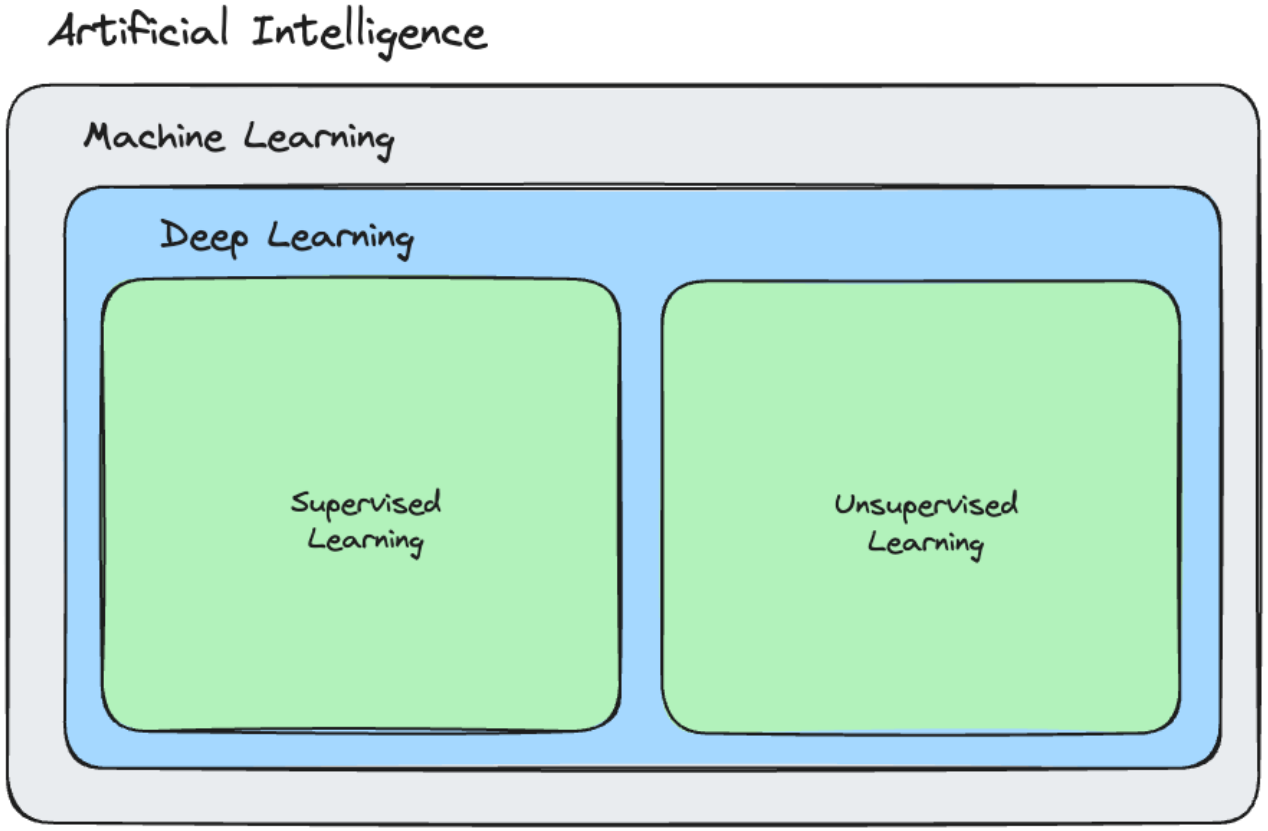

spring® AI

Artificial Intelligence



Supervised Learning

Supervised learning is a machine learning approach where the model is trained using a labeled dataset.

In this context, every training example is associated with an output label.

The aim of supervised learning is to establish a link between inputs and outputs, which can then be utilized to predict labels for new, unseen data.

Unsupervised Learning

Unsupervised learning is a type of machine learning where the model is trained on data without labeled responses.

In other words, the algorithm is given a dataset with input data but without any corresponding output values.

The goal of unsupervised learning is to find hidden patterns or intrinsic structures in the input data.

Large Language Models

LLMs are trained on vast amounts of training data and this is powered by lots of compute power.

LLMs use unsupervised or self-supervised learning during their training phase.

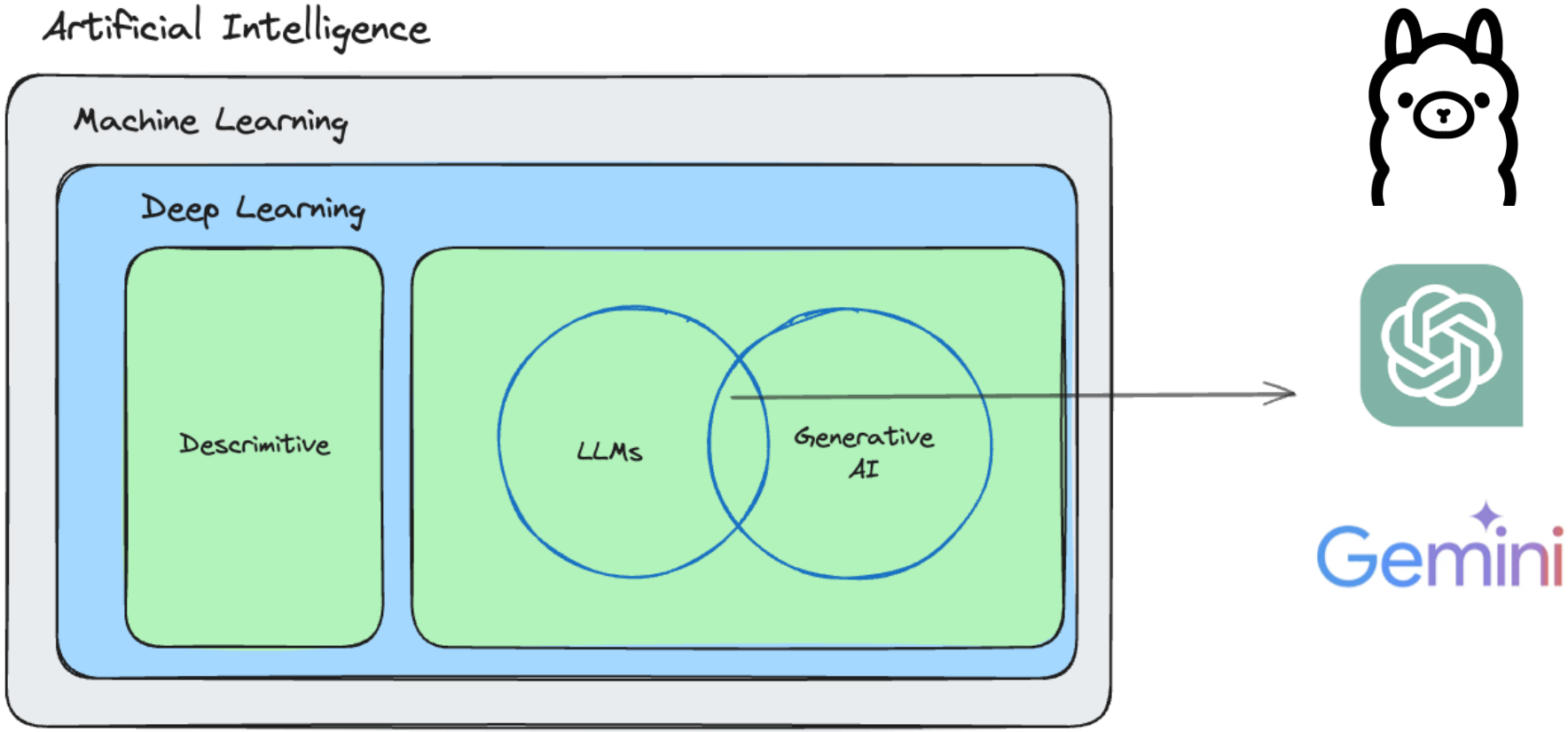
They predict the next word in a sentence (or fill in blanks) using context from the surrounding text, learning from the structure and patterns within the data itself without explicit labels.

Generative AI

Generative AI is based on the transformer architecture which is able to take that training data and generate something brand new at the intersection of LLMs.

Generative AI is a broader concept encompassing various techniques and models, including but not limited to LLMs.

Generative AI is what powers applications like Open AI's GPT & Google Gemini.



- OpenAi
- Google Gemini
- Ollama

Tokens

Tokens are the currency of LLMs - large language models process text using tokens, which are common sequences of characters found in a set of text.

- OpenAi pricing
- Tokenizer

Spring AI

AI for java developers!

Simplifies interactions with LLMs with a Spring abstraction.

Simplifies interactions with Vector databases, Observability, ChatMemory and many more...

Created in 2023 by Mark Pollack and Christian Tzolov.

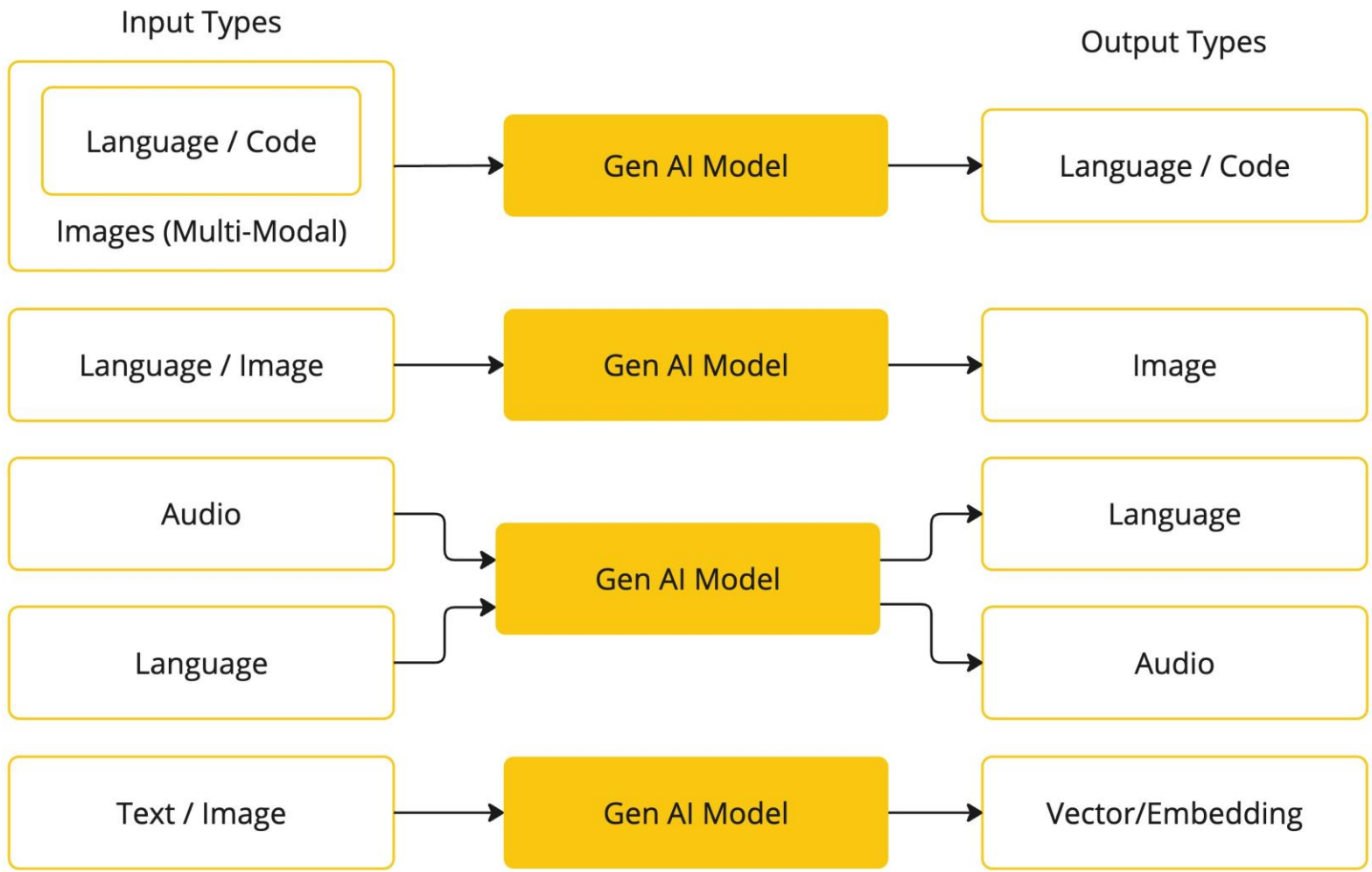
- Documentation

Spring AI

The project draws inspiration from notable Python projects, such as LangChain and LlamaIndex, but Spring AI is not a direct port of those projects.

The project was founded with the belief that the next wave of Generative AI applications will not be only for Python developers but will be ubiquitous across many programming languages.

Current version: Spring AI 1.0.0-M4


Input Types
Output Types
Language / Code
Images (Multi-Modal)
Gen AI Model
Language / Code
Language / Image
Gen AI Model
Image
Audio
Language
Gen AI Model
Language
Audio
Text / Image
Gen AI Model
Vector/Embedding

Chat Client

The ChatClient offers a fluent API for communicating with an AI Model.

It supports both a synchronous and reactive programming model.

DEMO package chat

Chat Memory

The web is stateless, and we need to remember that when working the various REST endpoints that the LLMs provide.

This might be a bit confusing because you have probably used ChatGPT before, and it remembers previous conversations and can build upon them. You have to remember that ChatGPT is a product that talks to an LLM like GPT-4o and the product is what preserves conversational history.

DEMO package memory

Prompts

Prompts serve as the foundation for the language-based inputs that guide an AI model to produce specific outputs.

For those familiar with ChatGPT, a prompt might seem like merely the text entered into a dialog box that is sent to the API.

However, it encompasses much more than that. In many AI Models, the text for the prompt is not just a simple string.

Prompt engineering

Prompt engineering is the process of structuring an instruction that can be interpreted and understood by a generative artificial intelligence model.

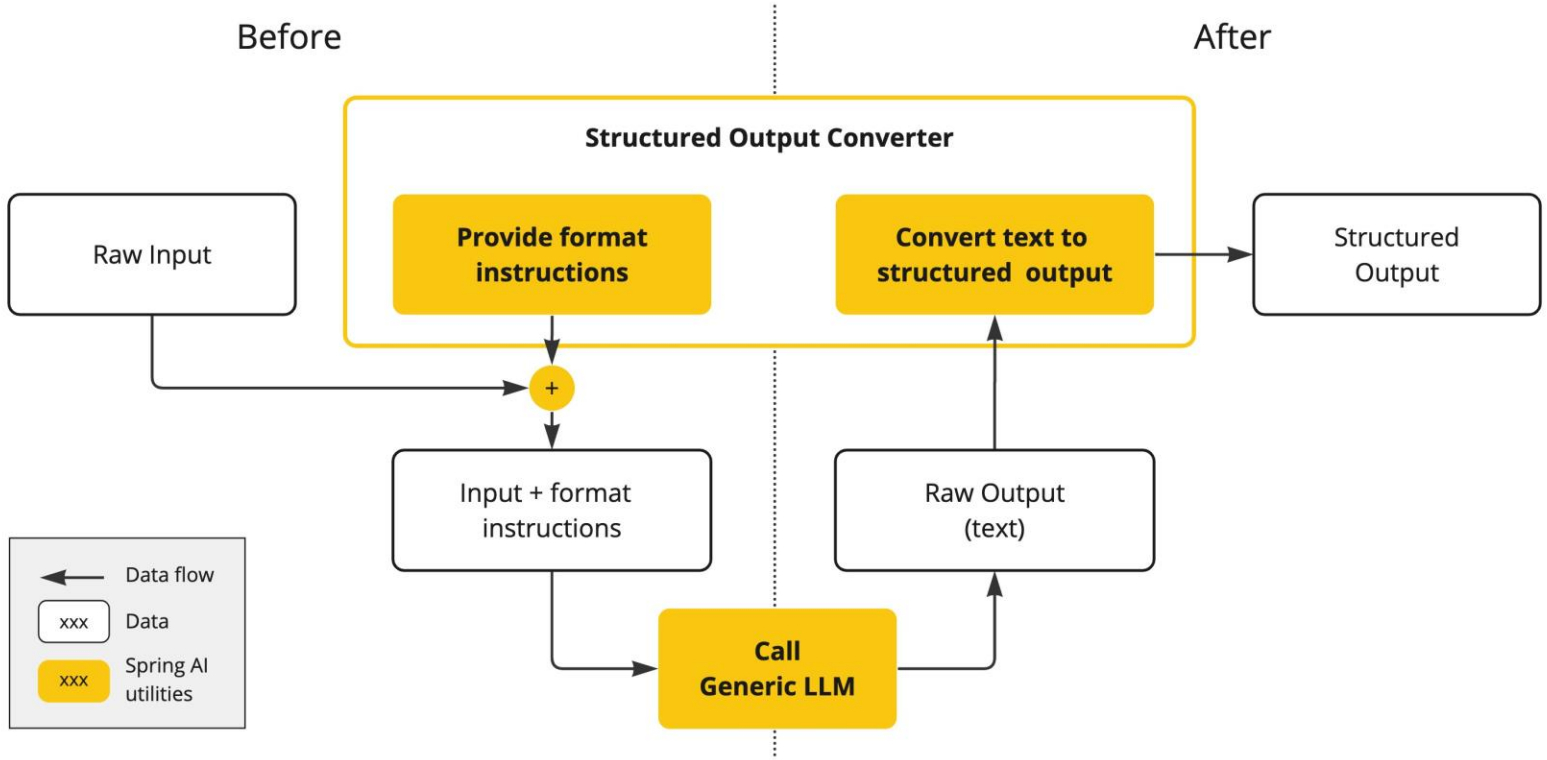
Six strategies for getting better results

- Write clear instructions
- Provide reference text
- Split complex tasks into simpler subtasks
- Give the model time to "think"
- Use external tools
- Test changes systematically

- Prompt Engineering Guidelines from Open AI

DEMO package prompt

Structured Output



DEMO package output

Limitations of AI Models

Are trained on public knowledge up to a certain date.
(Gpt-4o October 2023)

They don't know about your private / corporate data.

Don't have access to realtime data.

Can't learn from your data.

Bring Your Own Data

Train new models ❌

Fine tune existing models ❌

"Stuff the prompt" ✅

Retrieval Augmented Generation ✅

Function calling ✅

Fine Tuning

This traditional machine learning technique involves tailoring the model and changing its internal weighting.

However, it is a challenging process for machine learning experts and extremely resource-intensive for models like GPT due to their size.

Additionally, some models might not offer this option.

Prompt Stuffing

Involves embedding your data within the prompt provided to the model.

Given a model's token limits, techniques are required to present relevant data within the model's context window.

This approach is colloquially referred to as "stuffing the prompt."

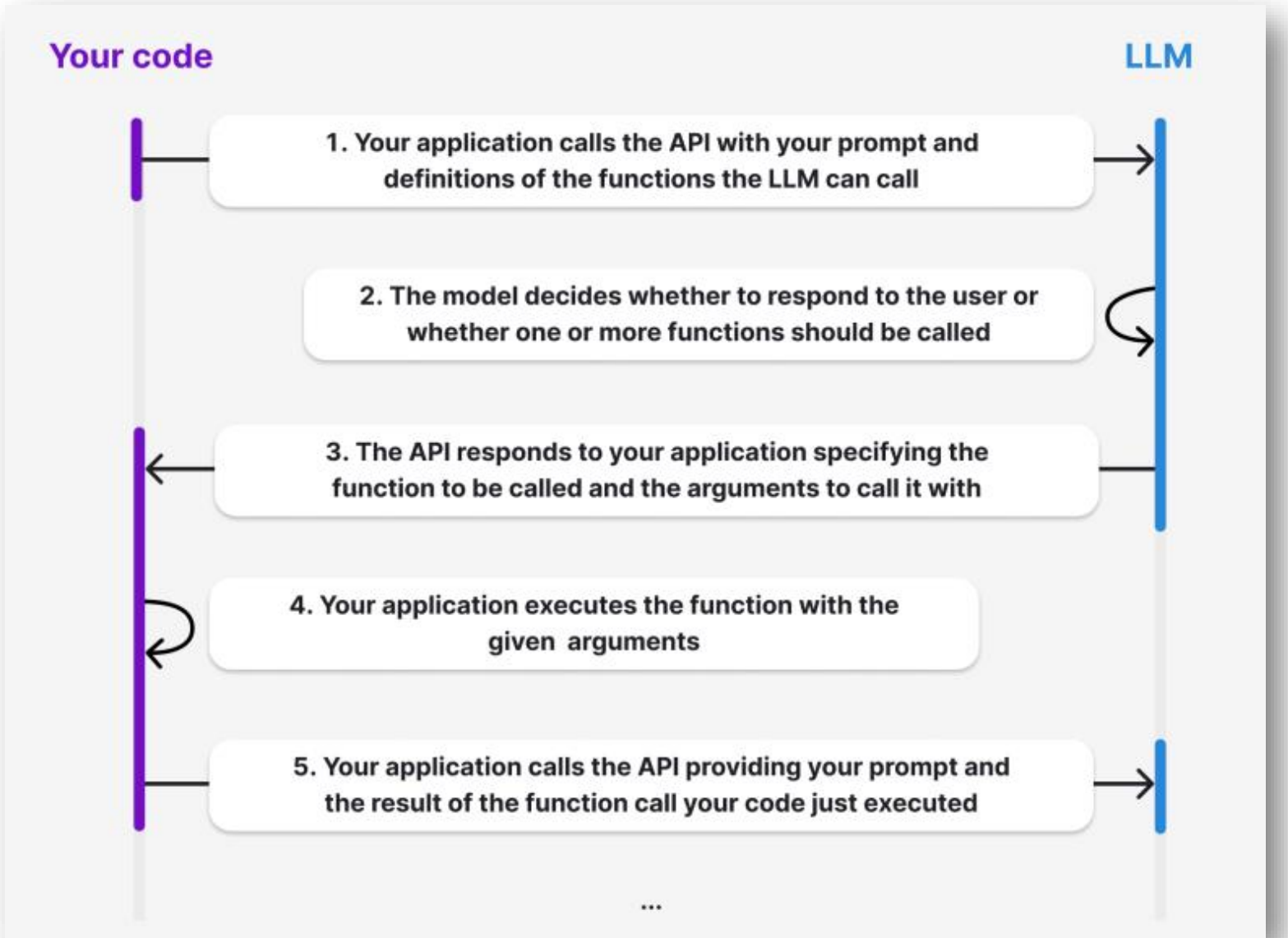
The Spring AI library helps you implement solutions based on the "stuffing the prompt" technique otherwise known as Retrieval Augmented Generation (RAG).

DEMO package stuff

Function Calling

This technique allows registering custom, user functions that connect the large language models to the APIs of external systems.

Spring AI greatly simplifies code you need to write to support function calling.



DEMO package functions

Retrieval Augmented Generation (RAG)

A technique termed Retrieval Augmented Generation (RAG) has emerged to address the challenge of incorporating relevant data into prompts for accurate AI model responses.

The approach involves a batch processing style programming model, where the job reads unstructured data from your documents, transforms it, and then writes it into a vector database.

At a high level, this is an ETL (Extract, Transform and Load) pipeline. The vector database is used in the retrieval part of RAG technique.

💰 Tokens are the currency of LLMs

Tokenizer

Learn about language model tokenization

OpenAI's large language models process text using **tokens**, which are common sequences of characters found in a set of text. The models learn to understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens. Learn more.

You can use the tool below to understand how a piece of text might be tokenized by a language model, and the total count of tokens in that piece of text.

GPT-4o & GPT-4o mini | GPT-3.5 & GPT-4 | GPT-3 (Legacy)

```
Java Champion, Spring Developer Advocate, YouTuber and Lifelong Learner
Hello 👋🏻 My name is Dan Vega, Java Champion, Spring Developer Advocate,
Husband and #GirlDad based outside of Cleveland OH. I created this
website as a place to document my journey as I learn new things and share
them with you. I have a real passion for teaching and I hope that one of
blog posts, videos or courses helps you solve a problem or learn
something new.
```

Clear | Show example

Tokens	Characters
95	439

```
Java Champion, Spring Developer Advocate, YouTuber and Lifelong Learner
Hello ▯▯ My name is Dan Vega, Java Champion, Spring Developer Advocate,
Husband and #GirlDad based outside of Cleveland OH. I created this
website as a place to document my journey as I learn new things and
share them with you. I have a real passion for teaching and I hope that
one of blog posts, videos or courses helps you solve a problem or learn
something new.
```

Text | Token IDs

A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

If you need a programmatic interface for tokenizing text, check out our tiktoken package for Python. For JavaScript, the community-supported @dbdq/tiktoken package works with most GPT models.

Model	Pricing	Pricing with Batch API*
gpt-4o	$2.50 / 1M input tokens	$1.25 / 1M input tokens
	$1.25 / 1M cached** input tokens	
	$10.00 / 1M output tokens	$5.00 / 1M output tokens
gpt-4o-2024-08-06	$2.50 / 1M input tokens	$1.25 / 1M input tokens
	$1.25 / 1M cached** input tokens	
	$10.00 / 1M output tokens	$5.00 / 1M output tokens
gpt-4o-audio-preview	Text	
	$2.50 / 1M input tokens	
	$10.00 / 1M output tokens	
	Audio***	
	$100.00 / 1M input tokens	
	$200.00 / 1M output tokens	
gpt-4o-audio-preview-2024-10-01	Text	
	$2.50 / 1M input tokens	
	$10.00 / 1M output tokens	
	Audio***	
	$100.00 / 1M input tokens	
	$200.00 / 1M output tokens	
gpt-4o-2024-05-13	$5.00 / 1M input tokens	$2.50 / 1M input tokens
	$15.00 / 1M output tokens	$7.50 / 1M output tokens

Model	Context Window Size	Description
GPT-4 (32K)	32,768 tokens	For handling larger, complex tasks.
GPT-3.5 Turbo	4,096 tokens	Streamlined, fast chat model from OpenAI.
Claude 2	100,000 tokens	Ideal for long documents and chat memory.
Claude 3.5	200,000 tokens	Twice the capacity of Claude 2, useful for extended workflows.
Gemini 1.5 Flash	1 million tokens	Designed for in-depth, multimodal analysis.
Gemini 1.5 Pro	2 million tokens	The largest context window, suitable for vast input like entire codebases or hours of media.
LLaMA 2	4,096 - 8,192 tokens	Open-source model with multiple versions.
Falcon LLM	2,048 - 4,096 tokens	Lightweight, efficient for various applications.

Why RAG Matters

- Large Language Models are trained on data up to a certain date
- Not trained on your data
- Hallucinations

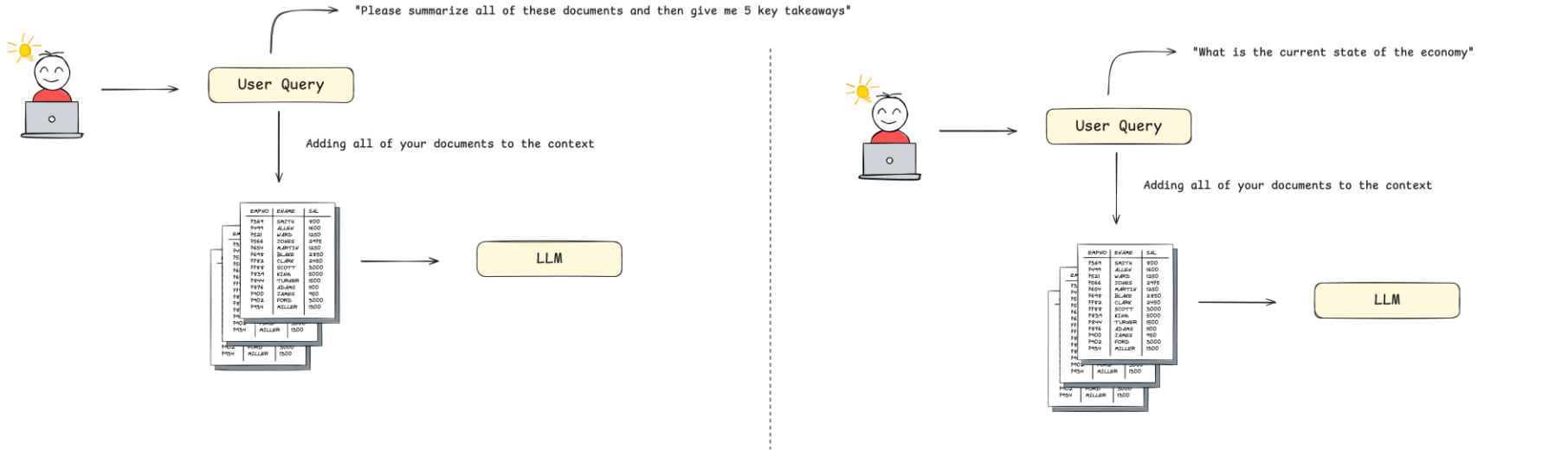
**What is RAG **

- Retrieval Augmented Generation
- Combines **retrieval-based methods** with **generative models** to produce more accurate and contextually relavant outputs
- Allows LLMs to access external knowledge sources, mitigating limitations like outdated information or knowledge cutoff

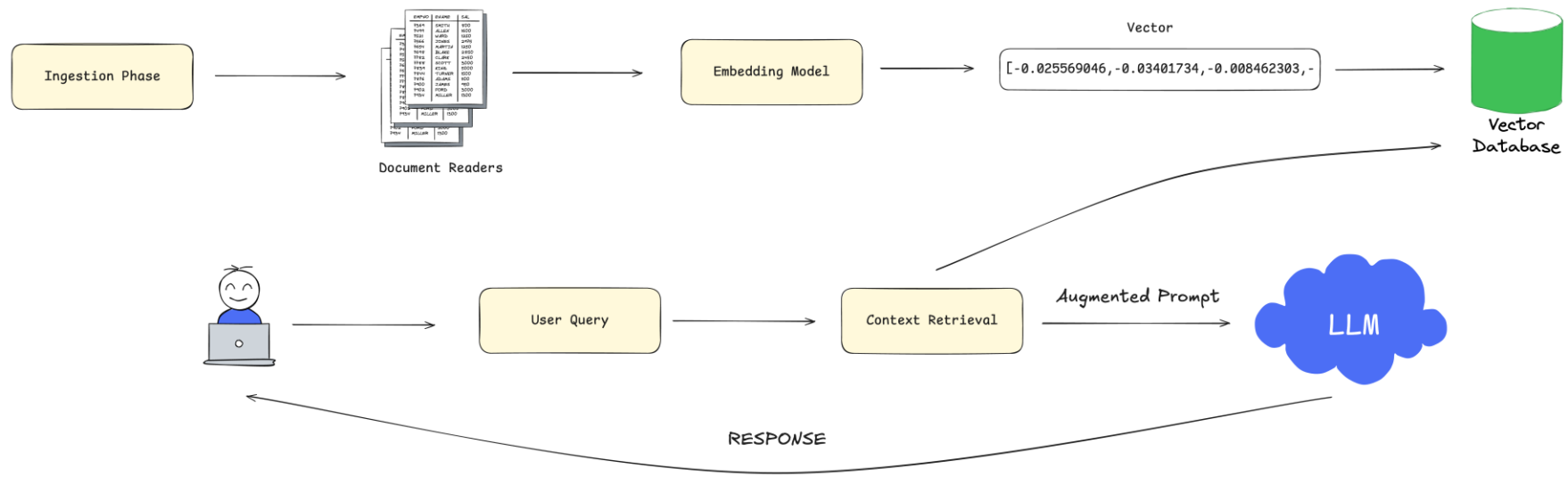
What it is NOT

- Sending all of your documents in the context of each request
- If you want to summarize an entire document this make sense but is NOT RAG
- Context Window Limits

NOT RAG



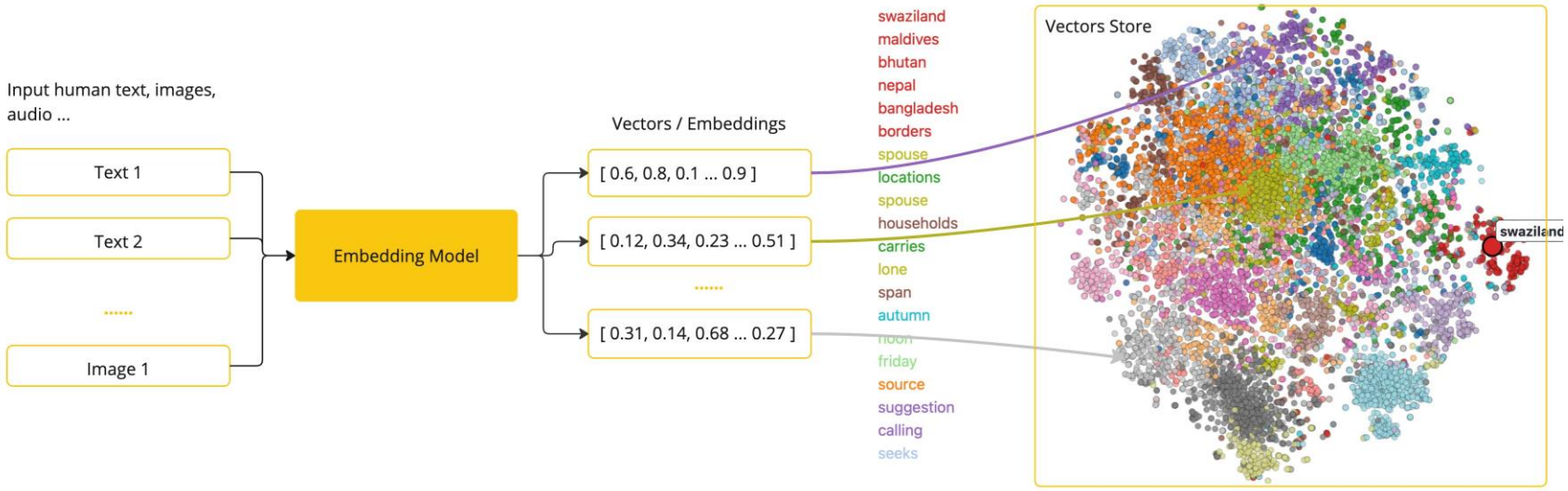
RAG



Embedding model

Embeddings transform text into numerical arrays or vectors, enabling AI models to process and interpret language data.

This transformation from text to numbers is a key element in how AI interacts with and understands human language.



Vector Databases

A vector databases is a specialized type of database that plays an essential role in AI applications.

In vector databases, queries differ from traditional relational databases. Instead of exact matches, they perform similarity searches.

When given a vector as a query, a vector database returns vectors that are “similar” to the query vector.

- PgVector Store
- Azure Vector Search
- Oracle Vector Store
- Redis Vector Store
- SimpleVectorStore

Similarity Search

Selects the closest vectors to a given vector

{-0.436, 0.578, 0.935, 0.2193}

Vector
{0.345, -0.465, 0.856, 0.1543}
{-0.445, 0.565, 0.956, 0.2543}
{0.545, 0.665, 0.056, 0.3543}
{0.645, 0.765, 0.156, -0.4543}
{0.745, 0.865, 0.256, 0.5543}
{0.845, 0.965, -0.356, 0.6543}

DEMO package rag and project rag-example

Multimodal

Humans process knowledge, simultaneously across multiple modes of data inputs.

The way we learn, our experiences are all multimodal.

We don’t have just vision, just audio and just text.

DEMO package vision

Generative AI Challenges

Challenge	Solution
Align responses to goals	System prompts
Stateless APIs	Chat Memory
No structured output	Output converters
Not trained on your data	Prompt stuffing
Limited context size	RAG
Not aware of your APIs or real-time data	Function calling

DEMO project multiple-llms

Github Repos

- Spring AI
- RAG
- Multiple LLMs