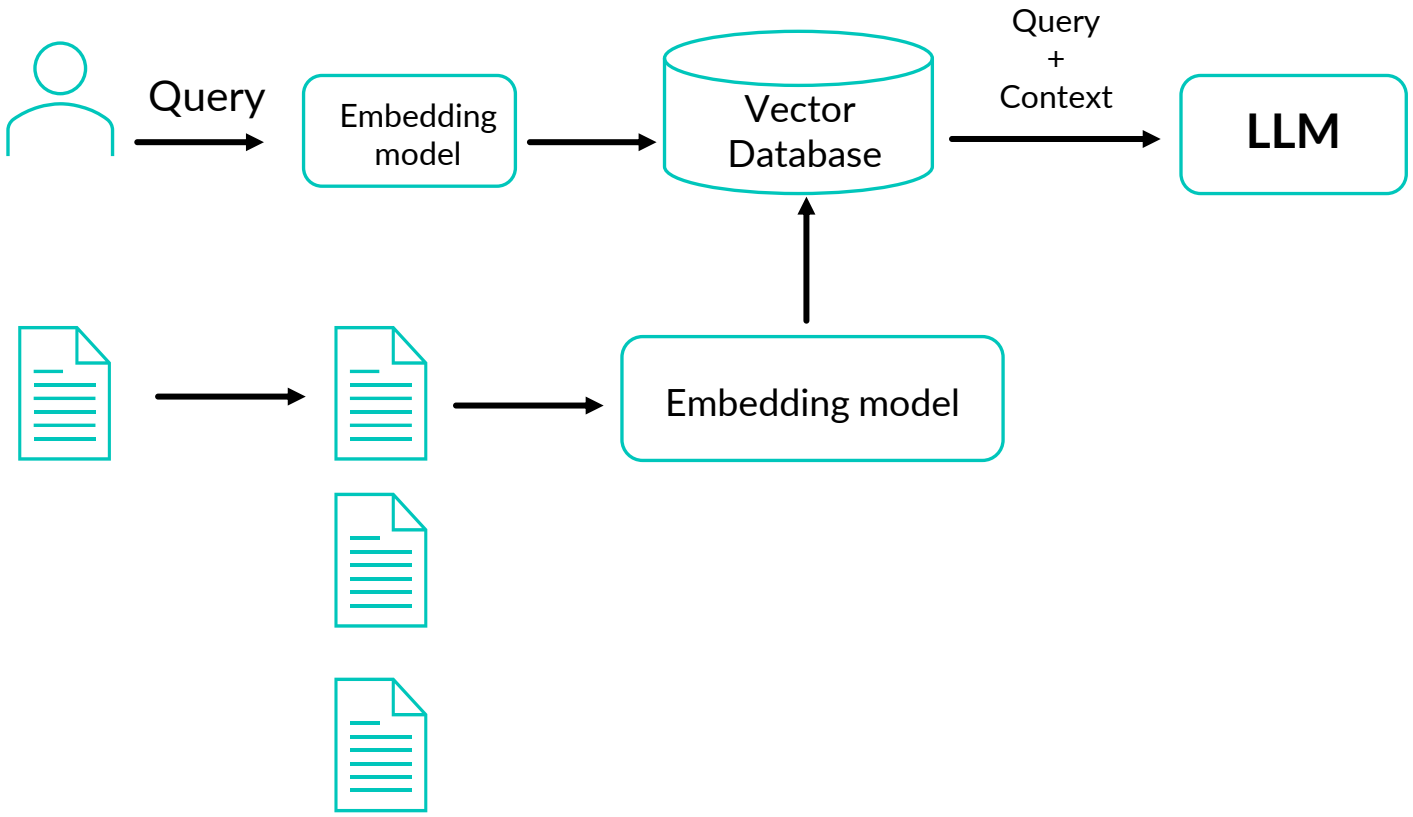
Ese lenguaje natural ha de representarse también en vectores: **vectorización.**

Semantic search



Búsqueda semántica: intenta encontrar el verdadero significado de la consulta del usuario y recuperar información relevante.

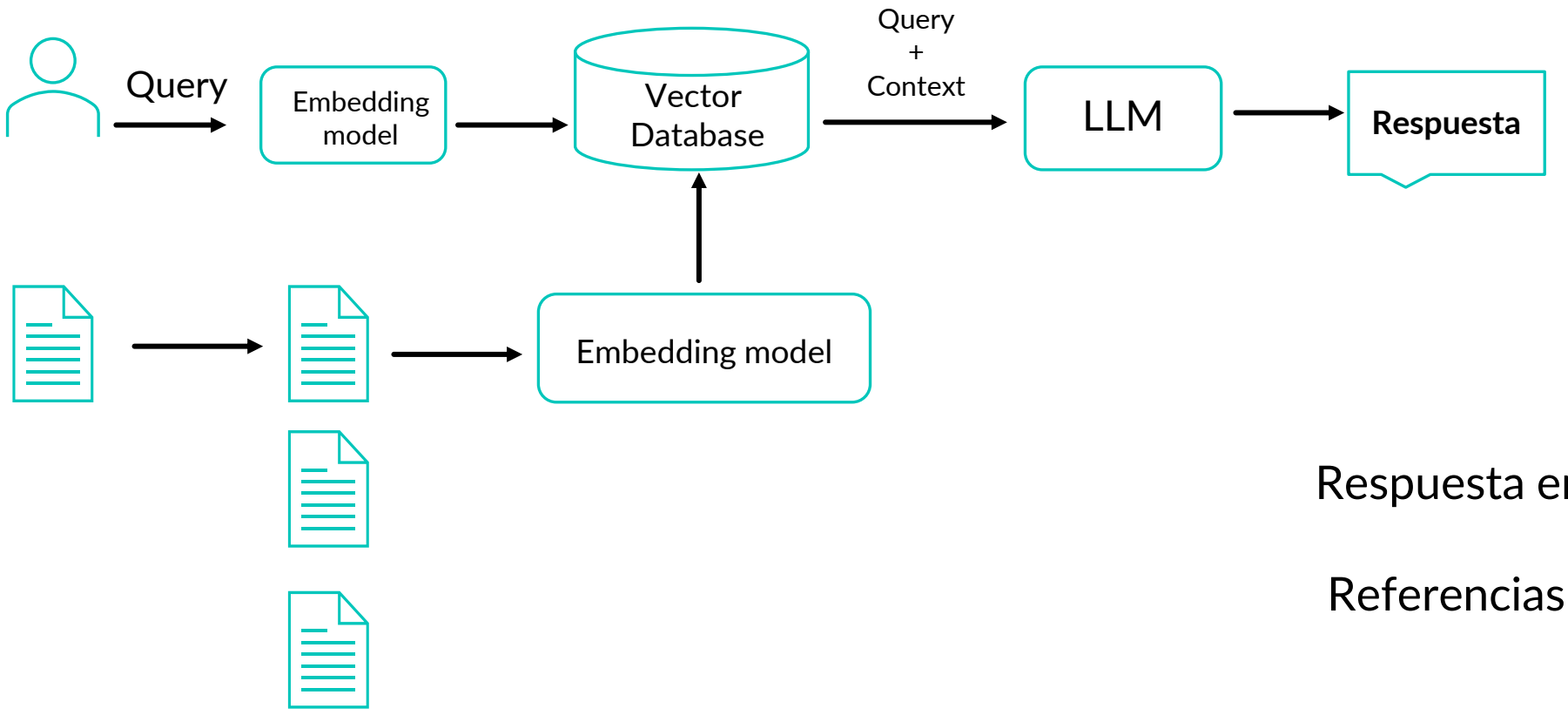
LLMs



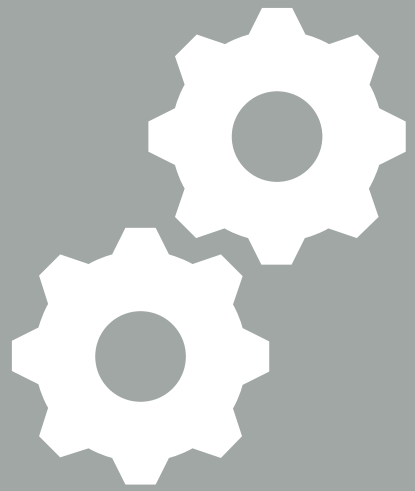
- GPT-3.5
- GPT-4

- Llama-2
- Mistral

Respuesta



Respuesta en Lenguaje Natural
+
Referencias a los documentos



Opciones de implementación

Opciones de implementación

SIN CÓDIGO

OpenAI On Your Data
- OpenAI Models
- Azure Cognitive Search
- Azure

Amazon bedrock

CON CÓDIGO

Con herramientas como **LangChain** o **LlamaIndex** + búsqueda vectorial mediante FAISS, ChromaDB, etc.

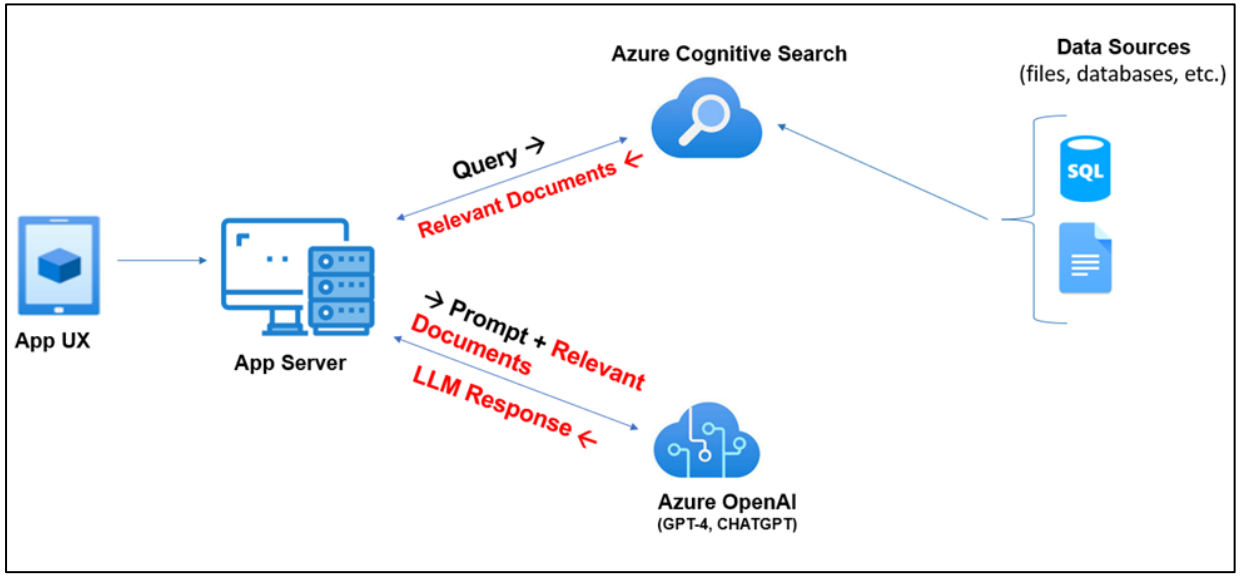
+

- Modelos privados: GPT
- Modelos open-source en Hugging-Face: LLaMA-2, Mistral, etc.

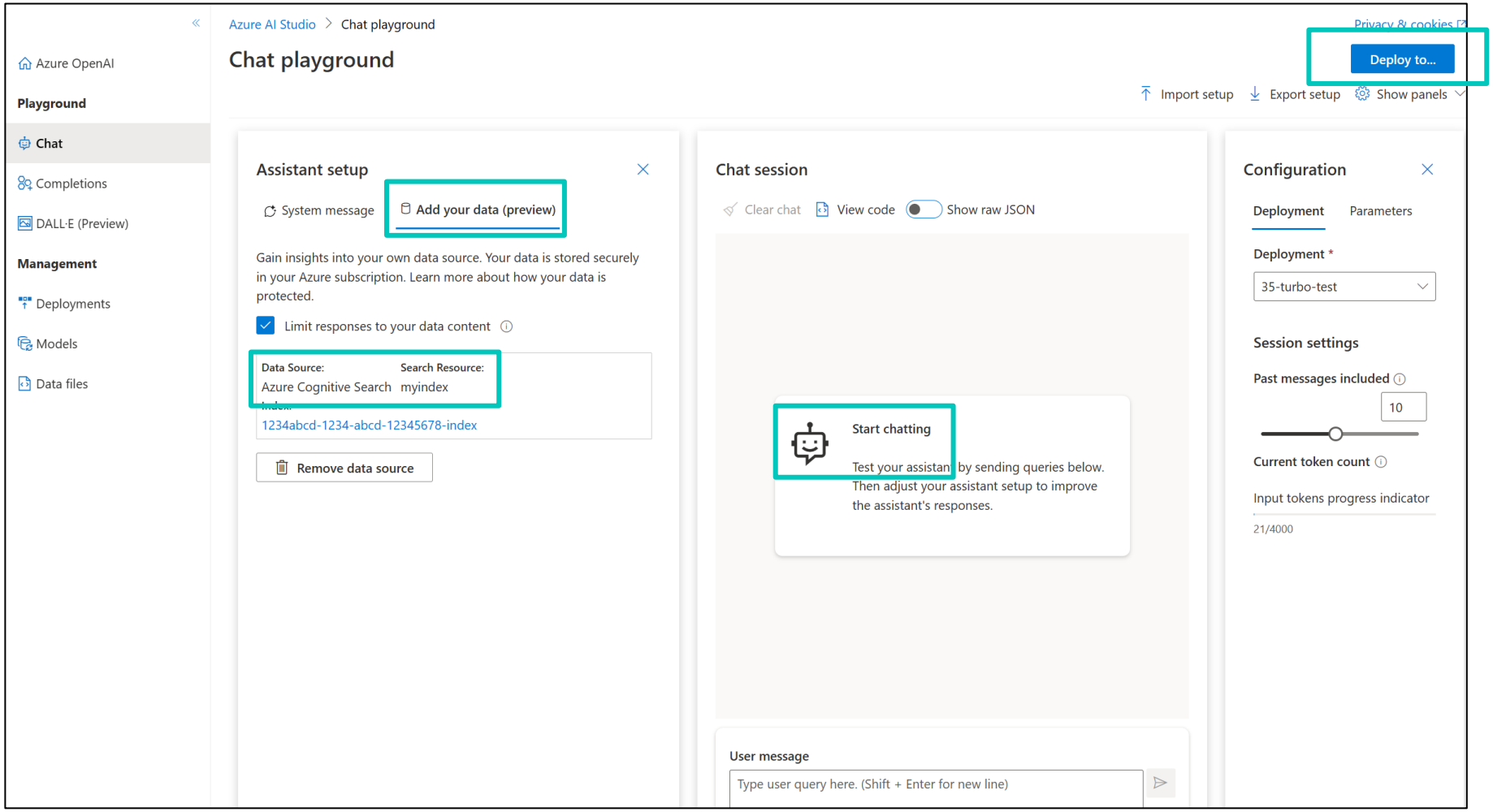
OpenAI sobre datos propios (OpenAI On your data)

#SIN CÓDIGO

Azure OpenAI On your data funciona con modelos de lenguaje como **GPT-3.5-Turbo y GPT-4 de OpenAI**, lo que les permite proporcionar respuestas en lenguaje **natural** basadas en **datos propios**.
Una de las características clave de Azure OpenAI en sus datos es su capacidad para recuperar y utilizar datos de una manera que mejore la salida del modelo.



OpenAI sobre datos propios (OpenAI On your data)

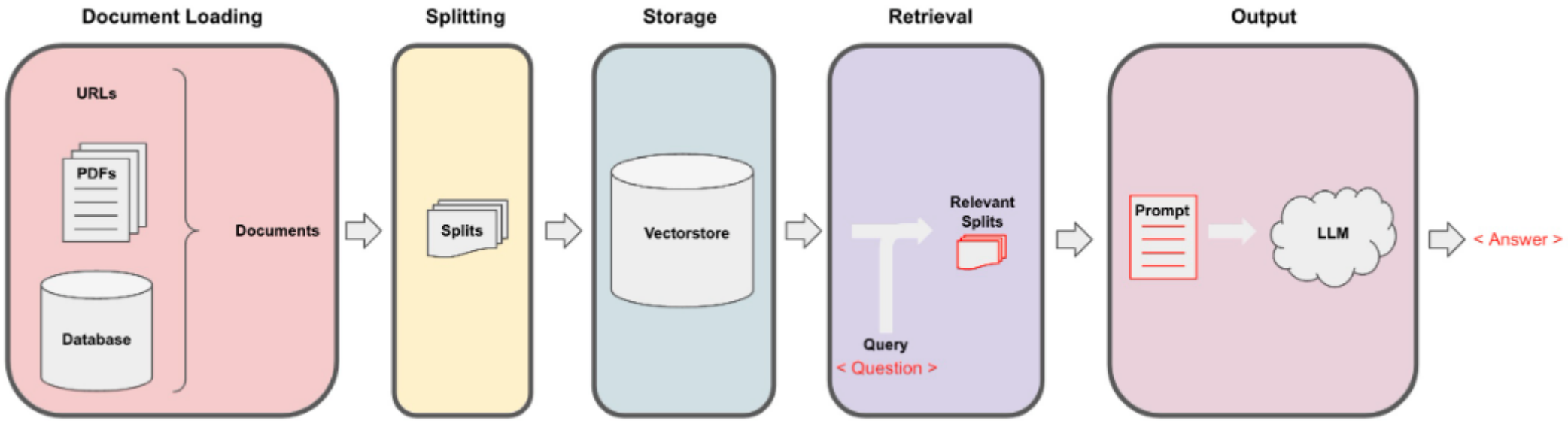


LANGCHAIN

#CON CÓDIGO

LangChain es un marco para desarrollar aplicaciones basadas en modelos de lenguaje. Permite conectar modelos de lenguaje con otras fuentes de datos.

https://python.langchain.com/docs/get_started/introduction



CSV

File Directory

HTML

JSON

Markdown

PDF

Llama Index

#CON CÓDIGO

LlamaIndex es un marco de datos simple y flexible para conectar fuentes de datos personalizadas a modelos de lenguaje grandes.

Conectores

LlamaIndex

Data Ingestion

Connect your existing data sources and data formats (API's, PDF's, documents, SQL, etc.) to use with a large language model application.

Data Indexing

Store and index your data for different use cases. Integrate with downstream vector store and database providers.

Query Interface

LlamaIndex provides a query interface that accepts any input prompt over your data and returns a knowledge-augmented response.

Conectores

https://gpt-index.readthedocs.io/en/latest/

Unstructured Data

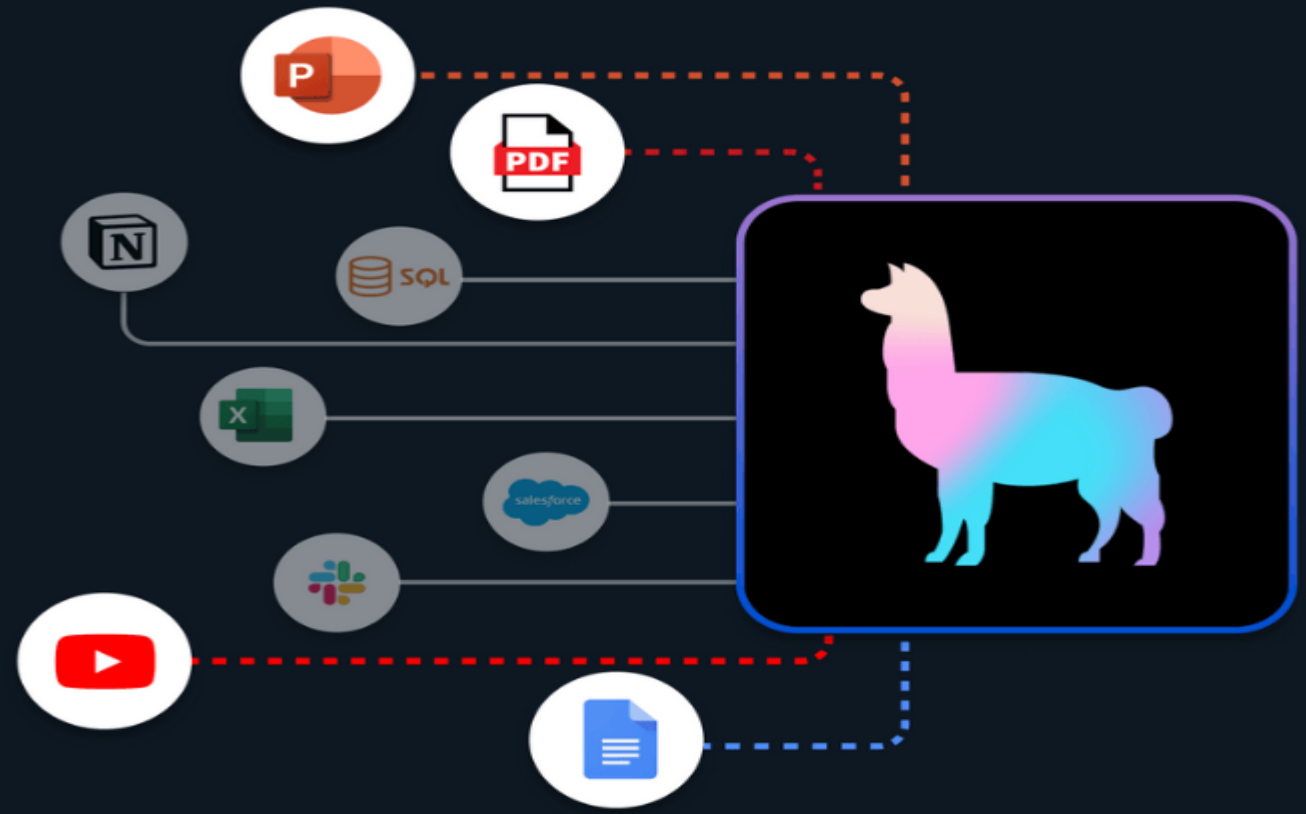
Connect unstructured sources such as documents, raw text files, PDF's, videos, images, etc.

Structured Data

Easily integrate structured data sources from Excel, SQL, etc.

Semi-Structured

Connect semi-structured data from API's like Slack, Salesforce, Notion, etc.



```
[19] from langchain.document_loaders import WebBaseLoader

     web_link = ["http://jalammar.github.io/illustrated-transformer/"]

     loader = WebBaseLoader(web_link)
     documents = loader.load()
```

```
[26] chat_history = [(query, result["answer"])]

     query = "What does “it” mean in the example sentence for the explanation of Self-Attention at a High Level?"
     result = chain({"question": query, "chat_history": chat_history})

     print(result['answer'])
```

```
In the explanation of Self-Attention at a High Level, "it" refers to the word "animal" in the input sentence.
```

Self-Attention at a High Level

Don't be fooled by me throwing around the word "self-attention" like it's a concept everyone should be familiar with. I had personally never came across the concept until reading the Attention is All You Need paper. Let us distill how it works.

Say the following sentence is an input sentence we want to translate:

"The animal didn't cross the street because it was too tired"

What does "it" in this sentence refer to? Is it referring to the street or to the animal? It's a simple question to a human, but not as simple to an algorithm.

When the model is processing the word "it", self-attention allows it to associate "it" with "animal".

```
print(result['source_documents'])
```

```
ata={'source': 'http://jalammar.github.io/illustrated-transformer/', 'title': 'The Illustrated Transformer – Jay Alammar – Visualizing machine learning
```

Aplicaciones

- Chatbots: RAG puede mejorar las respuestas de los chatbots recuperando respuestas relevantes de una base de conocimientos antes de generar una respuesta.

- Respuesta a preguntas: RAG puede mejorar los sistemas de respuesta a preguntas combinando la recuperación con un modelo generativo para generar respuestas precisas e informativas.

- Resumen de documentos: Las RAG pueden aplicarse para generar resúmenes contextualmente más relevantes recuperando las frases pertinentes de los documentos fuente.

LAB