Intelligent Applications

With Spring AI

Patrick Baumgartner
42talents GmbH, Zürich, Switzerland

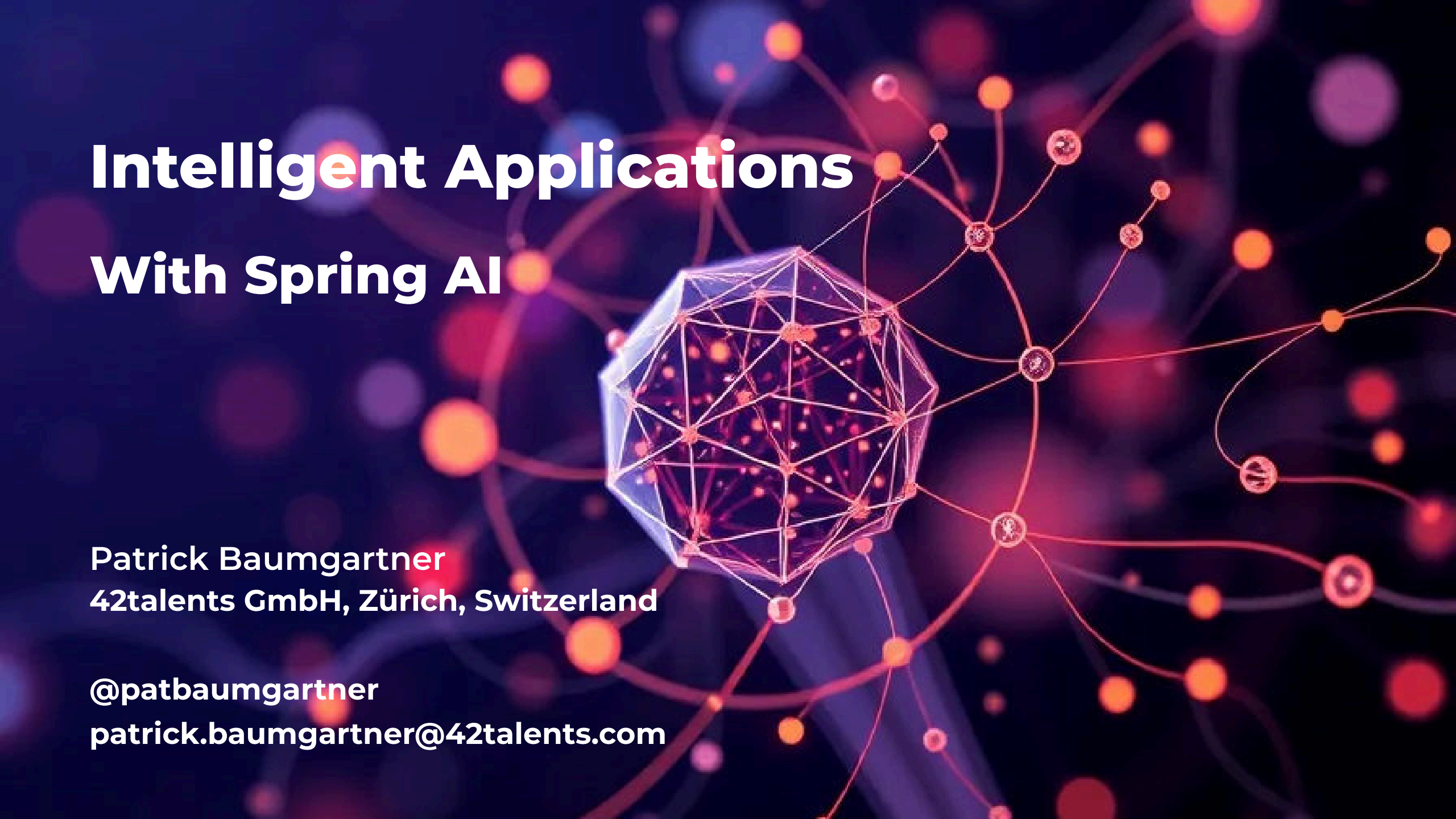
@patbaumgartner
patrick.baumgartner@42talents.com

Abstract

Intelligent Applications with Spring AI

The rise of AI has revolutionised our world and opened up countless new possibilities. Although the range of AI solutions is still growing, we can now develop applications with advanced language capabilities much faster.

In this talk, we will look at different language models (LLMs) and explore their application in different scenarios such as chat interaction, image generation and audio transcription. Based on the Spring AI project, we will learn about different AI models and their functionalities and look at their practical integration in the enterprise environment.



Intelligent Applications With Spring AI

Patrick Baumgartner
42talents GmbH, Zürich, Switzerland

@patbaumgartner
patrick.baumgartner@42talents.com

Introduction

Patrick Baumgartner

Technical Agile Coach @ **42talents**

My focus is on the **development of software solutions *with* humans**.

Coaching, Architecture, Development, Reviews, and Training.

Lecturer @ **Zurich University of Applied Sciences ZHAW**

Co-Organizer of **Voxxed Days Zurich**, **JUG Switzerland**, Oracle ACE Pro Java ...

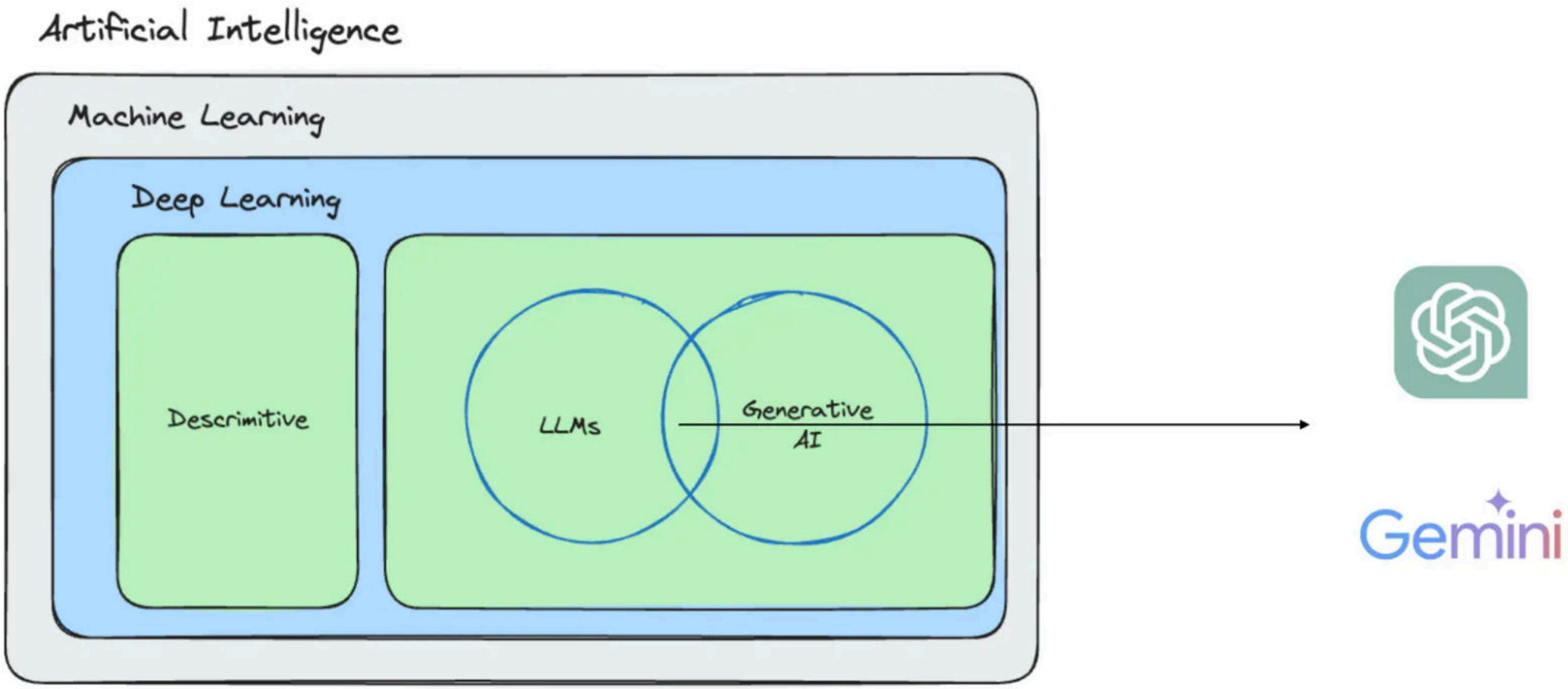
@patbaumgartner

⚠️ WARNING

This talk is focused exclusively on the technical integration of Spring AI with Large Language Models (LLMs). It will not address topics such as prompt engineering, fine-tuning, or model selection. Additionally, important considerations like sustainability, social impact, and the broader sense or nonsense of AI applications are beyond the scope of this presentation.

What is Artificial Intelligence (AI)?

What is Artificial Intelligence (AI)?



Examples ...

Examples ...

- Query data with natural language
  - *"Why did the user aks for their account to be closed?"*
    - Checks emails, complaints, logs, customer support tickets, social media etc.

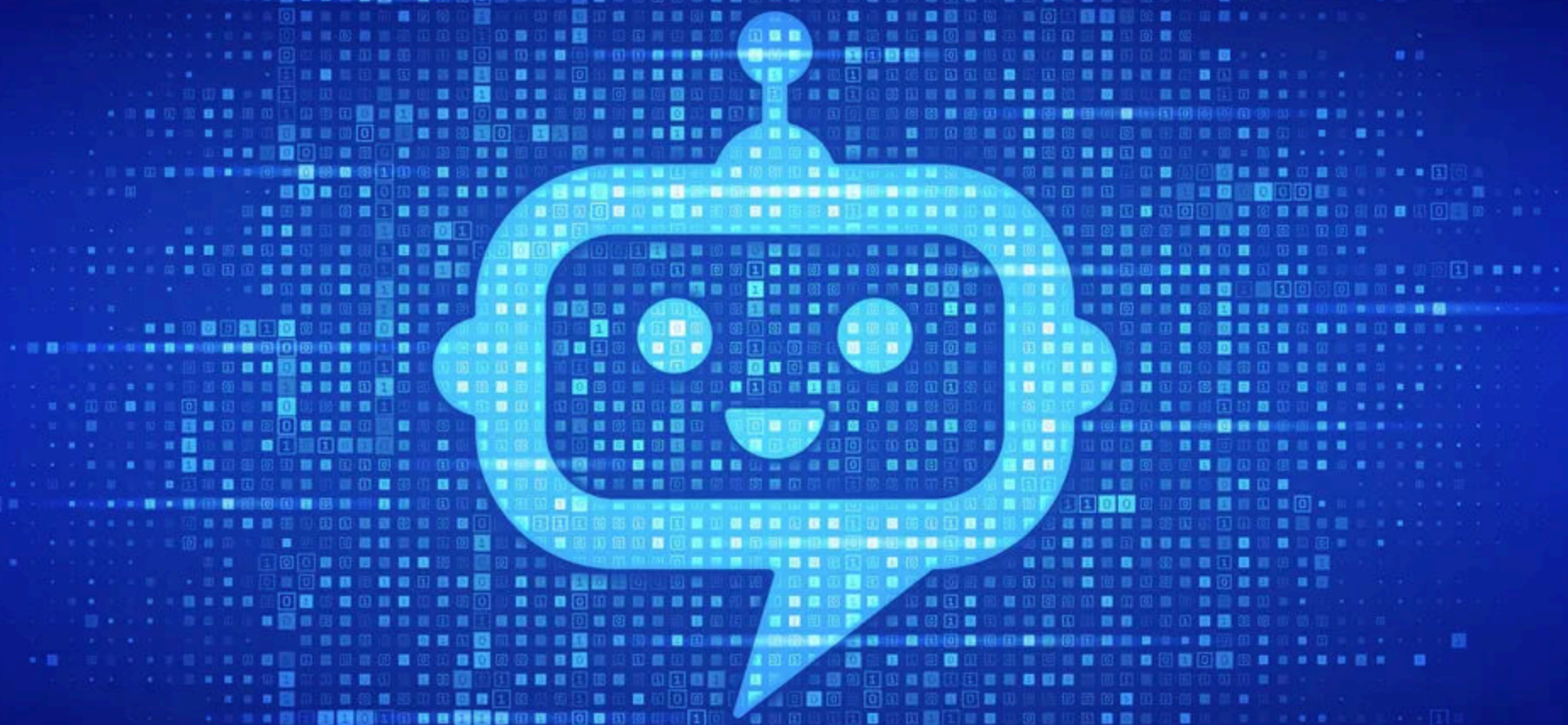
Examples ...

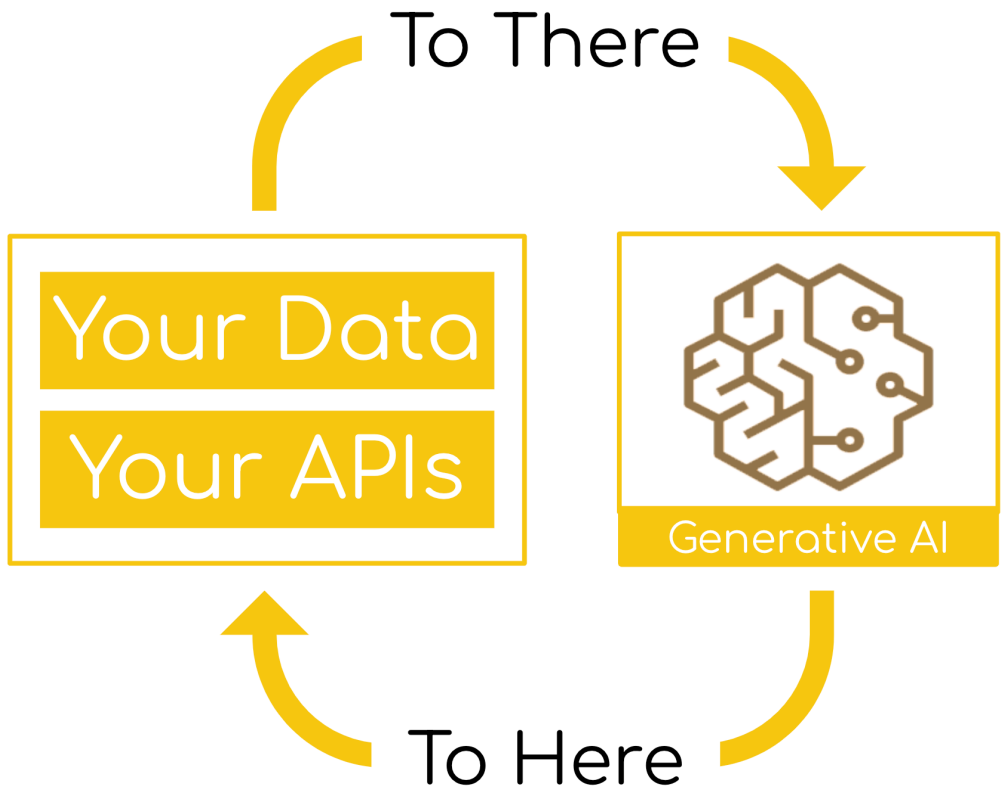
- Perform aggregation and transformation
  - Summarize text
  - Extract action items
  - Censor/identify sensitive data or abuse
  - Translate to different languages

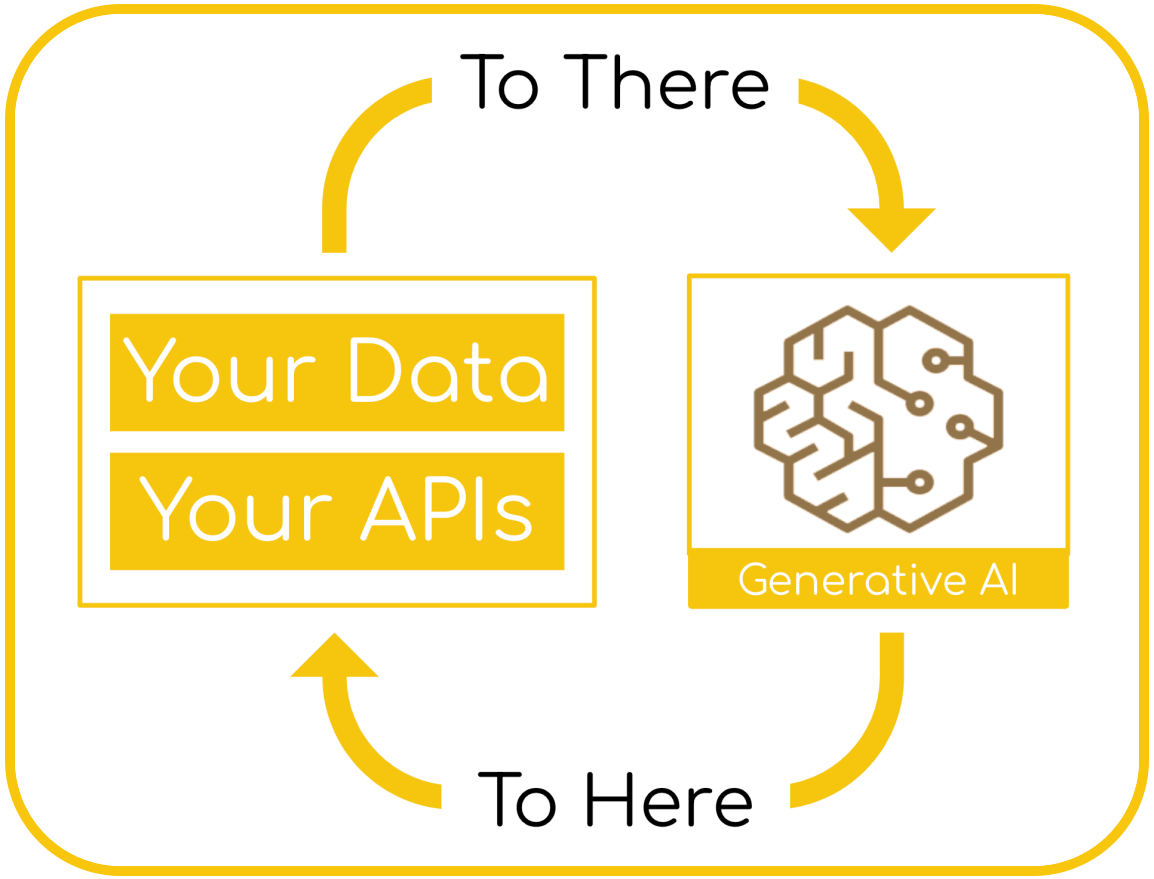
Examples ...

- Manipulate data
  - *"Reactivate customer account and notify them by email."*
- Define and execute business rules
  - *"If customer has not logged in for 6 months, send a reminder."*
- Generate code
  - *"Create a CSV exporter for all orders and write tests for it."*
- ...

Not all prompts are/need user input!




To There
Your Data
Your APIs
Generative AI
To Here

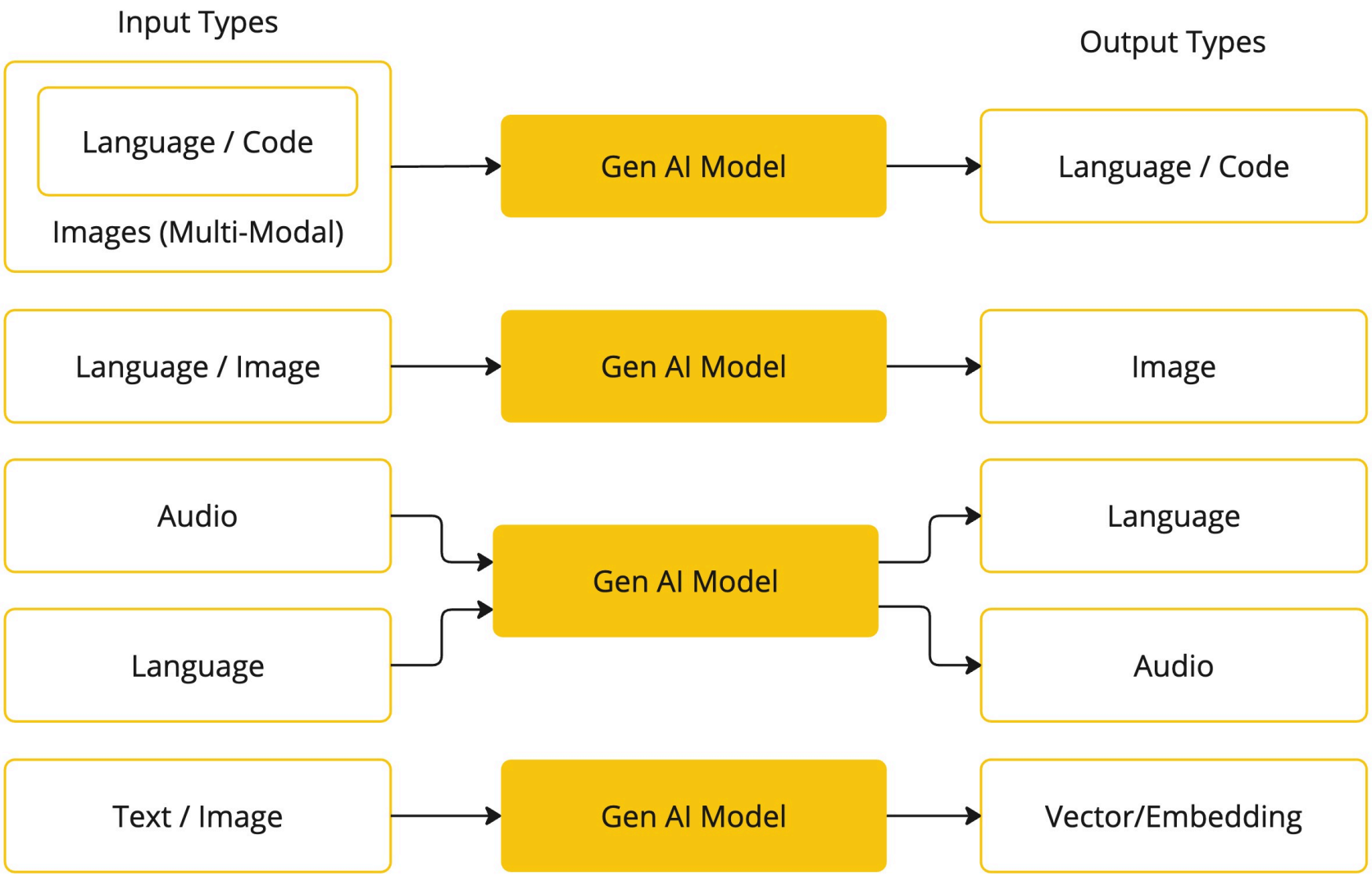


This is an integration problem!

Rest APIs Make AI Accessible

```
#!/usr/bin/env bash

echo "Calling OpenAI API ..."

PROMPT="Tell me a developer joke about Python."

curl https://api.openai.com/v1/chat/completions \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer $OPENAI_API_KEY" \
    --data \
"{
  \"model\": \"gpt-3.5-turbo\",
  \"messages\": [
    {
      \"role\": \"system\",
      \"content\": \"You are a friendly chatbot and you like to place emojis everywhere.\"
    },
    {
      \"role\": \"user\",
      \"content\": \"$PROMPT\"
    }
  ]
}"
```

AI Models



See: Spring AI: Models

https://olshansky.substack.com/p/vibe-checks-are-all-you-need

LLM Leaderboard - Comparison of GPT-4o, Llama 3, Mistral, Gemini and over 30 models

Comparison and ranking the performance of over 30 AI models (LLMs) across key metrics including quality, price, performance and speed (output speed - tokens per second & latency - TTFT), context window & others. For more details including relating to our methodology, see our FAQs.

For comparison of API Providers hosting the models see LLM API Providers Leaderboard

HIGHLIGHTS

Quality: GPT-4o (Aug 6) and Claude 3.5 Sonnet are the highest quality models, followed by GPT-4o & GPT-4 Turbo.

Output Speed (tokens/s): Gemma 7B (939 t/s) and Gemini 1.5 Flash (210 t/s) are the fastest models, followed by Llama 3.1 8B & Jamba 1.5 Mini.

Latency (seconds): Sonar Small (0.19s) and Sonar 3.1 Small (0.19s) are the lowest latency models, followed by Sonar 3.1 Large & Sonar Large.

Price ($ per M tokens): OpenChat 3.5 ($0.13) and Gemini 1.5 Flash ($0.13) are the cheapest models, followed by Gemma 7B & Mistral 7B.

Context Window: Gemini 1.5 Pro (2m) and Gemini 1.5 Flash (1m) are the largest context window models, followed by Codestral-Mamba & Jamba 1.5 Large.

PROMPT OPTIONS

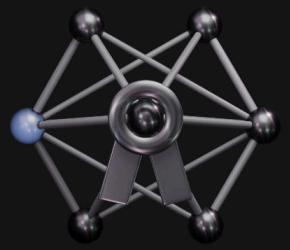
Filter, e.g. GPT, Meta

EXPAND COLUMNS

		CONTEXT	QUALITY	PRICE	OUTPUT TOKENS/S	LATENCY	
MODEL	CREATOR	CONTEXT WINDOW	INDEX *Normalized avg*	BLENDED *USD/1M Tokens*	MEDIAN *Tokens/s*	MEDIAN *First Chunk (s)*	FURTHER ANALYSIS
GPT-4o (Aug 6)	OpenAI	128k	77	$4.38	106.0	0.40	Model Providers
GPT-4o	OpenAI	128k	77	$7.50	103.2	0.40	Model Providers
GPT-4o mini	OpenAI	128k	71	$0.26	122.9	0.41	Model Providers
Llama 3.1 405B	Meta	128k	72	$5.00	28.8	0.67	Model Providers
Llama 3.1 70B	Meta	128k	65	$0.90	53.0	0.45	Model Providers
Llama 3.1 8B	Meta	128k	53	$0.16	174.3	0.35	Model Providers
Gemini 1.5 Pro	Google	2m	72	$5.25	61.6	0.92	Model Providers
Gemini 1.5 Flash	Google	1m	60	$0.13	209.9	0.39	Model Providers
Gemma 2 27B	Google	8k	49	$0.80	68.6	0.43	Model Providers
Gemma 2 9B	Google	8k	47	$0.20	126.1	0.33	Model Providers
Claude 3.5 Sonnet	ANTHROP\C	200k	77	$6.00	88.0	1.04	Model Providers
Claude 3 Opus	ANTHROP\C	200k	70	$30.00	26.6	1.76	Model Providers

scale | SEAL

Leaderboard



SEAL

Leaderboards

Expert-Driven Private Evaluations

Adversarial Robustness →

Learn More

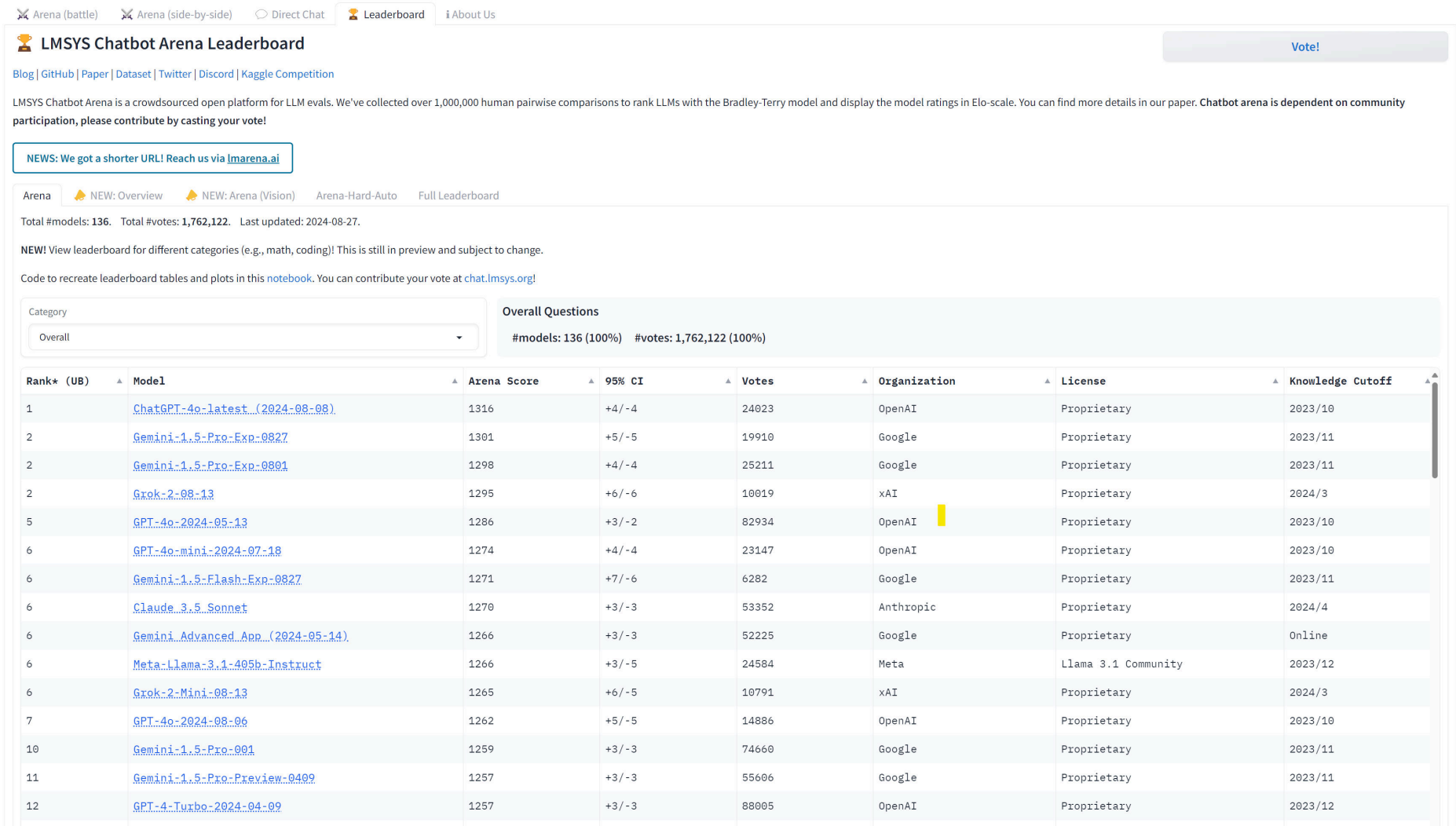
	Model	Number of Violations	95% Confidence
1st	Gemini 1.5 Pro (May 2024)	8	+8/-4
2nd	Llama 3.1 405B Instruct	10	+8/-5
3rd	Claude 3 Opus	13	+9/-5
4	Gemini 1.5 Flash Preview	14	+9/-6
5	Claude 3.5 Sonnet	16	+10/-6
6	GPT-4 Turbo Preview	20	+11/-7
7	Mistral Large	37	+14/-10
8	GPT-4o	67	+17/-14

Coding →

Learn More

	Model	Score	95% Confidence
1st	Claude 3.5 Sonnet	1165	+33/-33
2nd	GPT-4 Turbo Preview	1131	+24/-27
3rd	GPT-4o	1125	+28/-27
4	Llama 3.1 405B Instruct	1123	+33/-30
5	Gemini 1.5 Pro (May 2024)	1085	+28/-28
6	Claude 3 Opus	1045	+24/-24
7	Gemini 1.5 Flash Preview	1024	+29/-25
8	Gemini 1.5 Pro (April 2024)	984	+29/-27
9	Claude 3 Sonnet	970	+27/-28
10	Llama 3 70B Instruct	967	+22/-23
11	Mistral Large	908	+24/-25
12	Gemini 1.0 Pro	781	+29/-29
13	CodeLlama 34B Instruct	690	+33/-35

https://scale.com/leaderboard



Arena (battle) | Arena (side-by-side) | Direct Chat | Leaderboard | About Us

LMSYS Chatbot Arena Leaderboard

Vote!

Blog | GitHub | Paper | Dataset | Twitter | Discord | Kaggle Competition

LMSYS Chatbot Arena is a crowdsourced open platform for LLM evals. We've collected over 1,000,000 human pairwise comparisons to rank LLMs with the Bradley-Terry model and display the model ratings in Elo-scale. You can find more details in our paper. **Chatbot arena is dependent on community participation, please contribute by casting your vote!**

NEWS: We got a shorter URL! Reach us via lmarena.ai

Arena | NEW: Overview | NEW: Arena (Vision) | Arena-Hard-Auto | Full Leaderboard

Total #models: **136**. Total #votes: **1,762,122**. Last updated: 2024-08-27.

NEW! View leaderboard for different categories (e.g., math, coding)! This is still in preview and subject to change.

Code to recreate leaderboard tables and plots in this notebook. You can contribute your vote at chat.lmsys.org!

Category: Overall

Overall Questions

#models: 136 (100%) #votes: 1,762,122 (100%)

Rank* (UB)	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1	ChatGPT-4o-latest (2024-08-08)	1316	+4/-4	24023	OpenAI	Proprietary	2023/10
2	Gemini-1.5-Pro-Exp-0827	1301	+5/-5	19910	Google	Proprietary	2023/11
2	Gemini-1.5-Pro-Exp-0801	1298	+4/-4	25211	Google	Proprietary	2023/11
2	Grok-2-08-13	1295	+6/-6	10019	xAI	Proprietary	2024/3
5	GPT-4o-2024-05-13	1286	+3/-2	82934	OpenAI	Proprietary	2023/10
6	GPT-4o-mini-2024-07-18	1274	+4/-4	23147	OpenAI	Proprietary	2023/10
6	Gemini-1.5-Flash-Exp-0827	1271	+7/-6	6282	Google	Proprietary	2023/11
6	Claude 3.5 Sonnet	1270	+3/-3	53352	Anthropic	Proprietary	2024/4
6	Gemini Advanced App (2024-05-14)	1266	+3/-3	52225	Google	Proprietary	Online
6	Meta-Llama-3.1-405b-Instruct	1266	+3/-5	24584	Meta	Llama 3.1 Community	2023/12
6	Grok-2-Mini-08-13	1265	+6/-5	10791	xAI	Proprietary	2024/3
7	GPT-4o-2024-08-06	1262	+5/-5	14886	OpenAI	Proprietary	2023/10
10	Gemini-1.5-Pro-001	1259	+3/-3	74660	Google	Proprietary	2023/11
11	Gemini-1.5-Pro-Preview-0409	1257	+3/-3	55606	Google	Proprietary	2023/11
12	GPT-4-Turbo-2024-04-09	1257	+3/-3	88005	OpenAI	Proprietary	2023/12

https://lmarena.ai/

Advanced AI Concepts

Integrating Custom Data & APIs

How can you equip the AI model with information on which it has not been trained? GPT-3.5/4.0 only has knowledge up until October 2023*, limiting its ability to answer questions about more recent data.

* Why not asking Chat GPT with a prompt?

```
You said:

Can you tell me the date of the most recent information in your training data?
```

```
ChatGPT said:

The most recent information in my training data is from October 2023. If you have questions about
developments or events after this date, I'll do my best to help with the information I have or
guide you on where to find the latest updates.
```

Bring Your Own Data

Limitations of AI Models

AI models:

- Are trained on public knowledge up to a certain date
- 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

Ways of using your data in AI applications:

- Train new models (❌)
- Fine tune existing models (❌)
- "Stuff the prompt" (✅)
- Retrieval Augmented Generation (✅)
- Function calling (✅)

Java and AI

Java and AI

- Why Java & AI?
  - AI becomes ubiquitous across the IT landscape
  - Java is the language of enterprise
  - Creating Java AI apps is a new requirement
- Spring AI, LangChain4j, Semantic Kernel
  - Simplify AI integration
  - Provide a unified API
  - Support multiple AI providers

Spring AI vs LangChain4j

r/SpringBoot • 3 hr. ago
Different_Rafal

Spring AI vs LangChain4J

Which of these projects do you think will dominate Spring Boot AI applications in the future?

Spring AI seems to be the obvious choice here since it is from Spring Projects. On the other hand, at conferences I saw that more companies use LangChain4J than Spring AI.

I tested both of these libraries and for my requirements they provided similar functionality. Spring AI has the advantage of being more in the spirit of Spring Boot, as are my applications.

Langchain4J seems more developed now, but I expect Spring AI to catch up. The only question is whether, if most of companies do decide to go with LangChain4J, Spring AI won't be phased out in a few years as unpopular.

What experience do you have with your companies and the materials you see? Do you foresee Spring AI dominance or not? I am wondering what would be a better choice to go with.

1 2 Share

WuhmTux • 47m ago

Does it really matter? Both are just API wrappers. The choice of LLM is more important than the API wrapper.

Vote Reply Award Share ...

Spring AI

See Spring AI on GitHub or on spring.io.

What is Spring AI?

- AI for Java developers!
- Simplifies interactions with LLMs with a Spring abstraction
  - Change model by changing one line of configuration!
- Simplifies interactions with Vector databases
- Observability
- ChatMemory
- And more ...

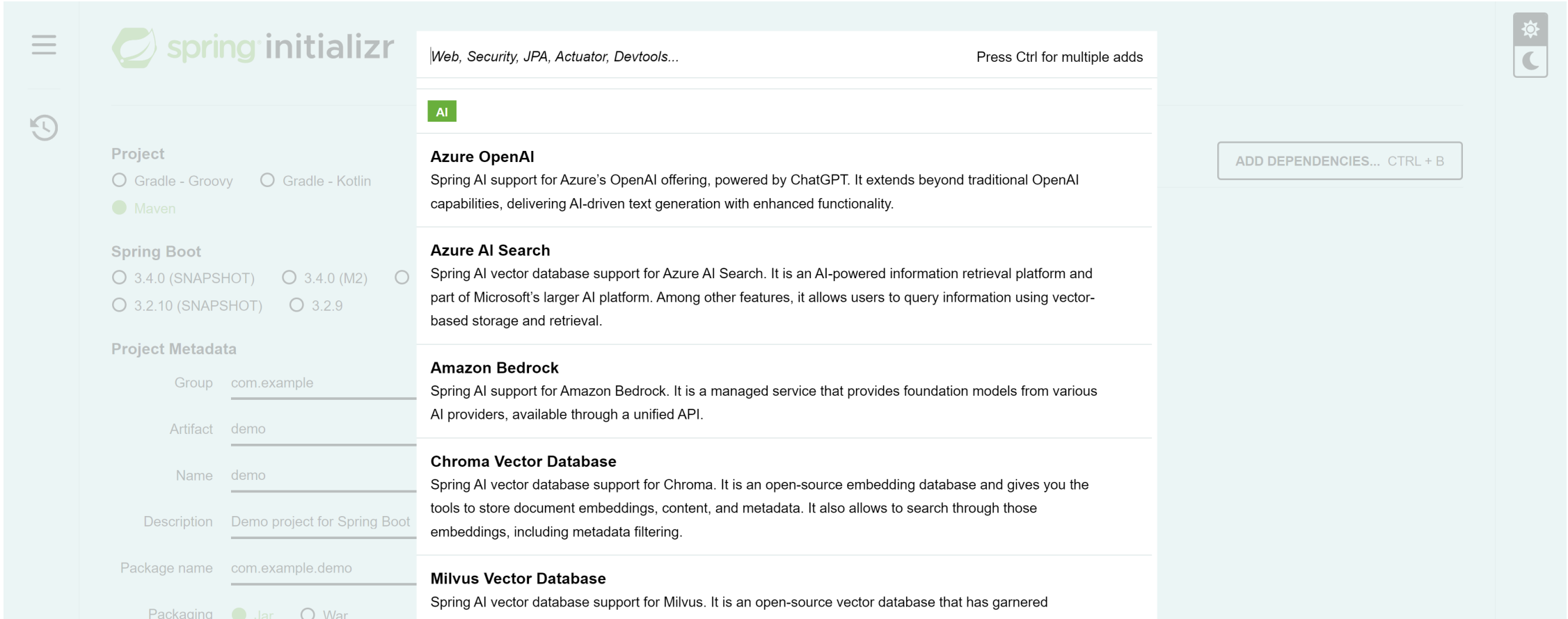
Spring AI Ecosystem

- Created in 2023 by Mark Pollack and Christian Tzolov
- Inspired by LangChain and LlamaIndex
- Current version: Spring AI 1.0.0-M3
  - Not in a final release version yet!
- Based on Spring Framework 6.2 and Spring Boot 3.3

Getting Started

Spring Initializr

Head on over to start.spring.io and select the AI Models and Vector Stores that you want to use in your new applications.



Milestone Repository

Because Spring AI is still in Milestone, you need to add the Spring Milestones repository to your `pom.xml` file.

```
<repositories>
  <repository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>https://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
```

Spring AI BOM

Add the Spring AI BOM to your `pom.xml` file and reference the version in the properties section.

```
<properties>
    <spring-ai.version>1.0.0-M3</spring-ai.version>
</properties>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.ai</groupId>
      <artifactId>spring-ai-bom</artifactId>
      <version>${spring-ai.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Spring Boot Starter

Add the specific Spring AI dependency to your `pom.xml` file.

```
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-openai-spring-boot-starter</artifactId>
</dependency>
```

Then, you can configure model parameters in the `application.properties` file as follows:

```
spring.ai.openai.api-key=${OPENAI_API_KEY}
spring.ai.openai.chat.options.model=gtp-4-turbo
...
```

Chat Client

Prompt Templates

Using the StringTemplate Engine

```
String message = """
    List 10 of the most popular YouTubers in {genre}, along with their current subscriber numbers.
    If you don't know the answer, just say "I don't know".
    """;
PromptTemplate promptTemplate = new PromptTemplate(message);
Prompt prompt = promptTemplate.create(Map.of("genre", genre));
```

```
String answer = chatClient.prompt(prompt).call().content();
```

```
ChatResponse response = chatClient.prompt(prompt).call().chatResponse();
String answer = response.getResult().getOutput().getContent();
```

Chat Client

```
Joke joke = chatClient.prompt()
  .user("Tell me a developer joke about Python.")
  .call()
  .content();
```

```
Joke joke = chatClient.prompt()
  .system("You are a friendly chatbot and you like to place emojis everywhere.")
  .user("Tell me a developer joke about Python.")
  .call()
  .entity(Joke.class);
```

```
ChatClient chatClient = chatClientBuilder
  .defaultSystem("You are a friendly chatbot and you like to place emojis everywhere.")
  .build();
```

Request / Response

Based on OpenAI chat model:

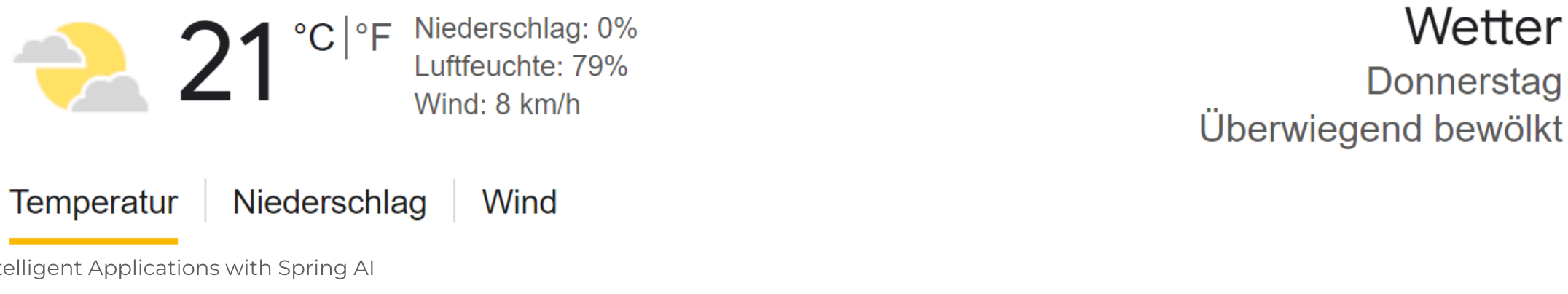
- Request
  - model
  - list of messages, each with a role and content
- Response
  - model
  - list of choices, each with a message
  - finish reason
  - usage, with the number of tokens used
  - ...

Working with Images

Working with Images

```
@Value("classpath:images/basel-weather.png")
private Resource imageResourceWeather;
```

```
chatClient.prompt()
  .user(userSpec -> userSpec
    .text("What will be the weather like on Thursday?")
    .media(MimeTypeUtils.IMAGE_PNG, this.imageResourceWeather))
  .call().content();
```



Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation

- Bring your own data to the prompt
- Give a lot of context to the prompt
  - Text content, vectors etc.

Prompt Data Takes Precedence (1)

```
You said:

How many sports were there in the Paris Olympics?
```

```
ChatGPT said:

For the 2024 Paris Olympics, there are 32 sports featured, with a total of 329 events.
New additions include breaking (breakdancing), and skateboarding, sport climbing,
and surfing are also returning after debuting in Tokyo 2020.
```

Prompt Data Takes Precedence (2)

```
You said:

Here are the sports for the Paris Olympics: Archery, Athletics, Badminton, Basketball,
Basketball 3x3, Boxing, Canoe Slalom, Canoe Sprint, Road Cycling, Track Cycling,
Mountain Biking, and BMX.

How many sports were there?
```

```
ChatGPT said:

Based on the list you provided, there are 12 sports for the Paris Olympics.
```

Prompt Stuffing Structure

```
@Value("classpath:/olympics/context.st")
private Resource queryTemplate;
```

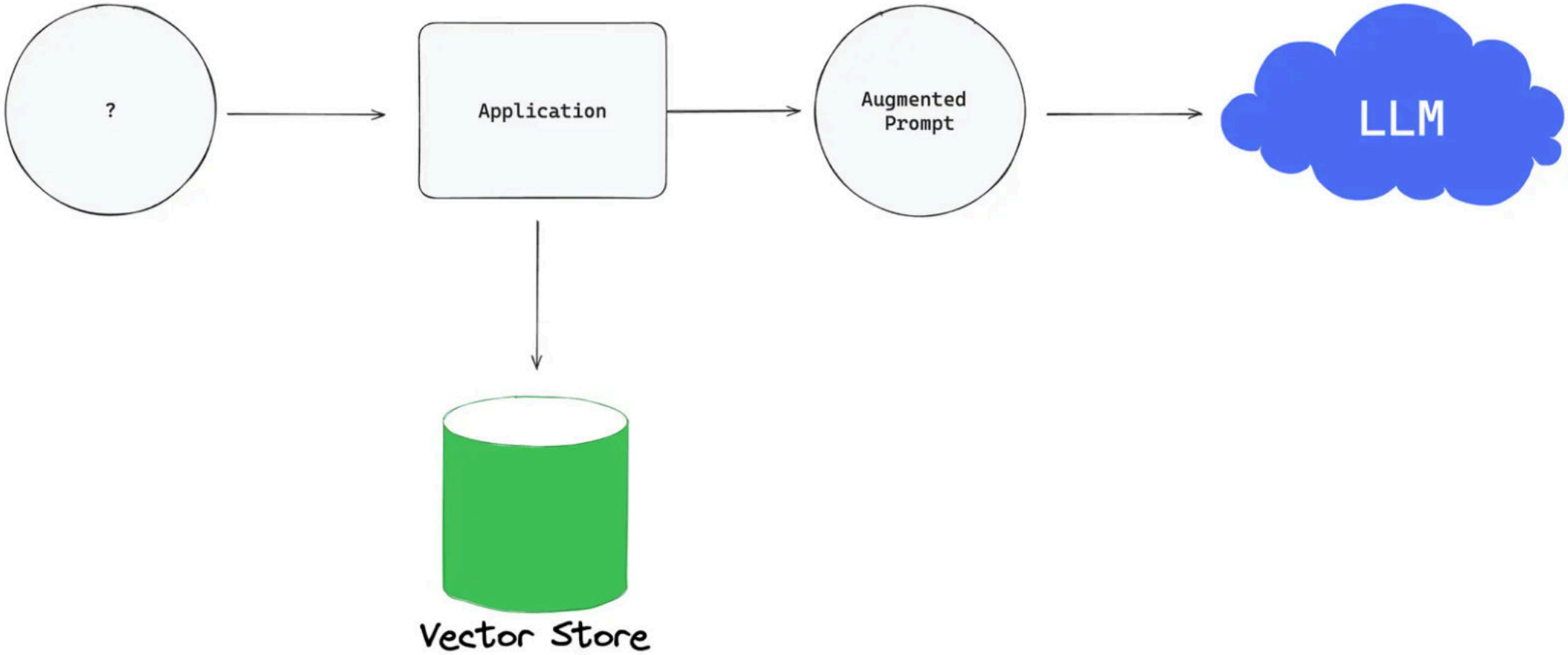
```
Use the following pieces of context to answer the question at the end.

{context}

Question: {question}
```

Using a Stuffed Prompt

```
chatClient.prompt()
  .user( userSpec -> userSpec.text(this.queryTemplate)
    .param( "context" , "Archery, athletics, badminton, basketball , boxing")
    .param( "question" ,"How many sports are being included in the 2024 Summer Olympics?")
     )
    .call().content();
```

```
Use the following pieces of context to answer the question at the end.

Archery, athletics, badminton, basketball , boxing

Question: How many sports are being included in the 2024 Summer Olympics
```

Application Architecture - RAG



Vector Stores

Vector Stores

Definition of a Vector

- A Vector is just a type of data
  - Typically an array of floating-point numbers
  - Commonly consists of 1'536 decimal values
    - OpenAI uses the ada002 embedding model
  - Each value typically ranges between -1 and 1

```
CREATE TABLE paragraph (
  id SERIAL PRIMARY KEY,
  paragraph_text TEXT,
  vector VECTOR(1536)
);
```

Example: `{-0.345, 0.465, 0.856, ..., 0.1543}`

How are Vectors created?

- Vectors typically represent the output generated by machine learning models, encapsulating the learned features or embeddings of the input data.
- The example below is based on text AI models. Vectors may also be used with computer vision AI models or audio-based AI model

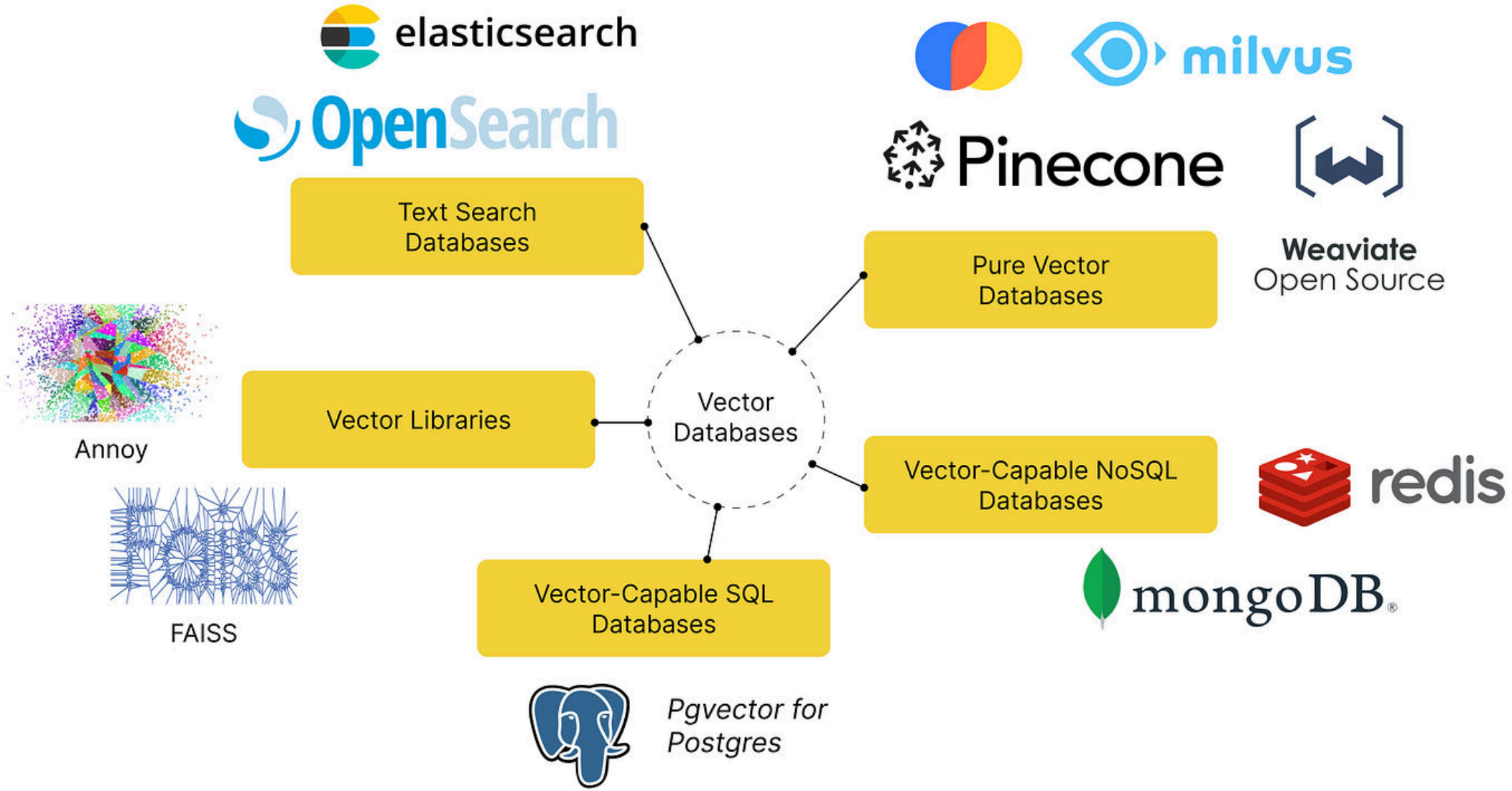
String	Vector
"My house is black"	{0.345, 0.465, 0.856, 0.1543}
"My garden is big"	{0.545, 0.665, 0.056, 0.3543}
"My dog is playful"	{0.645, 0.765, 0.156, 0.4543}

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}


elasticsearch
OpenSearch
milvus
Pinecone
Weaviate
Open Source
Text Search Databases
Pure Vector Databases
Vector Databases
Vector Libraries
Vector-Capable NoSQL Databases
Vector-Capable SQL Databases
Annoy
FAISS
redis
mongoDB.
Pgvector for Postgres

Using a Vector Store

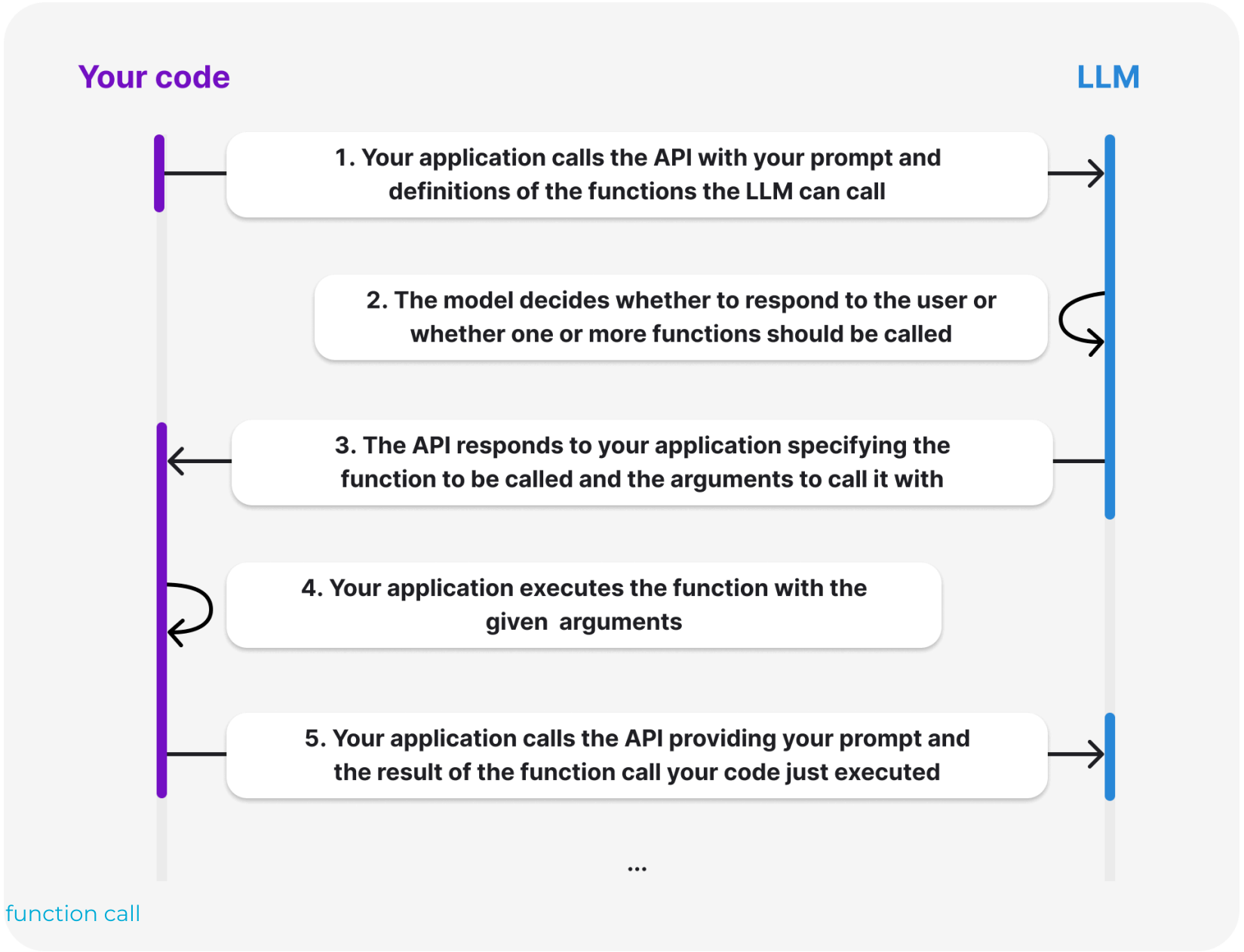
```
@Value("Artificial intelligence - Wikipedia.pdf")
private Resource pdf;
```

```
PagePdfDocumentReader reader = new PagePdfDocumentReader(pdf);
TokenTextSplitter splitter = new TokenTextSplitter();
List<Document> documents = splitter.apply(reader.get());
vectorStore.accept(documents);

String answer = chatClient.prompt()
  .system("""
      You are a virtual assistant and answers questions with the data provided.
      If you are not sure or don't know, honestly say you don't know.
      """)
  .user("Why is the definition of AI difficult?")
  .advisors(new QuestionAnswerAdvisor(vectorStore))
  .call()
  .content();

log.info("ChatGPT answered: {}", answer);
```

Function Calling

Registering Custom Functions

- The model generates a JSON object with arguments for the registered functions.
- The model does not call the function directly.
- Your code executes the function, and the result is returned to the model.
- ...



See The lifecycle of a function call

Function Calling

```
@Bean
// Request annotated with @JsonClassDescription and JsonPropertyDescription
public Function<WeatherService.Request, WeatherService.Response> weatherFunction() {
  return new WeatherService();
}
```

```
String answer = chatClient.prompt()
  .user("What's the weather like in Munich, Zurich, and New York?")
  .functions("weatherFunction")
  .call()
  .content();

log.info("ChatGPT answered: {}", answer);
```

Generative AI Challenges

Generative AI Challenges

Challenges	Patterns
Align responses to goals	System prompts
No structured output	Output converters
Not trained on your data	Prompt stuffing
Limited context size	RAG
Stateless APIs	Chat Memory
Not aware of your APIs	Function calling
Hallucinations	Evaluators

"Artificial Intelligence Will Change the World"

How AI Will Impact the Future

- Improved automation
- Job disruption
- Data privacy issues
- Ethical considerations
- Increased regulation
- Climate change concerns

Famous Last Words ...


100M
token context

We [at Magic] have recently trained our first 100M token context model: LTM-2-mini. 100M tokens equals ~10 million lines of code or ~750 novels.

https://magic.dev/blog/100m-token-context-windows

100M Token Context Windows

Research update on ultra-long context models, our partnership with Google Cloud, and new funding.

Magic Team, on August 29, 2024

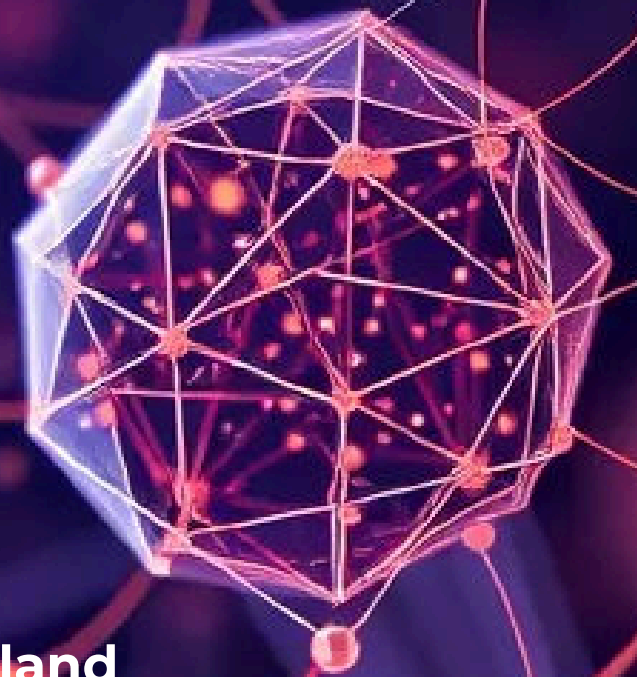
There are currently two ways for AI models to learn things: training, and in-context during inference. Until now, training has dominated, because contexts are relatively short. But ultra-long context could change that.

Instead of relying on fuzzy memorization, our LTM (Long-Term Memory) models are trained to reason on up to 100M tokens of context given to them during inference.

While the commercial applications of these ultra-long context models are plenty, at Magic we are focused on the domain of software development.

It's easy to imagine how much better code synthesis would be if models had all of your code, documentation, and libraries in context, including those not on the public internet.

Intelligent Applications With Spring AI



Patrick Baumgartner
42talents GmbH, Zürich, Switzerland

@patbaumgartner
patrick.baumgartner@42talents.com