Retrieval-Augmented Generation for talking with your private data using LLM

Enrico Zimuel, *Tech Lead & Principal Software Engineer*

The Italian Artificial Intelligence Conference
1st Dec 2023, Turin (Italy)

Summary

- Introduction to RAG
- Semantic search and vectors
- Embedding
- Vector databases
- Split documents in chunk
- LangChain
- Demo

elastic

About me

- Professional programmer since 1996
- Open source contributor (e.g. Zend Framework, Elasticsearch, LangChain)
- Tech Lead & Principal Software Engineer at Elastic (Mountain View, USA)
- Research Programmer at UvA
- Adj. Professor at University of Turin and ITS-ICT Piemonte
- TEDx and international speaker
- Author of books and articles

elastic

Retrieval-Augmented Generation (RAG)

- **RAG** is a technique in natural language processing that combines information retrieval systems with **Large Language Models** (LLM) to generate more informed and accurate responses
- It is composed by the following parts:
  - **Retrieval-Augmented**
  - **Generation**

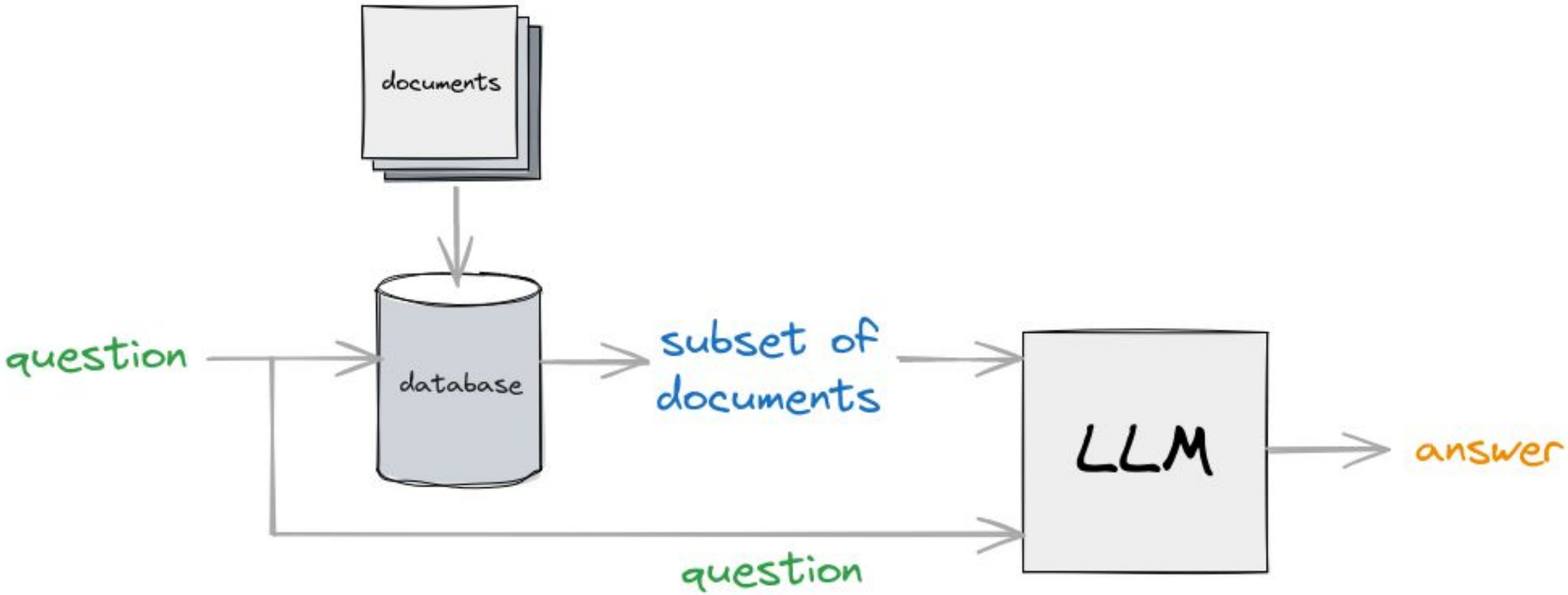
Generation

- LLMs like ChatGPT are a disruptive technology
- They are very useful and powerful in many industries
- But they have some limitations:
  - **No source** (potential hallucinations)
    - How can I verify the information coming from an LLM?
    - What sources has been used to generate the answer?
  - **Out of date**
    - An LLM is trained in a period of time
    - For update we need to retraining the model

Retrieval-Augmented

- We collect sets of private or public document
- We build a **retrieval system** (e.g. a database) to extract a subset of documents using a **question**
- Then we pass the **question + documents found** to an LLM as prompt with a context
- The LLM can give an answer using the updated documents

RAG diagram



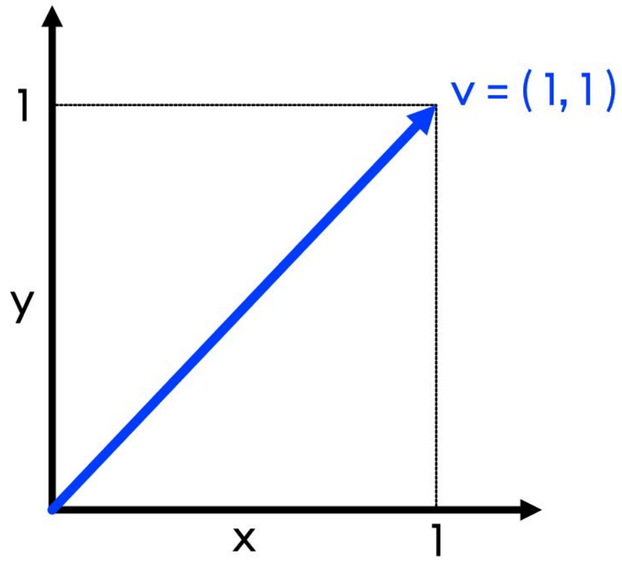
elastic

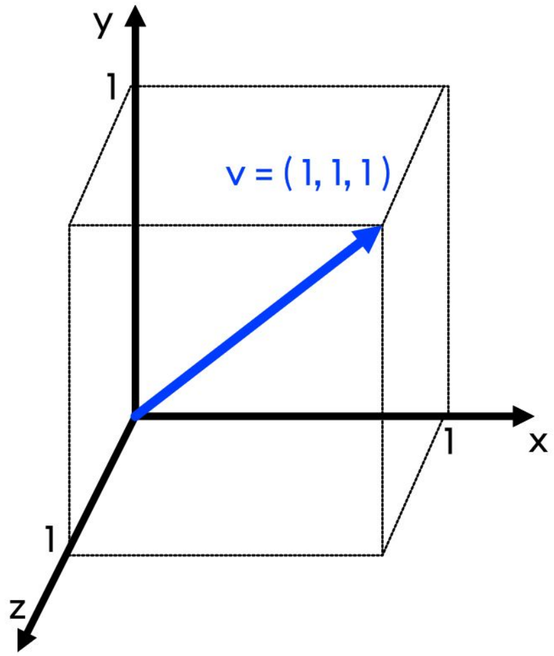
Retrieve documents from a question

- How we can retrieve documents in a database using a question?
- We need to use **semantic search**
- One possible solution is to use a **vector database**
- A vector database is a system that uses **vectors** to retrieve information

What is a vector?

- A vector is a set of numbers
- Example: a vector of 3 elements [10.5, 11.23, -10]
- A vector can be represented in a multi-dimensional space

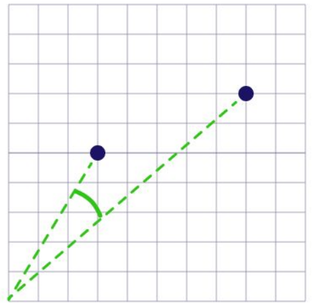




elastic

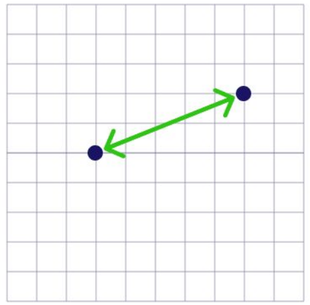
Similarity between two vectors

- Two vectors are (semantically) similar if they are close to each other
- We need to define a way to measure the similarity



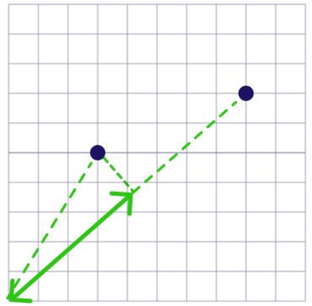
Cosine Distance

$$1 - \frac{A \cdot B}{||A|| \quad ||B||}$$



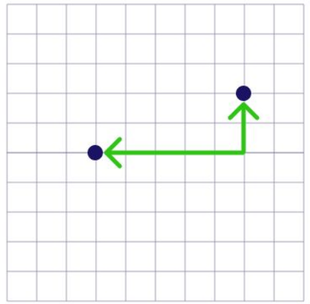
Squared Euclidean (L2 Squared)

$$\sum_{i=1}^{n} (x_i - y_i)^2$$



Dot Product

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$



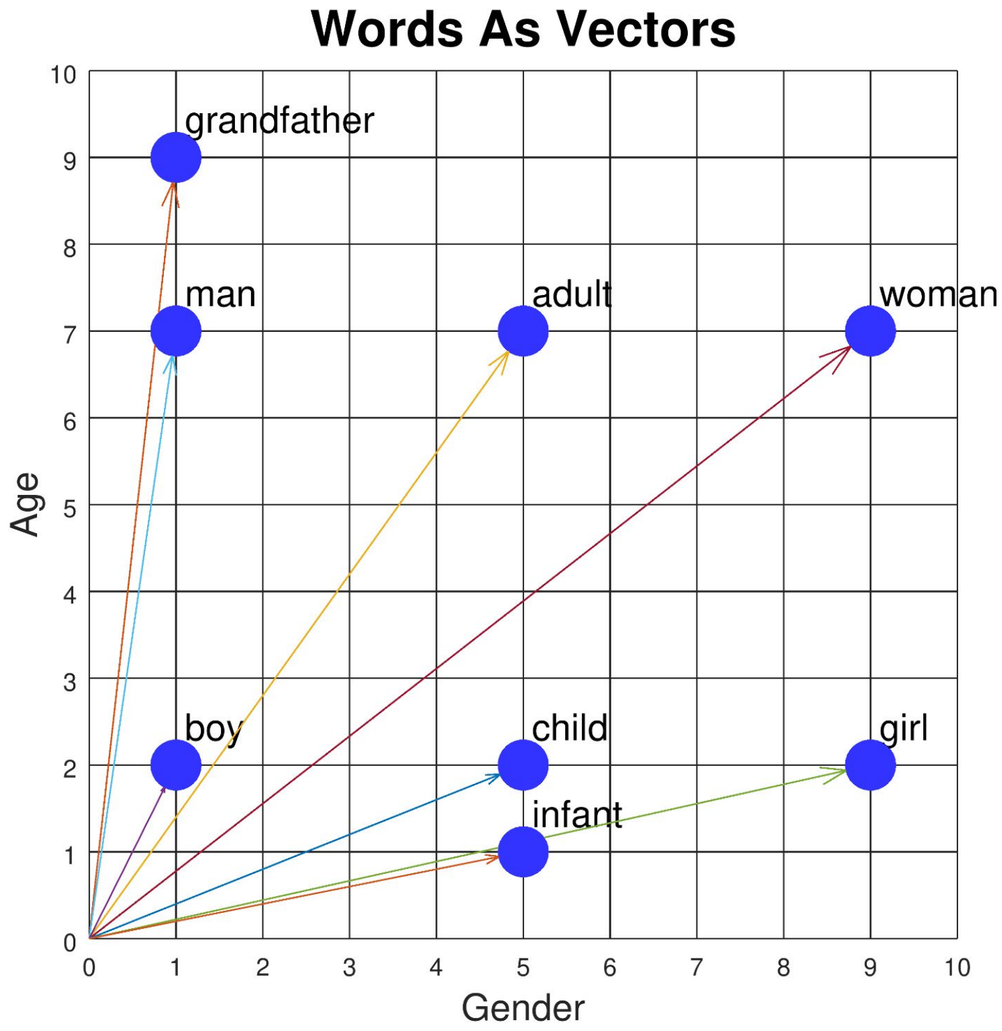
Manhattan (L1)

$$\sum_{i=1}^{n} |x_i - y_i|$$

elastic

Embedding

- Embedding is the translation of an input (document, image, sound, movie, etc) to a vector
- There are many techniques, using an LLM typically this is done by a neural network
- The goal is to group information that are semantically related to each other



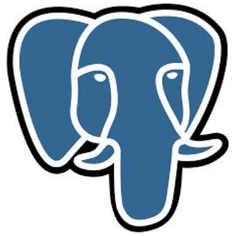
elastic

Vector databases

- There are many vector databases available:
  - Chroma
  - Elasticsearch
  - Weaviate
  - Qdrant
  - PostgreSQL (using pgvector)
  - etc







elastic

Vector database + LLM

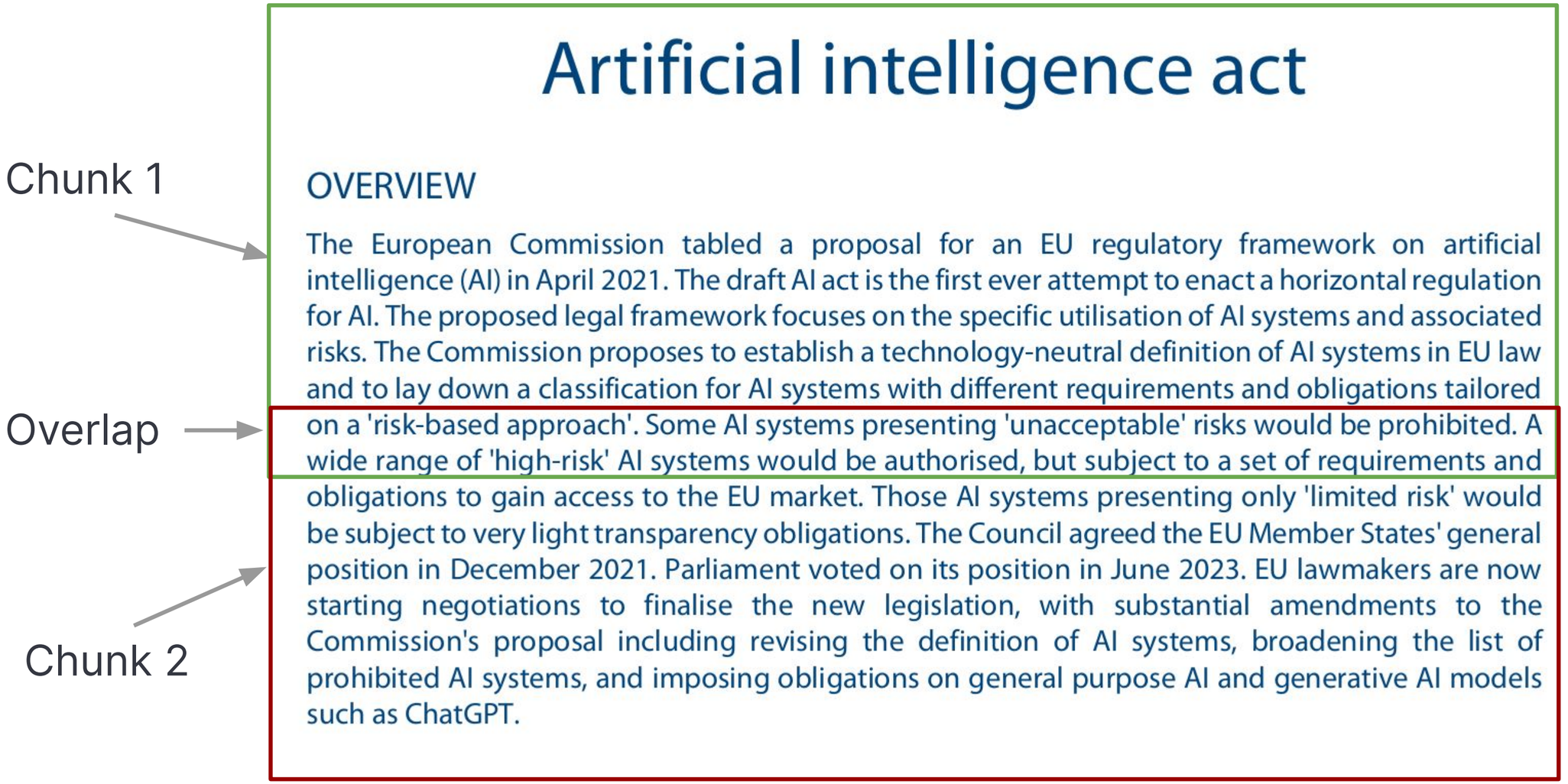
- We can query a vector database using natural language (e.g. a question)
- The query produces a set of relevant documents ordered by a score
- We can extract the top-3 score documents and pass it as **context** for a prompt using the previous **question**
- The prompt used for the LLM will be something like this:
  - *Given the following **{context}** answer to the following **{question}***

elastic

Split the documents in chunk

- We need to store data in the vector database using chunk of information
- We cannot use big documents since we need to pass it in the context part of the prompt for an LLM that typically has a token limit (e.g. gpt-3.5-turbo from 4k to 16k)
- We need to split the documents in chunk (size of characters)
- There are some techniques to split the documents to avoid semantic breakings

Text splitter

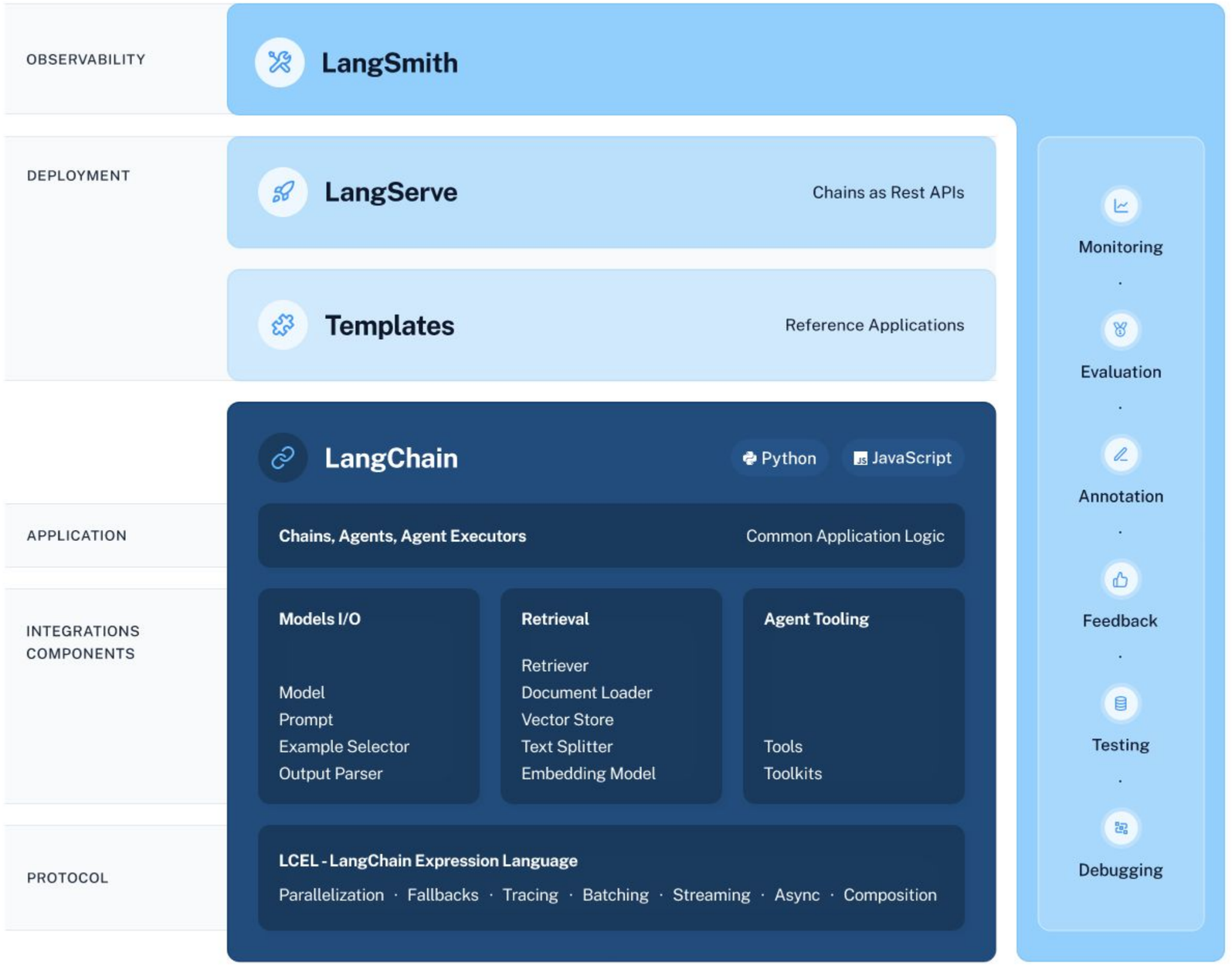


elastic

LangChain

- LangChain is a framework designed to simplify the creation of applications using LLM
- Features: Chains, Agents, Execution, Memory, Retriever (vector store), Tools
- Available for Python (⭐70K) and Javascript (⭐9.1K)

elastic


OBSERVABILITY
LangSmith
DEPLOYMENT
LangServe
Chains as Rest APIs
Templates
Reference Applications
LangChain
Python
JavaScript
APPLICATION
Chains, Agents, Agent Executors
Common Application Logic
INTEGRATIONS
COMPONENTS
Models I/O
Model
Prompt
Example Selector
Output Parser
Retrieval
Retriever
Document Loader
Vector Store
Text Splitter
Embedding Model
Agent Tooling
Tools
Toolkits
PROTOCOL
LCEL - LangChain Expression Language
Parallelization · Fallbacks · Tracing · Batching · Streaming · Async · Composition
Monitoring
Evaluation
Annotation
Feedback
Testing
Debugging

elastic

DEMO

Build a RAG using LangChain + ChatGPT + Elasticsearch

elastic

Thanks!

More information:
https://search-labs.elastic.co/search-labs

Contact:
enrico.zimuel (at) elastic.co