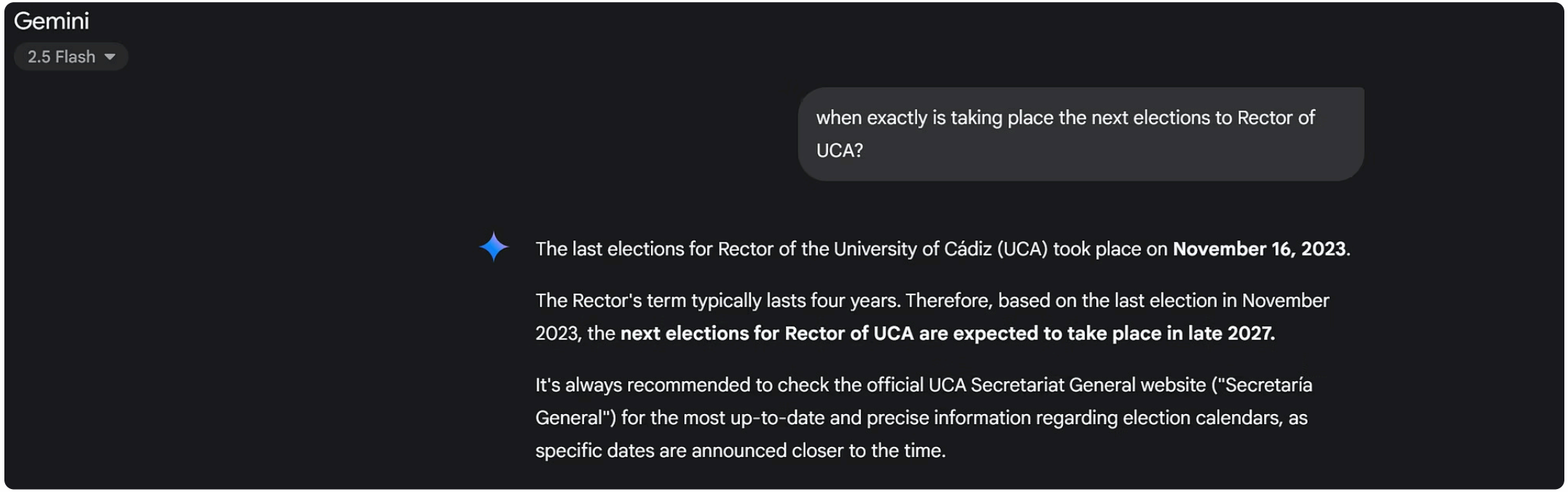
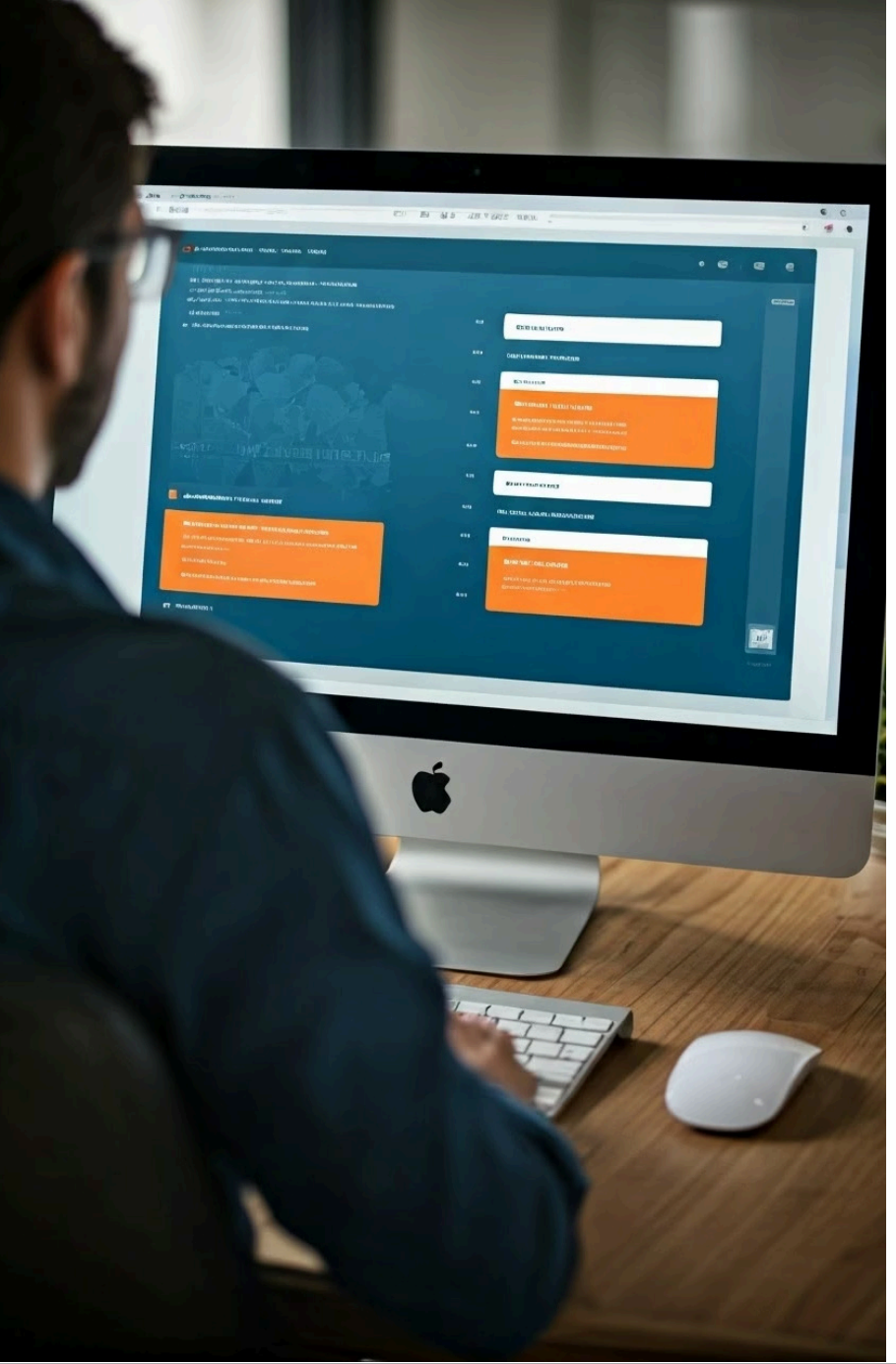
Retrieval Augmented Generation (RAG)

Iván Ruiz / Andrés Muñoz

Contents


Gemini
2.5 Flash
when exactly is taking place the next elections to Rector of UCA?
The last elections for Rector of the University of Cádiz (UCA) took place on **November 16, 2023**.
The Rector's term typically lasts four years. Therefore, based on the last election in November 2023, the **next elections for Rector of UCA are expected to take place in late 2027.**
It's always recommended to check the official UCA Secretariat General website ("Secretaría General") for the most up-to-date and precise information regarding election calendars, as specific dates are announced closer to the time.

Introduction



Information Quality: Issues

Outdated Information

LLMs have limited knowledge of the world. They cannot access information created after their training.

Inaccessible Information

Public web information that was not captured during the data extraction process prior to model generation is also inaccessible to LLMs.

Private Information

LLMs cannot access protected information from organisations, limiting their ability to provide complete responses.

Hallucinations

LLMs can generate incorrect, inaccurate or completely fabricated information that appears plausible.

Information Quality: Remediation

Fine-tuning

Fine-tuning involves **adjusting** the **parameters** of a pre-trained language model to adapt it to a specific dataset.

This method is ideal for improving the model's accuracy on a particular task, such as generating text with a **specific style or tone**.

Retrieval-Augmented Generation

A more **efficient** technique than fine-tuning to reduce hallucinations in LLMs.

It consists of augmenting the input given to the LLM, incorporating the entire knowledge base it needs to provide a coherent response.

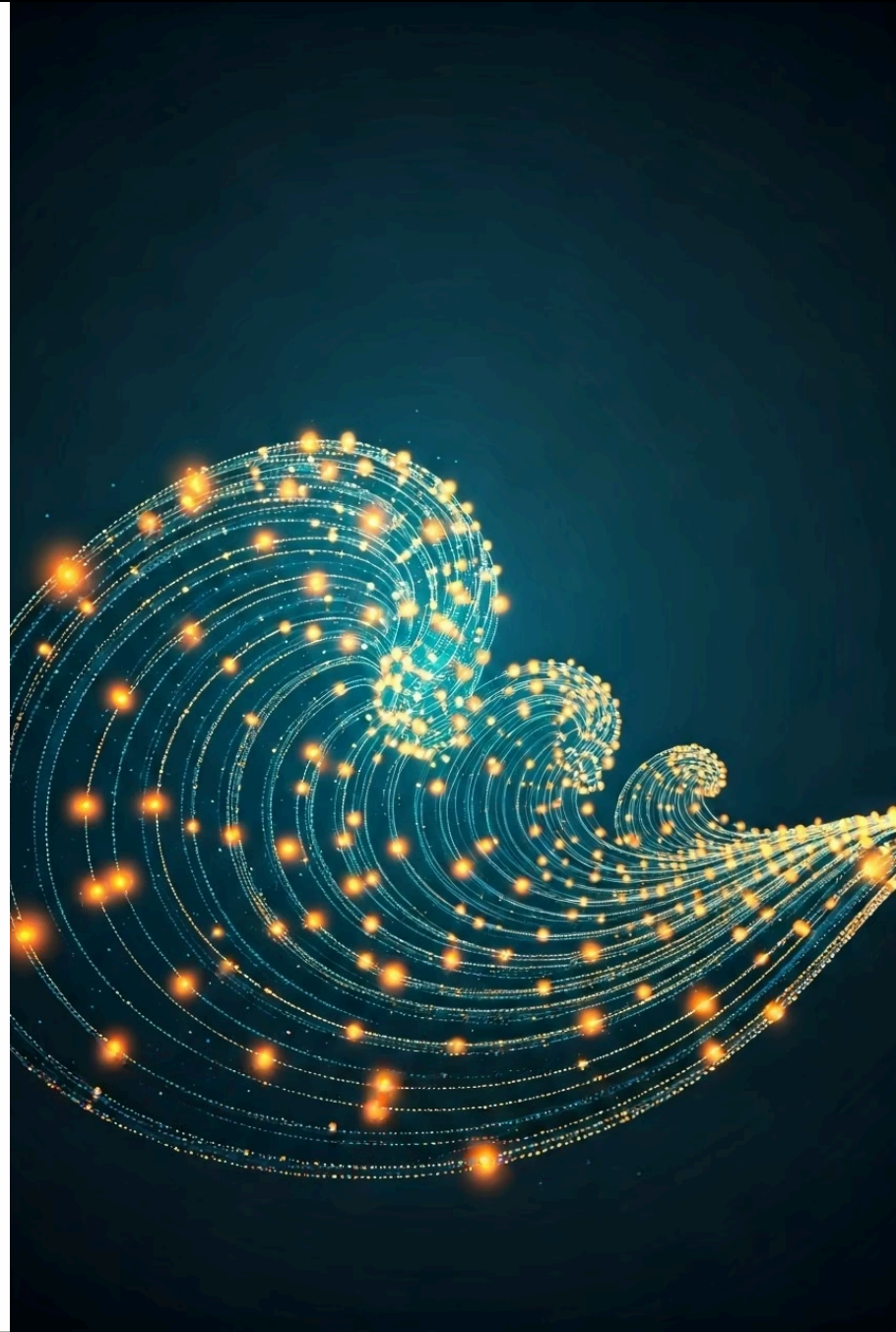
Retrieval-Augmented Generation (RAG)

Improved quality

Retrieval-augmented text generation seeks to improve the quality of language model responses.

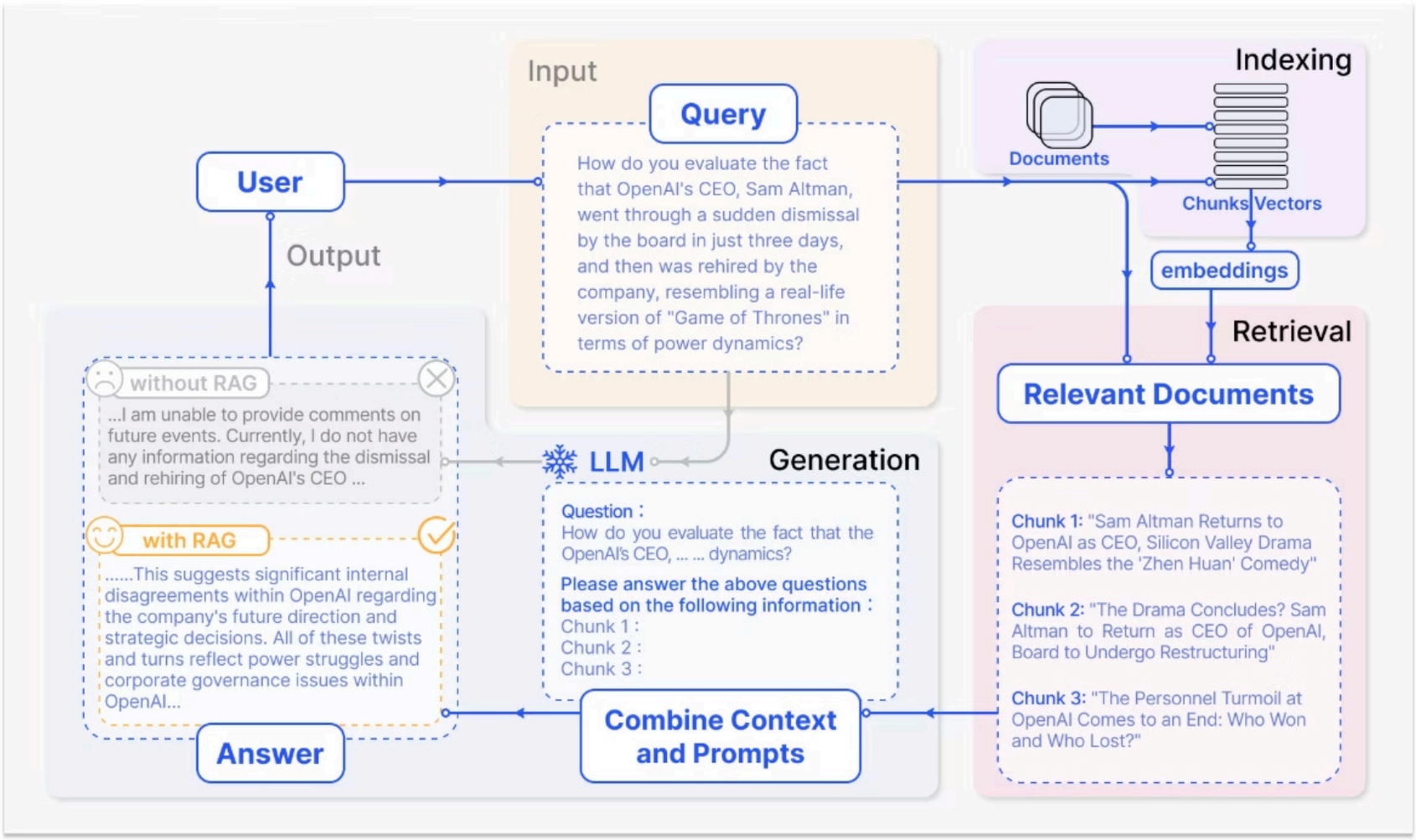
Knowledge base

The language model connects to a knowledge base, retrieving relevant information to generate enriched responses.




Naive RAG

Retrieval-Augmented Generation



Yunfan Gao et al. Retrieval-Augmented Generation for Large Language Models: A Survey, 2023

Stages of the RAG

Indexing

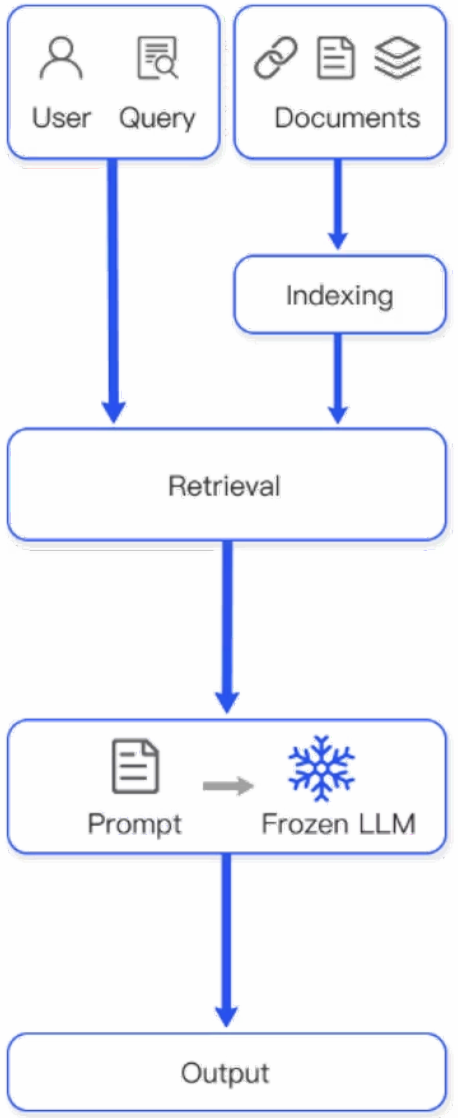
Offline pre-processing of information (documents) to enable the following stages

Retrieval

Retrieval of information that occurs online when the user launches the query to the LLM

Generation

Response synthesised by the LLM as a response to the query and the retrieved texts



Naive RAG

Information Indexing

Information Indexing

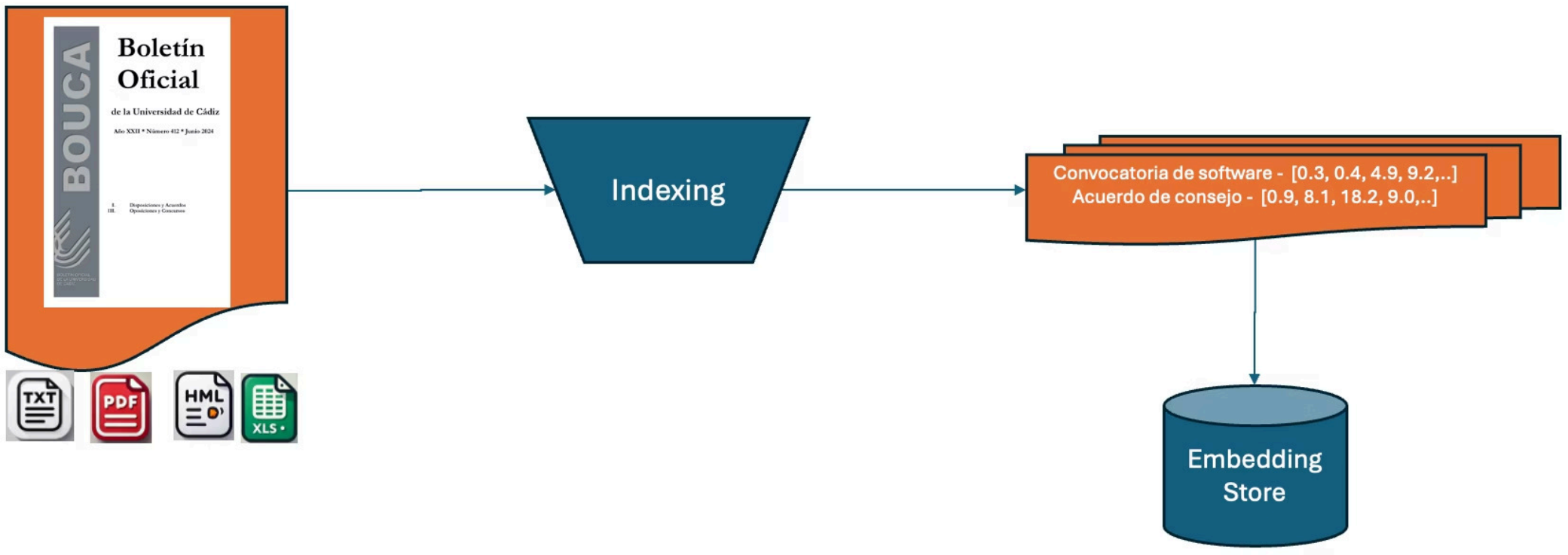
Transformation

Converting information into a format that enables efficient searches.

Importance

This process is fundamental for subsequent retrieval and text generation.

Indexing (example)

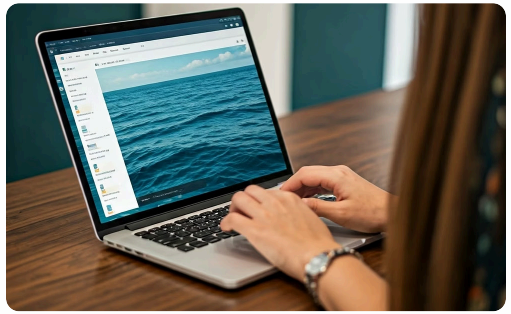


ETL

The indexing process can be divided into successive stages that are integrated into an ETL (Extract, Transform, Load) pipeline.

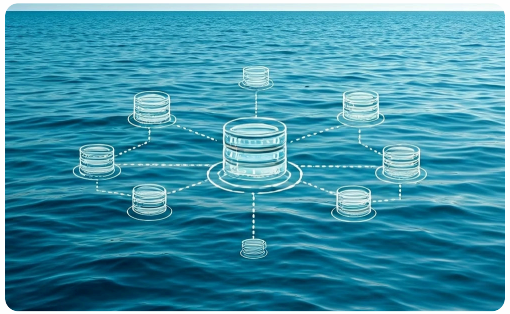
1. Documents are **loaded** from external sources.
2. Documents are **analysed** using some parser.
3. Documents are **transformed** to clean or enrich them with metadata.
4. Documents are **divided** into smaller segments.
5. Numerical vectors (**embeddings**) are generated for each segment.
6. The vectors are **stored** in a dedicated database.

Loading



File systems

Information collected in files stored in local file systems or remote systems (e.g. NextCloud or GDrive)

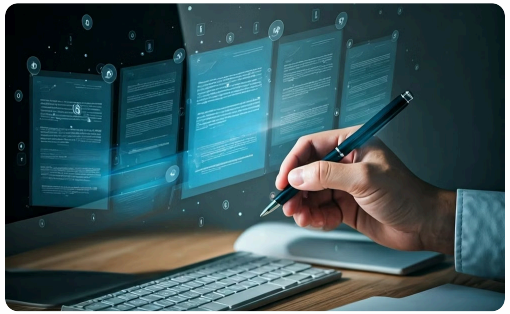


Database systems

Information collected in tables in relational databases (e.g. Oracle) or in other formats of non-relational databases (e.g. MongoDB documents)

Storage services

Information collected in open source object storage systems (e.g. MinIO) or proprietary (e.g. AWS S3 or Google Cloud Storage)



Document managers

Open source document management systems (e.g. Alfresco) or proprietary (e.g. SharePoint)

Parsing

Documents must be analysed and their data extracted with the help of an appropriate parser.

Unstructured information

Files with layout and images (e.g. DOC, PPT or PDF)

Semi-structured information

Files with some form of tagging (e.g. JSON, XML or HTML)

Structured information

Databases accessible with SQL, spreadsheets or CSV files

Transforming: Cleaning

Removing irrelevant information

Irrelevant headers or footers, watermarks or digital signatures should be removed to optimise the content.

Extracting visual elements

Tables, images or graphs should be properly processed to generate text.

Normalising the text

Extra spaces, strange characters are removed and the format is normalised for coherent analysis.

Removing HTML markup

To have the text ready for indexing, HTML tags are removed.

Transforming: enriching

Enriching with metadata is essential for:

- Providing additional context to the LLM.
- Optimising searches with filtering.
- Re-indexing information after updates to the source documents.

Common Metadata

URL

Access the source of the document and verify its content.

File name and page number

Uniquely identify the document and/or access a specific section.

Retrieval date

Determine the timeliness of the document.

Category

Facilitate the classification and search of the document by the LLM.

Authorship

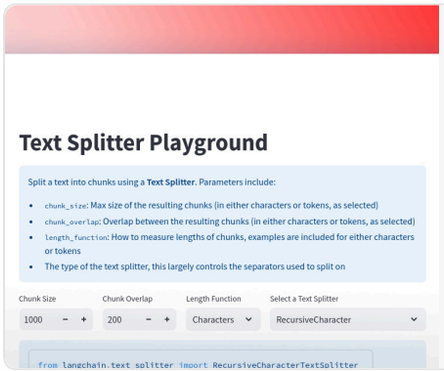
Evaluate the reliability of the content.

Identifiers at source

Efficiently locate information records in the source

Chunking (splitting)

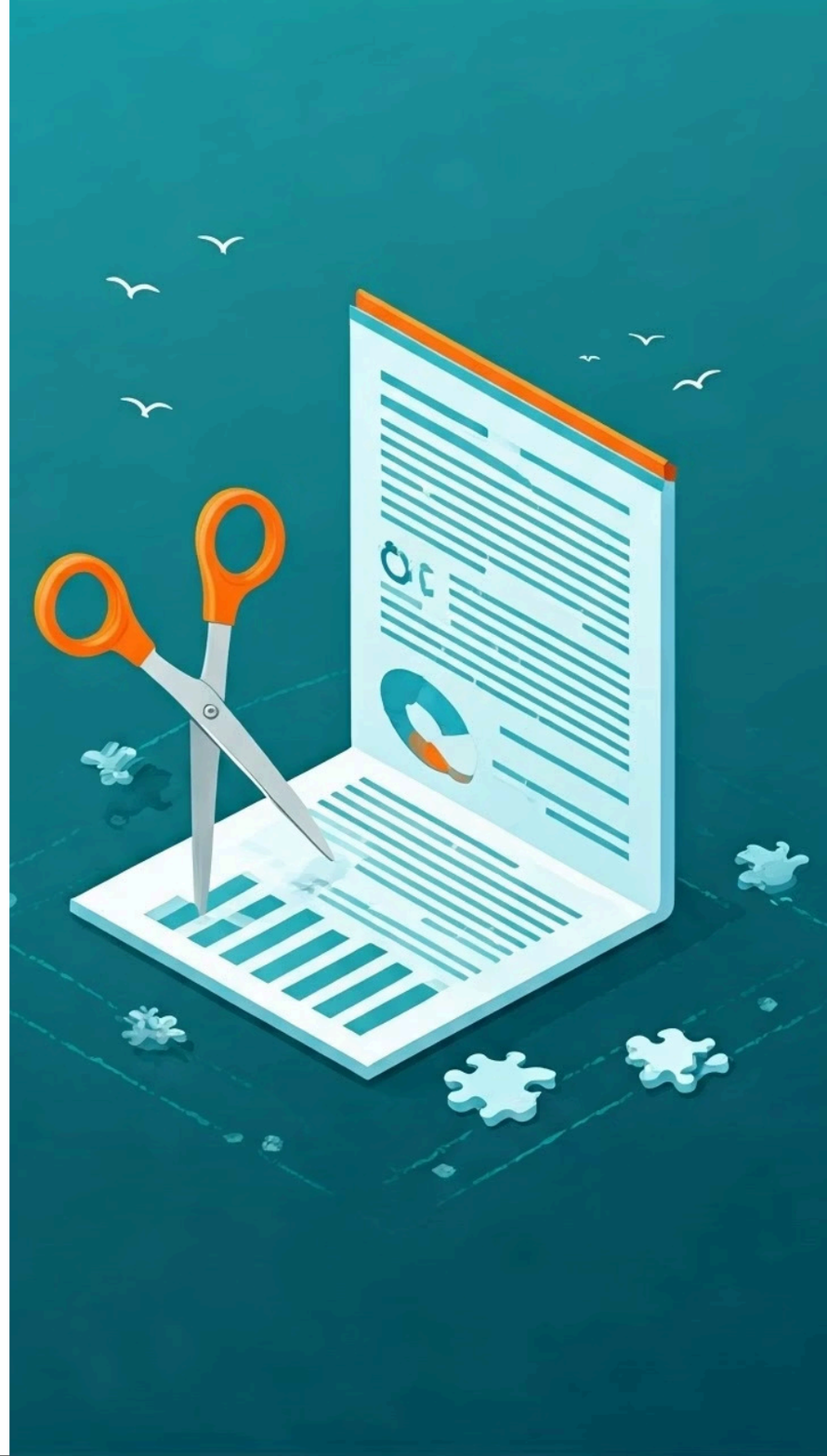
- **Fixed size:** division into fixed-length segments of characters or tokens (approximately 300).
- **Recursive:** division into paragraphs, lines, spaces and, finally, into words.
- **Document-based:** division by structural delimiters of the document. Examples:
  - **#** in Markdown
  - **<section>** in HTML
  - **\section** in LATEX
  - **class** in Python



Streamlit

splitter

ui.png



Indexing: embedding creation

Semantic Representation

Embeddings capture the meaning of text, representing it as numerical vectors of a certain length (1536 floats).

Embedding Models

Specific machine learning models are used to generate the embeddings.

Google's Word2Vec

Google's Word2Vec was one of the first embedding models.

Types of Tasks

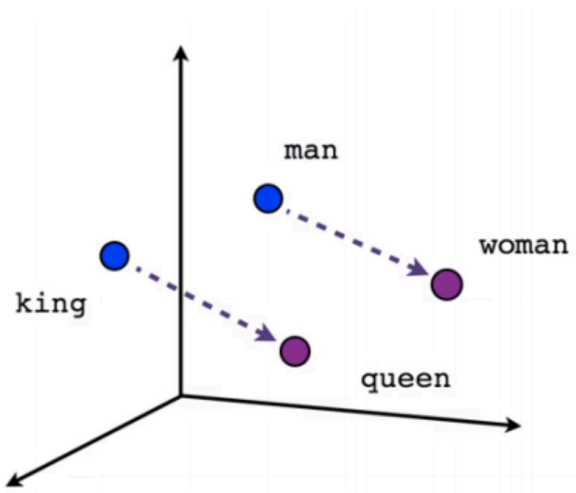
In addition to information retrieval, embeddings can be used to perform classification or clustering tasks.

Local Execution

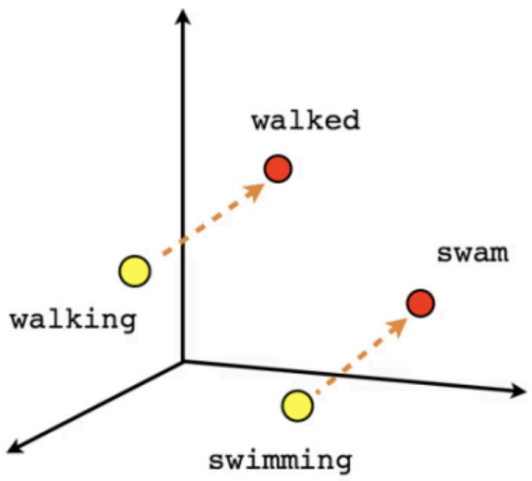
There are embedding models that can be executed locally (ONNX Runtime or Jllama) in the same execution process.

Remote Execution

We can use HTTP APIs to access embedding models, such as OpenAI's *text-embedding-3-small*.



Male-Female



Verb tense

Ranking of embedding models

Choosing the most appropriate embedding model is also an important decision...

huggingface

MTEB Leaderboard – a Hugging Face Space by mteb

Discover amazing ML apps made by the community

Indexing: storing embeddings

Storage

The generated embeddings must be stored along with the corresponding text segments to enable information retrieval.

Vector stores

Vector stores are specialised databases for storing high-dimensional vectors, such as embeddings.

Similarity search

These systems allow for searches based on the similarity between vectors.

Integration

Vector stores offer APIs for integration with different programming languages, which facilitates their use in AI applications.

Comparison of Vector Databases

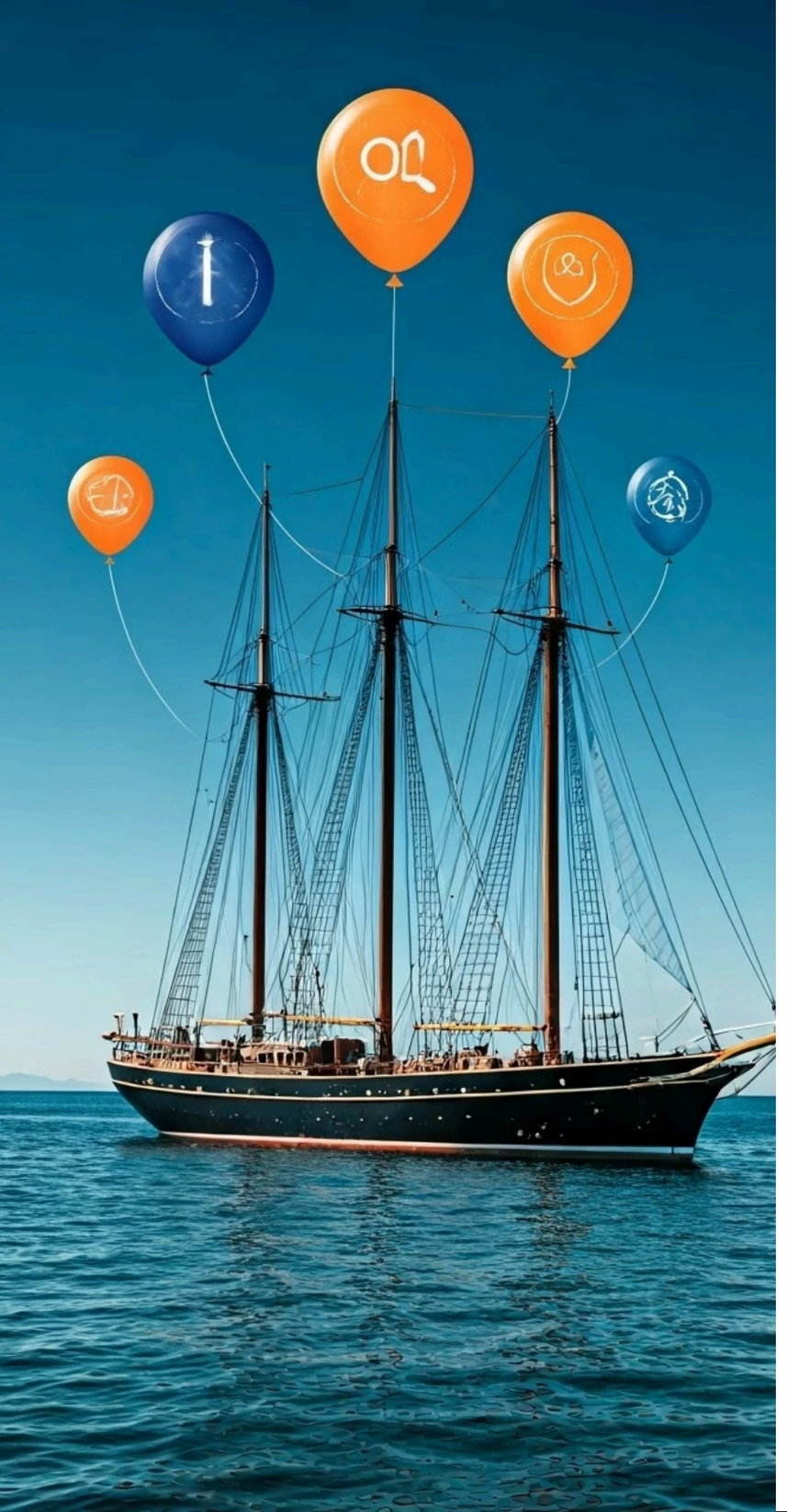
And choosing the right database...

superlinked.com

Vector DB Comparison

Vector DB Comparison is a free and open source tool from VectorHub to compare vector databases.

Information Retrieval



Data Sources

Vector Databases

Perform semantic searches with filters.

Web

Search engines like Google, Bing or DuckDuckGo.

Relational Databases

Access data collected in engines (Oracle, PostgreSQL) through SQL queries.

NoSQL Databases

Queries in NoSQL databases, such as Neo4J, using Cypher language.

Search Servers

Search engines with storage capabilities, such as Solr or ElasticSearch.

Web Services

HTTP APIs to obtain up-to-date information.

Information Retrieval

Information retrieval is a fundamental part of the QA process, as it allows access to relevant data for generating information.

The quality of the retrieved information directly affects the accuracy and quality of the generated text.

SQL Search

- With the SQL language, you can perform searches in string fields using operators like LIKE or REGEXP_LIKE
- Exact or partial matches of keywords within the documents are searched, according to the defined patterns.

```
SELECT * FROM products WHERE description LIKE '%camera%';
```

```
SELECT * FROM products WHERE REGEXP_LIKE(description, 'camera|phone');
```

```
SELECT * FROM products WHERE CONTAINS(description, 'camera') > 0;
```

Full-text search

Full-text searches look for the occurrence of keywords in documents.

These systems assign a score to documents based on the frequency and relevance (TF-IDF algorithm) of the keywords in the query.

The score is based on factors such as frequency of appearance, creation date, proximity of keywords to each other, geolocation, etc.

They have limitations when working with natural language queries due to the difficulty of identifying synonyms, paraphrases or context.

elasticsearch

Semantic search in vector stores

Embeddings

Documents are converted into numerical vectors that represent their meaning.

Vector Stores

Allow for fast and efficient searches in the vector space, based on the similarity of vectors.

Similarity Distance

Documents are ranked based on the similarity of their embeddings to the embedding of the input query, using angular distance (cosine similarity).

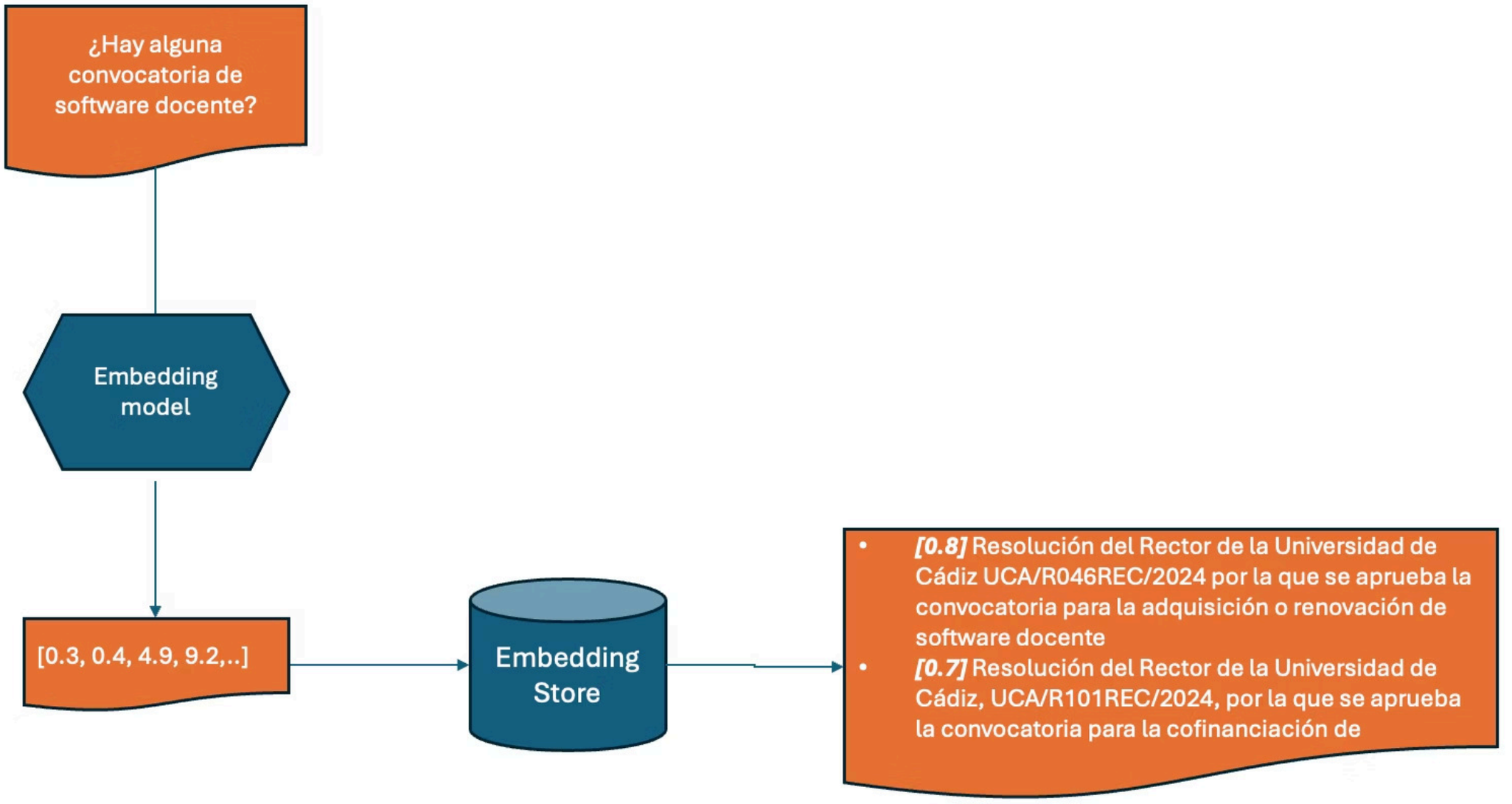
Different embedding models produce different outputs

Do not compare embeddings generated with a particular model to embeddings generated by others.

Obtaining Results

The k most similar chunks to the query are obtained, providing a set of relevant information.

Retrieval (example)



Ex 1. Creation of a semantic search engine on a vector database

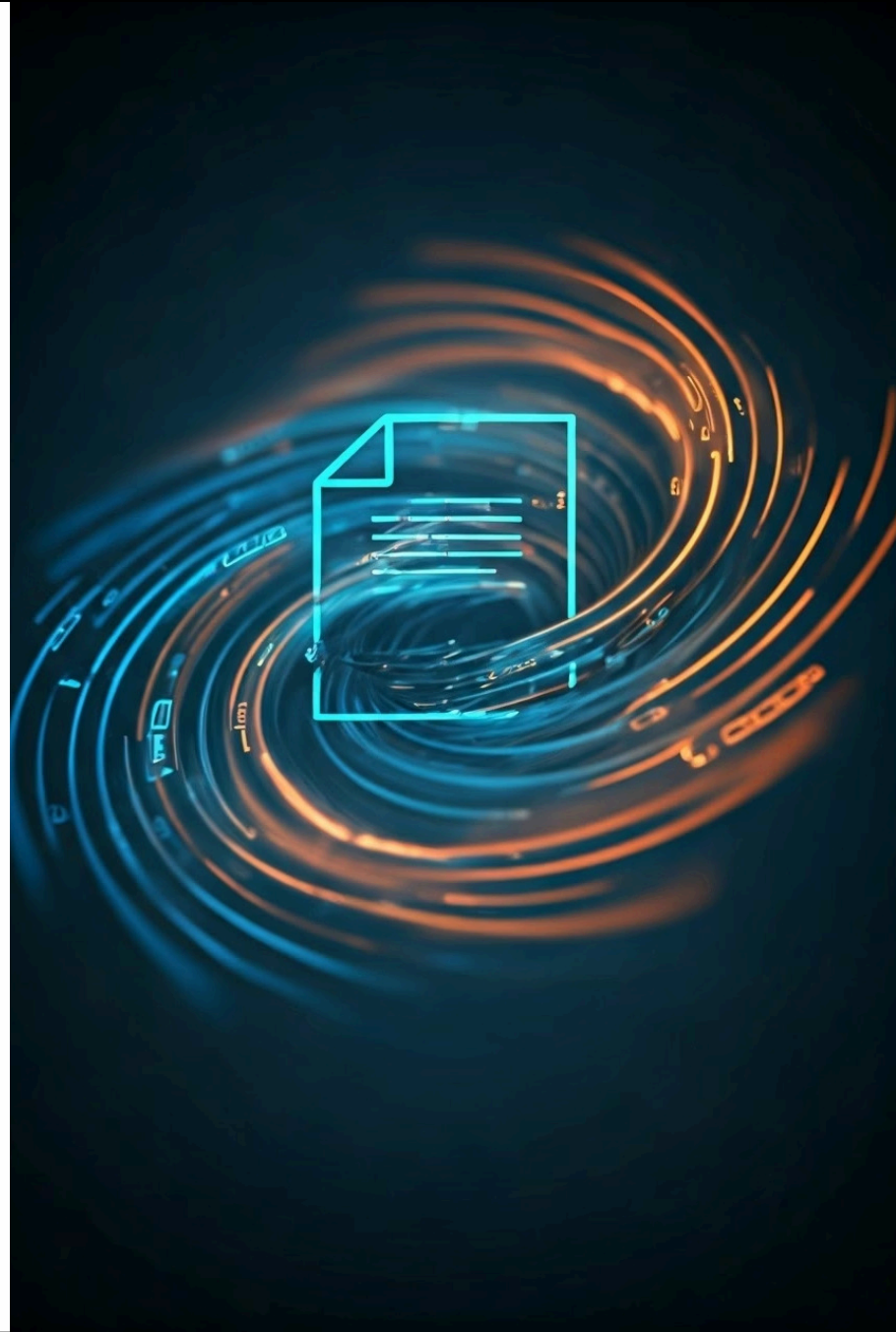
Semantic searches combined with filters

User Query

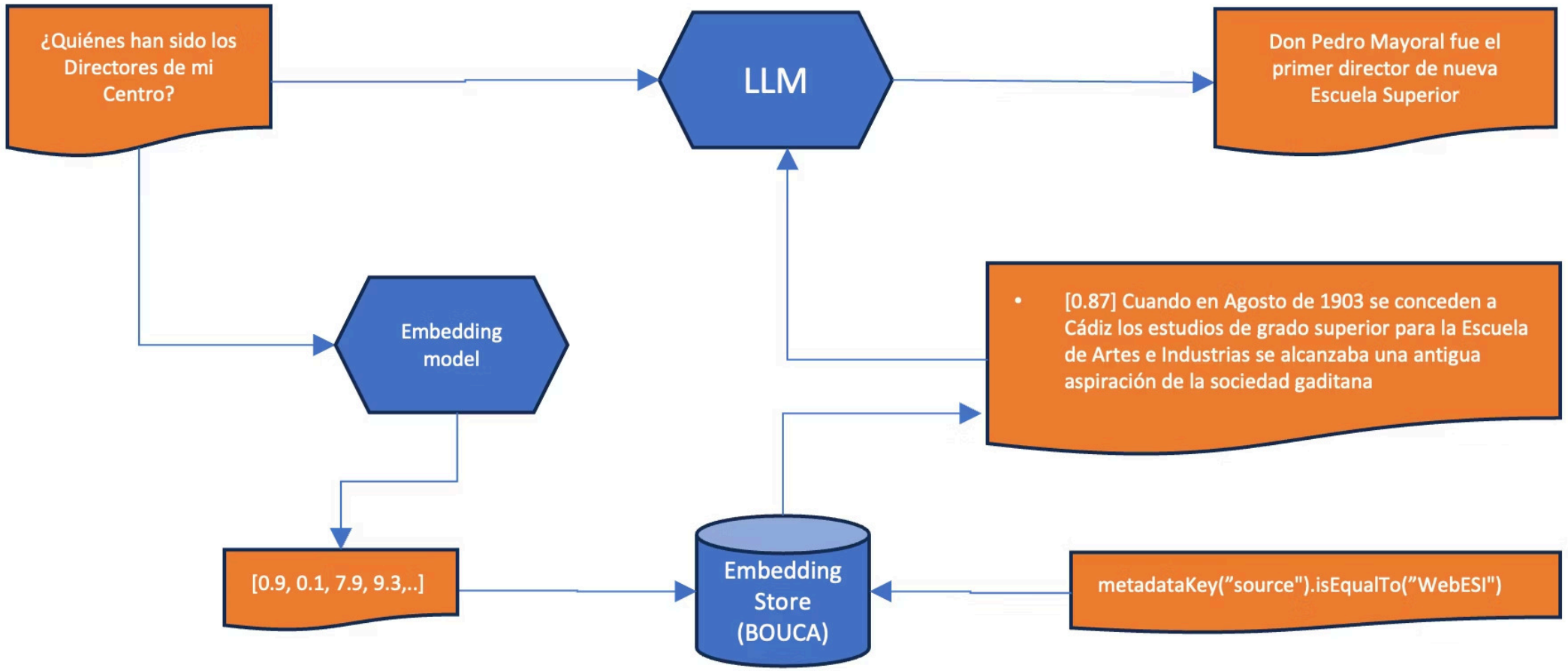
The metadata accompanying the text segments and their embeddings enable the possibility of applying additional filters

Metadata Filters

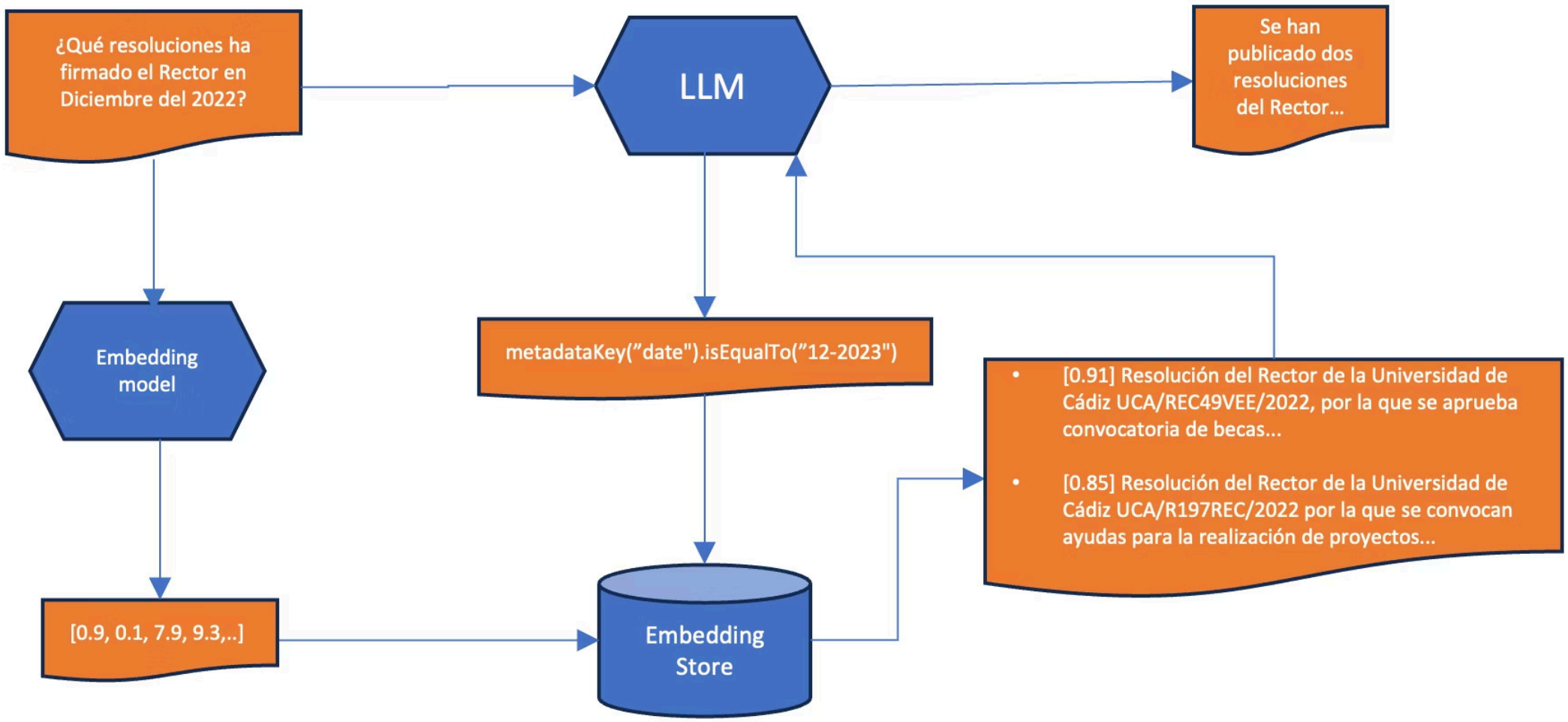
Filters can be defined statically or dynamically (e.g. according to user permissions) or generated by an LLM



Metadata Filters

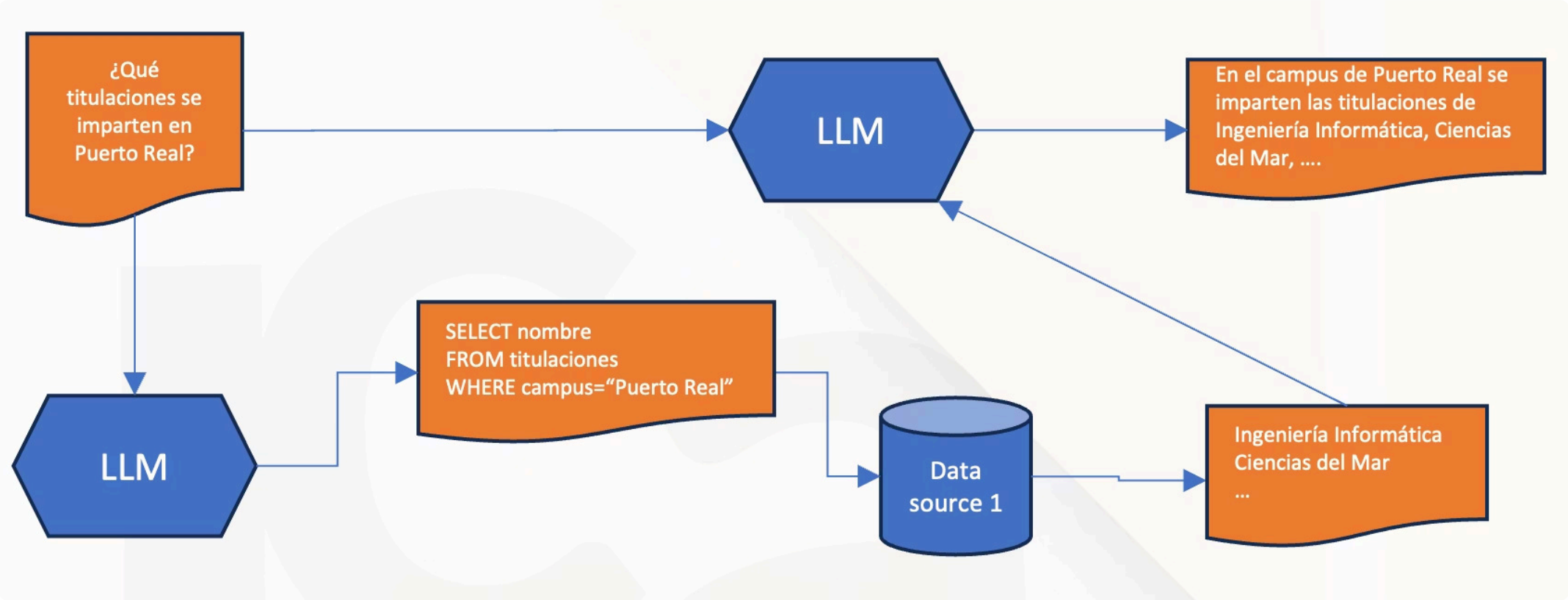


Generated Filters (Self-Querying)



Semantic Search in Relational Databases

The user makes a natural language query that an LLM translates into a valid SQL statement.

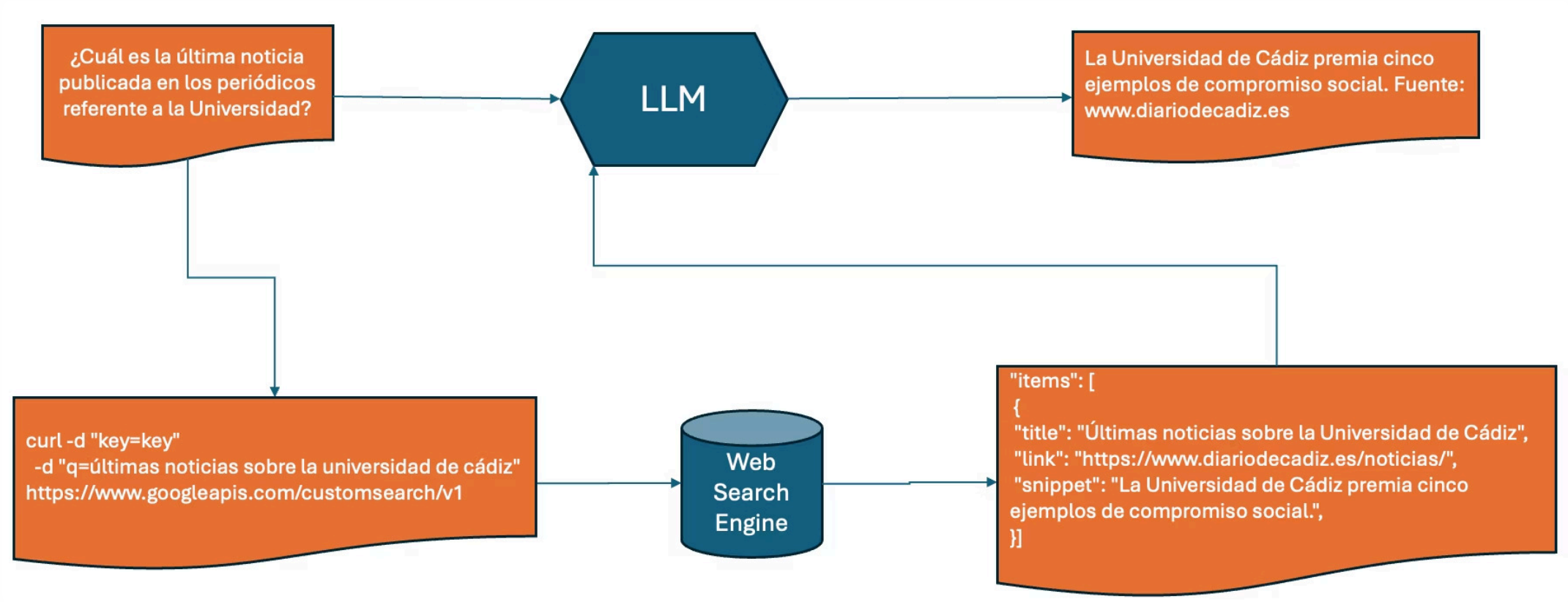


Semantic Search in Relational Databases: system prompt

"You are an expert in writing SQL queries.
You have access to a {{sqlDialect}} database with the following structure:
{{databaseStructure}}
If a user asks a question that can be answered by querying this database, generate an SQL SELECT query. Do not output anything else aside from a valid SQL statement!"

Ex 2. Creating a semantic search engine in a relational database

Semantic search on the web



Ex 3. Creating a semantic search engine on the web

Web search platforms

Search engines

Google, Bing, and DuckDuckGo offer APIs to access online information.

Web search APIs

Tavily, SerpApi, Serper and SearchApi are platforms specialised in web information search.

Web search APIs

These APIs allow applications to integrate search functions without developing their own engine.

to do

Information Generation

Information Generation

Information generation is a complex process that requires understanding natural language, processing information, and generating creative text, adapting the language and style to the user's needs.

The LLM combines the input prompt with relevant information to generate the output.

The quality of the output depends on the quality of the information retrieved and the capability of the LLM itself.

Information Generation

The LLM must be instructed with a specific prompt in order to generate the desired response

"

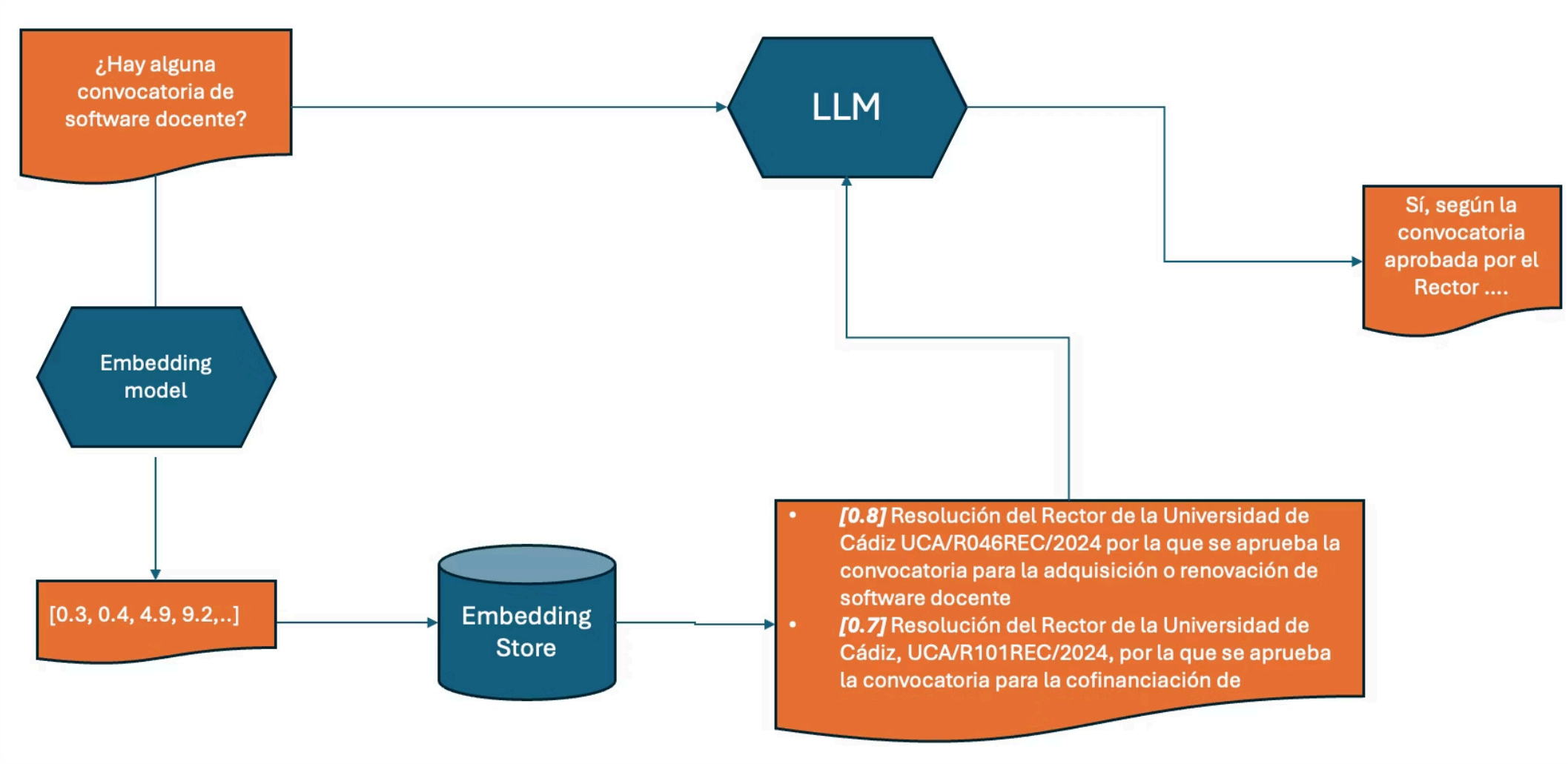
...
Answer the question based only on the following context:

{context}

Question: {question}

"

Generation (example)



Ex 4. Creating assistants powered by RAG

Summary

This presentation covered the evolution of information in the field of artificial intelligence (AI), specifically in the context of information retrieval and generation.

We explored how RAG, a technology that combines information retrieval with text generation, revolutionised the way machines understand and interact with knowledge.