What do AI Software Engineers do?

Introduction to agentic frameworks, llm tooling and RAG

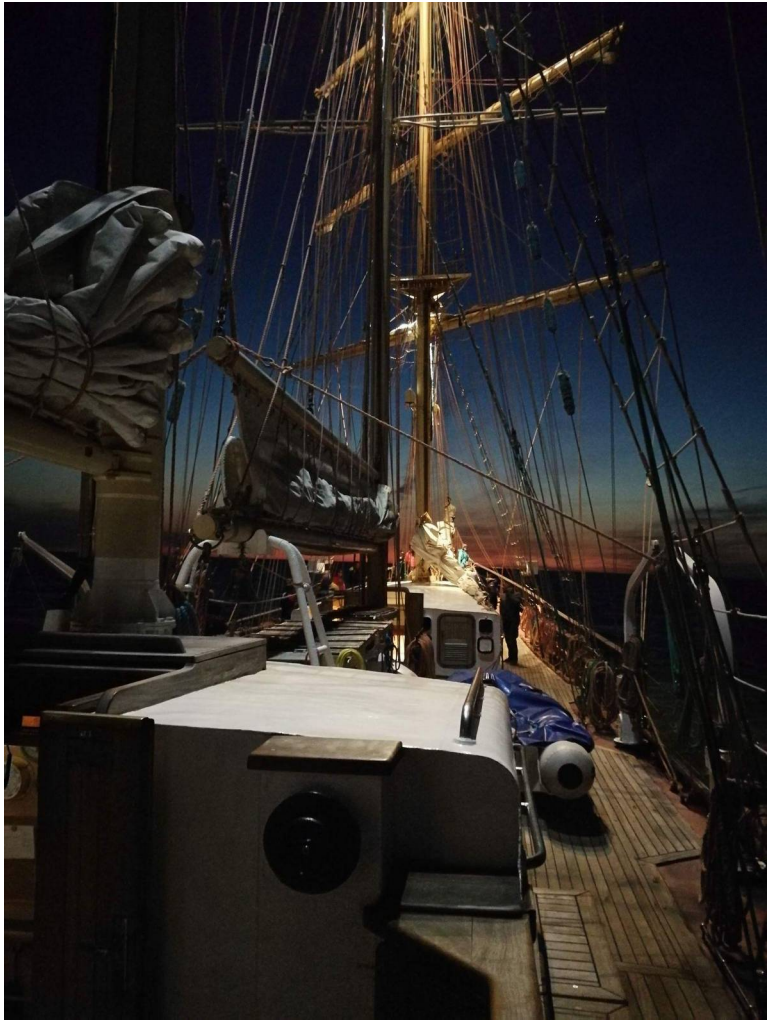
About me

Jarek Jaryszew

Software engineer since 2007

For a few years interested in GenAI and switched to related work in 2024

Currently contracting as a senior AI software engineer for big automotive (“Agentic systems developer” would fit better)



What is this presentation about

We will talk about building blocks around LLM models to create something useful.

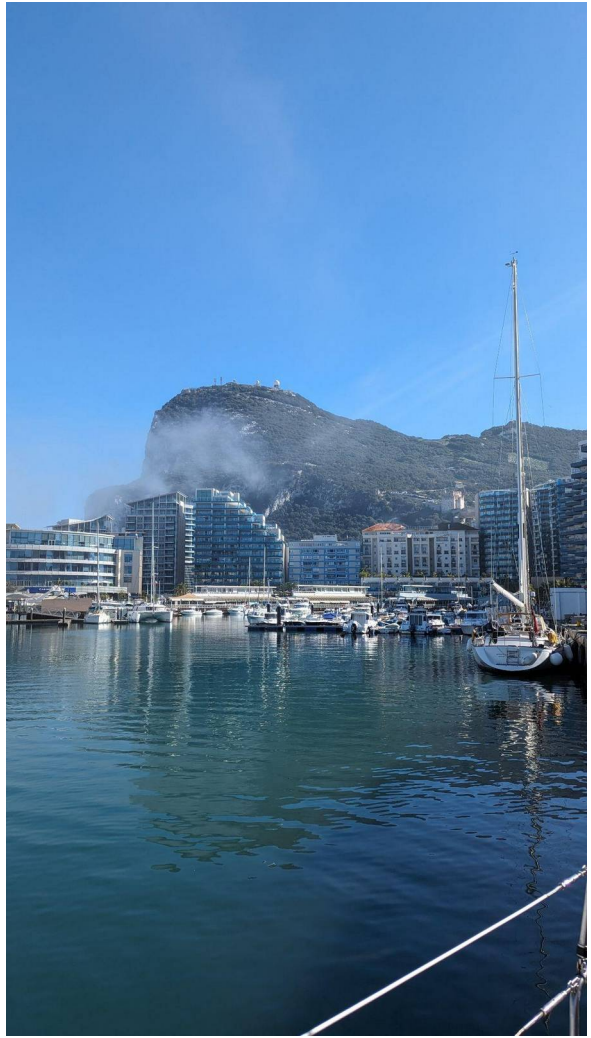
We will treat LLM model as black box so no training, distilling or fine tuning will be covered.

Demos will be based on LangChain / LangGraph framework.

I will focus on the tools and techniques I know, new ones pop up like mushrooms. If I tried to cover everything I would never finish this deck.

The agenda

- How to run models
- Agentic frameworks
  - LLM tooling and workflows
  - Multi-agent systems
- Model Context Protocol
- Resource Augmented Generation (RAG)
- Few words about quality and safety
- DEMO!



Definitions (for this presentation)

- Model is just a big binary blob, set of floating point number arrays
- Software that runs models is called inference engine
- Inference is a conclusion reached on the basis of evidence and reasoning; In ML it is the process of using a trained model to make predictions or decisions on new, unseen data.
- An agent is an application or a service using LLM model and providing it with tools, system prompt and other utilities
- A chat is a special case of an agentic system which also provide user with text-based UI
- Memory is the context given to LLM when executing a workflow

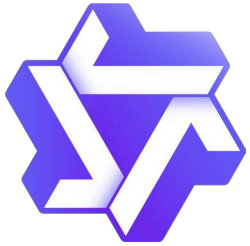
Running the models

Models - online API (MaaS)

ChatGPT

AWS Bedrock

Azure OpenAI



Online API - pricing and tokens

https://artificialanalysis.ai/leaderboards/models

- Tokens are not exactly words
- Definitions vary between providers, check documentation to be sure
- There are multiple calculators available as libraries or stand alone tools

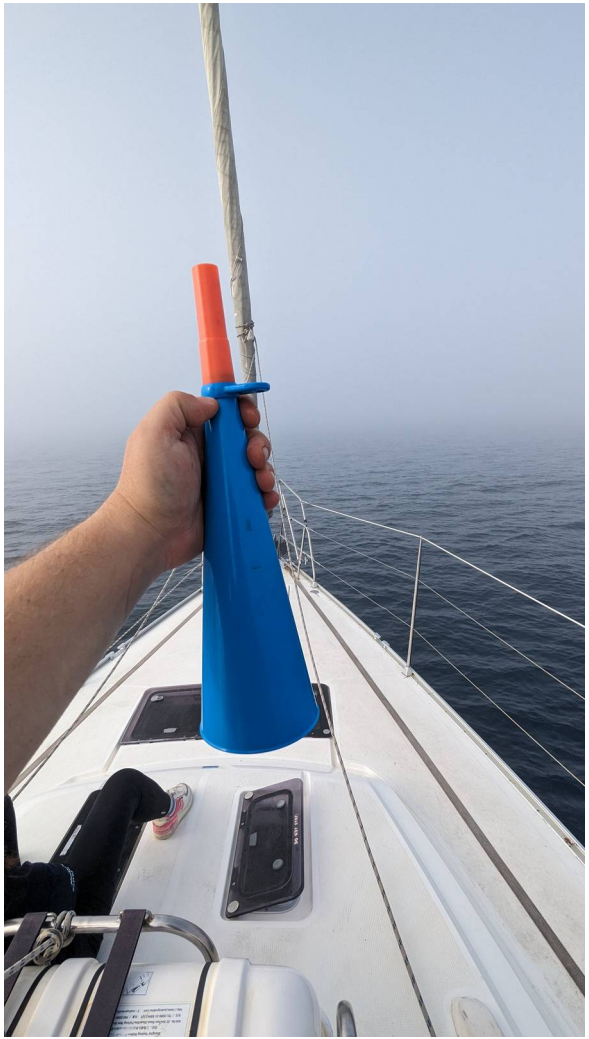
Example:

“Those are Charlie’s dogs.”

After tokenization:

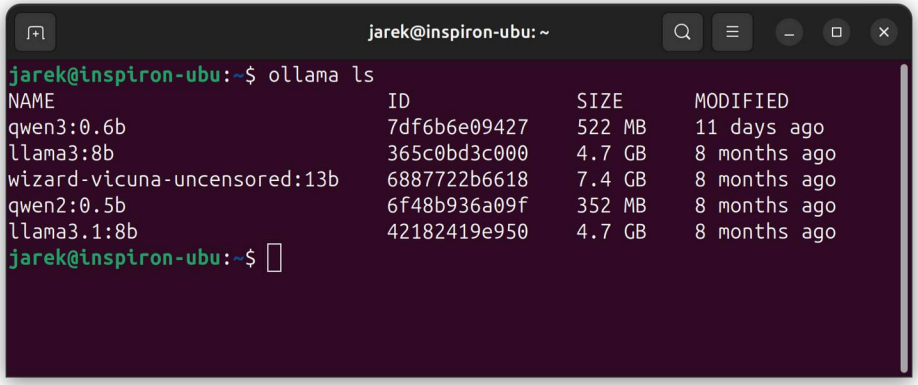
[“Those”,“are”, “Charlie”, “’s”, “dog”, “s”, “.”]

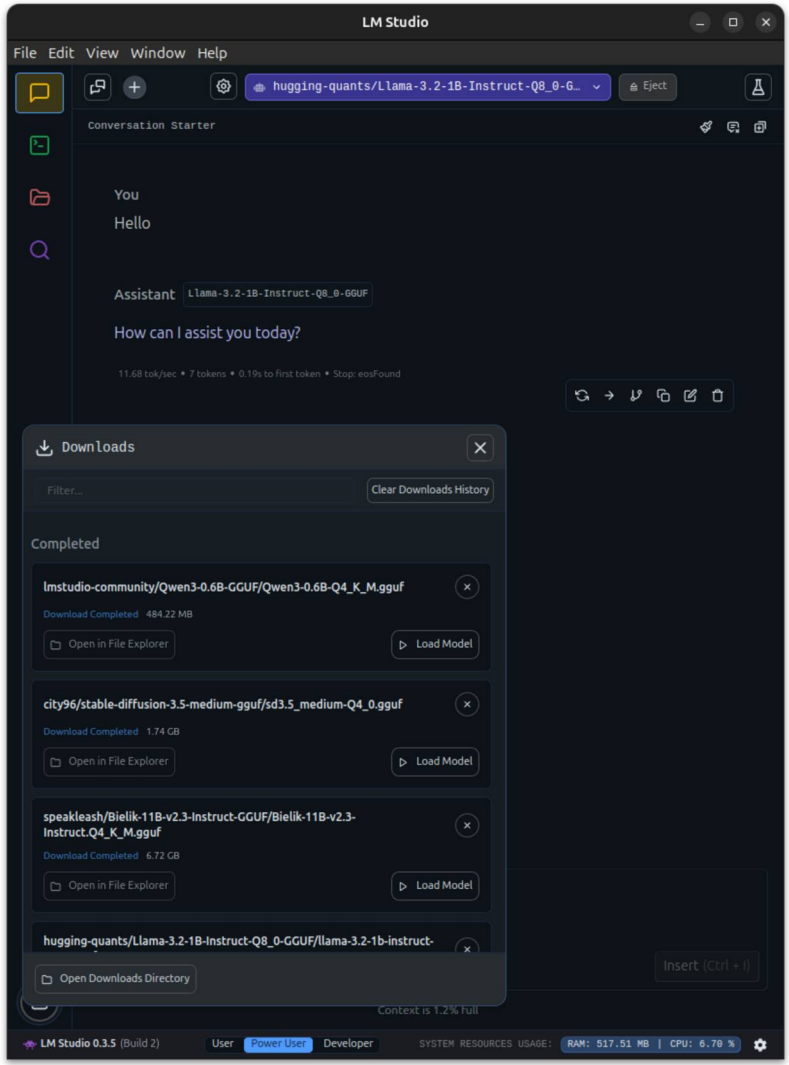
Is 1M token a lot?



Local LLM – inference engines

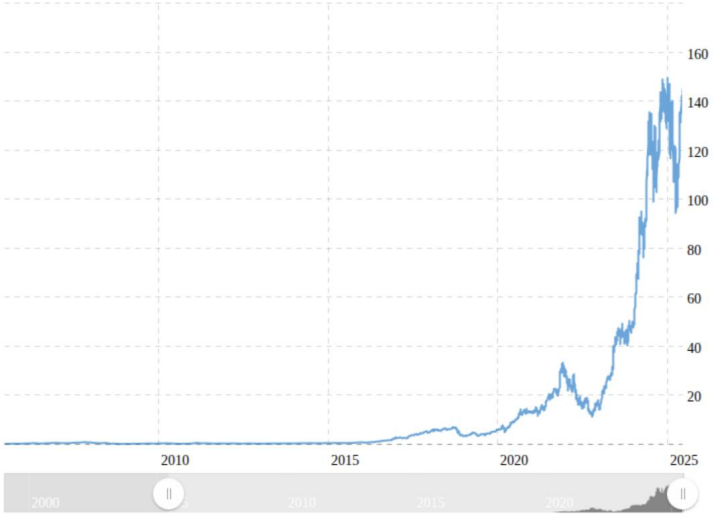
- GUI tools like LM Studio
- CLI tools like ollama
- All based on llama.cpp (thank you Meta?)
- Most provide an http interface so it can be used like MaaS



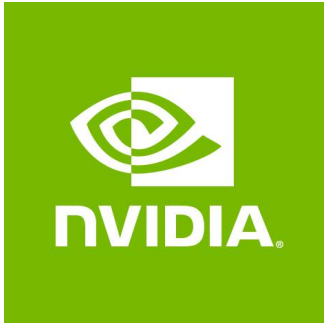


Local LLM – how to run

- Very small models (<2b) can run smoothly on a decent CPU
- Consumer level (gaming) GPUs have less then 32GB of RAM
- New server GPUs cost a lot (A100, H100, H200...)
- Second hand market? Modding?
- Table shows memory requirements in GB depending on model parameters and quantization



Quantization / Size	1b	7b	30b	300b
i4	0.6	4.28	18	180
fp8	1.2	8.4	36	360
fp16	2.4	16.8	72.8	720



Local LLM – unified / shared RAM

M4 MAX

Up to 16-core CPU
Up to 40-core GPU
Up to 128GB unified memory
Up to 546GB/s memory bandwidth
16-core Neural Engine

M3 ULTRA

Up to 32-core CPU
Up to 80-core GPU
Up to 512GB unified memory
Up to 819GB/s memory bandwidth
32-core Neural Engine

Qualcomm snapdragon

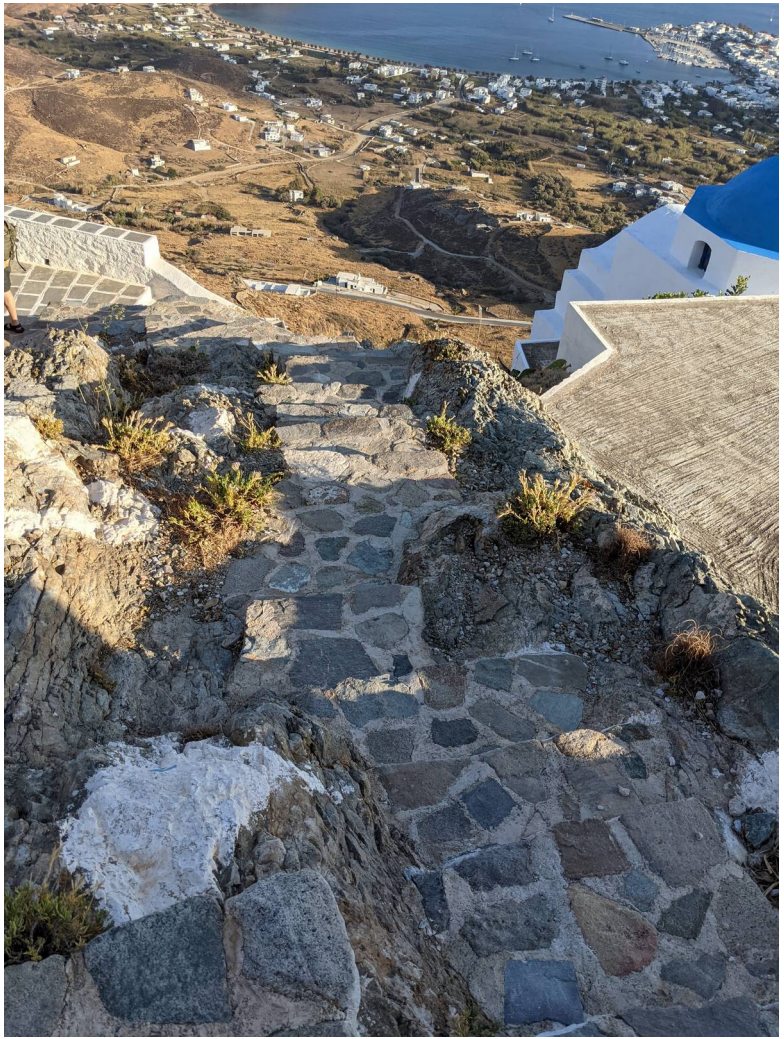
AMD Ryzen™ AI Max and
AMD Ryzen™ AI Max PRO
Series Processors

Creating a new class of AI PCs

Announcing at CES 2025

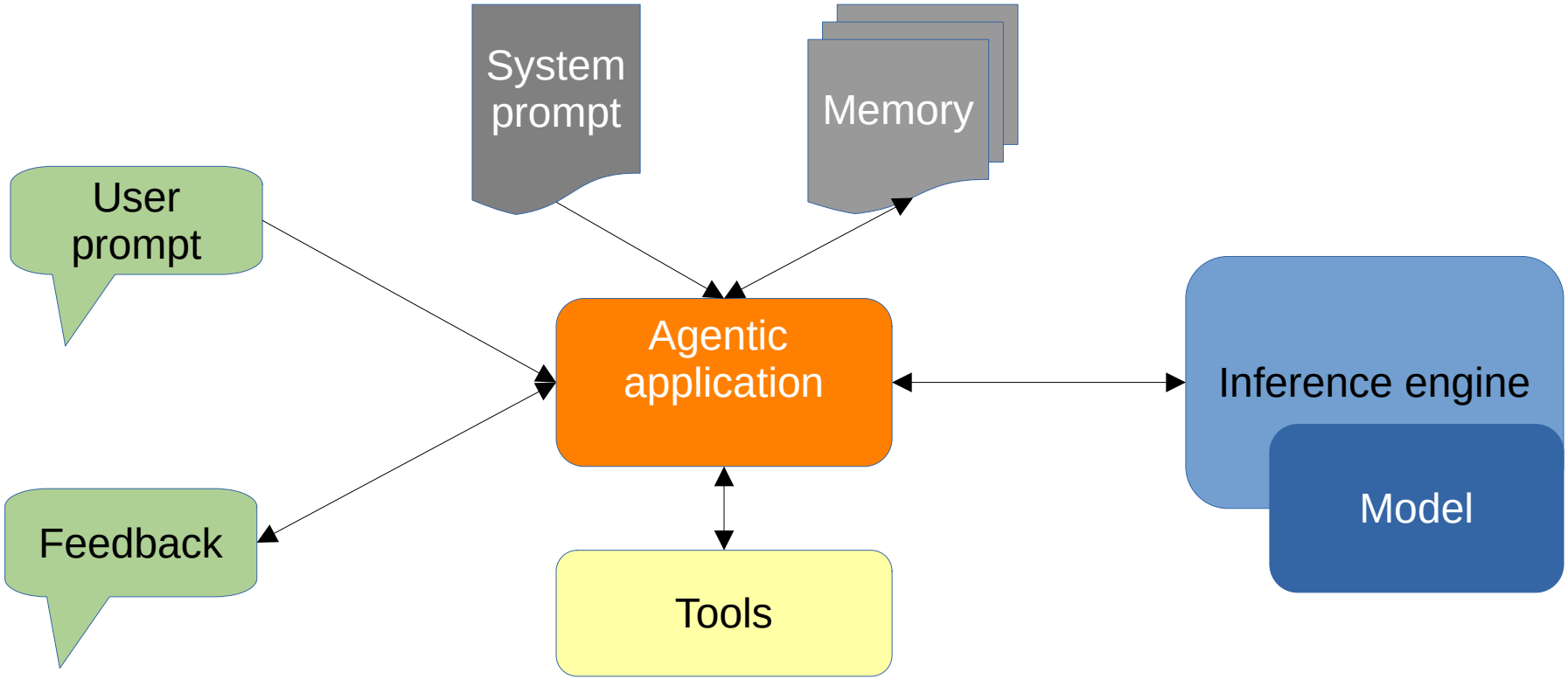
Models – which to pick?

- Very small models (0.6b – 2b) are useful for testing integrations and debugging code (up to the stage when you start looking into what is generated) computers w/o dedicated GPU
- Common opinion is that you need at least ~30b model to try do something actually meaningful
- https://ollama.com/library or https://huggingface.co/models are good places to find more info about particular models (you can sort by popularity)
- Subreddit r/localLLama is also a great place to find opinions, experiences and advice
- And there also is Bielik AI



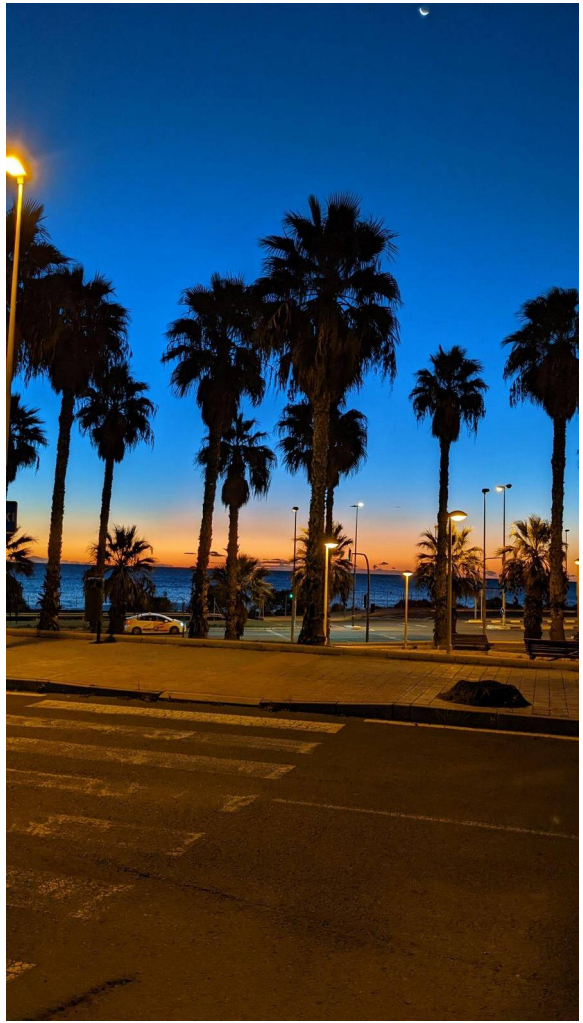
Agentic frameworks

Agentic app - common components



Why do we need a framework

- Abstraction over different LLM providers
- They provide chat workflow, history, human-in-the-loop and handle tools
- Enable multi-agent set up
- Provide standard tooling
- RAG pipeline support

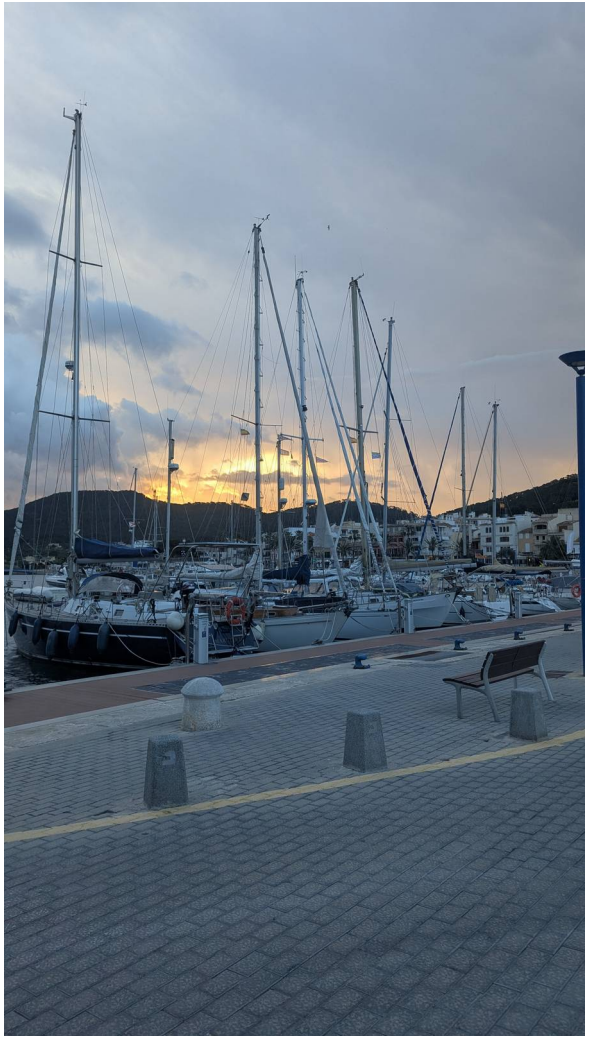


Most popular frameworks (Python*)

Semantic Kernel

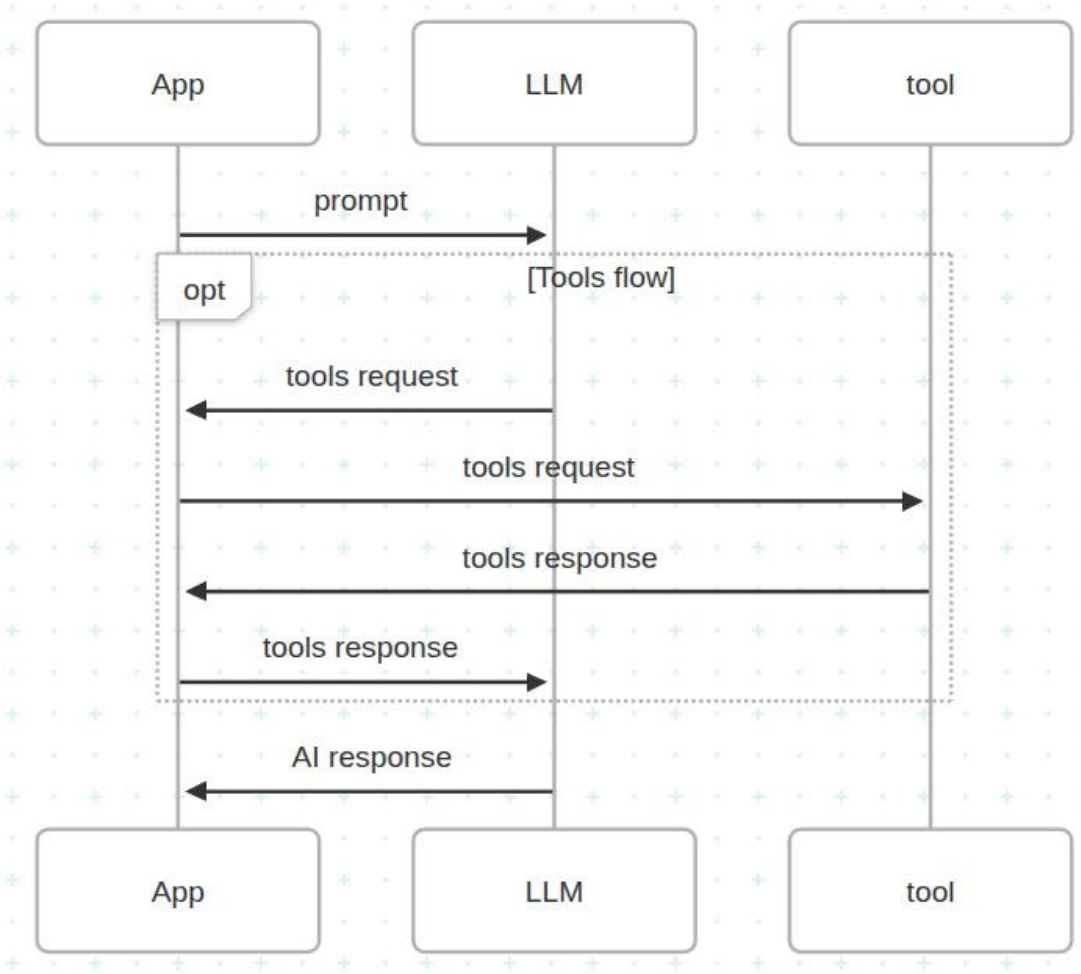
Agentic frameworks – which to choose

- LangChain and LangGraph also have TS version
- LangGraph – very precise flow control, did not start as a multi-agent framework
- CrewAI – multi-agent first. Goal and role-play oriented. Easy to set up workflows out of the box
- ...



LLM Tools

- Tools are registered for the inference
- The model may choose to use tools
- Tools are handled by the application
- A tool is a just a function with a description
- Tool may be used to fetch data or control something
- It is possible to specify tool usage policies like “always”, “optional”



LLM Tools - Examples

On the right some tools that come with CrewAI

BrowserbaseLoadTool	A tool for interacting with and extracting data from web browsers.
CodeInterpreterTool	A tool for interpreting python code.
CSVSearchTool	A RAG tool designed for searching within CSV files, tailored to handle structured data.
DirectorySearchTool	A RAG tool for searching within directories, useful for navigating through file systems.
DOCXSearchTool	A RAG tool aimed at searching within DOCX documents, ideal for processing Word files.
DirectoryReadTool	Facilitates reading and processing of directory structures and their contents.
EXASearchTool	A tool designed for performing exhaustive searches across various data sources.
FileReadTool	Enables reading and extracting data from files, supporting various file formats.
GithubSearchTool	A RAG tool for searching within GitHub repositories, useful for code and documentation search.
TXTSearchTool	A RAG tool focused on searching within text (.txt) files, suitable for unstructured data.
JSONSearchTool	A RAG tool designed for searching within JSON files, catering to structured data handling.
MDXSearchTool	A RAG tool tailored for searching within Markdown (MDX) files, useful for documentation.

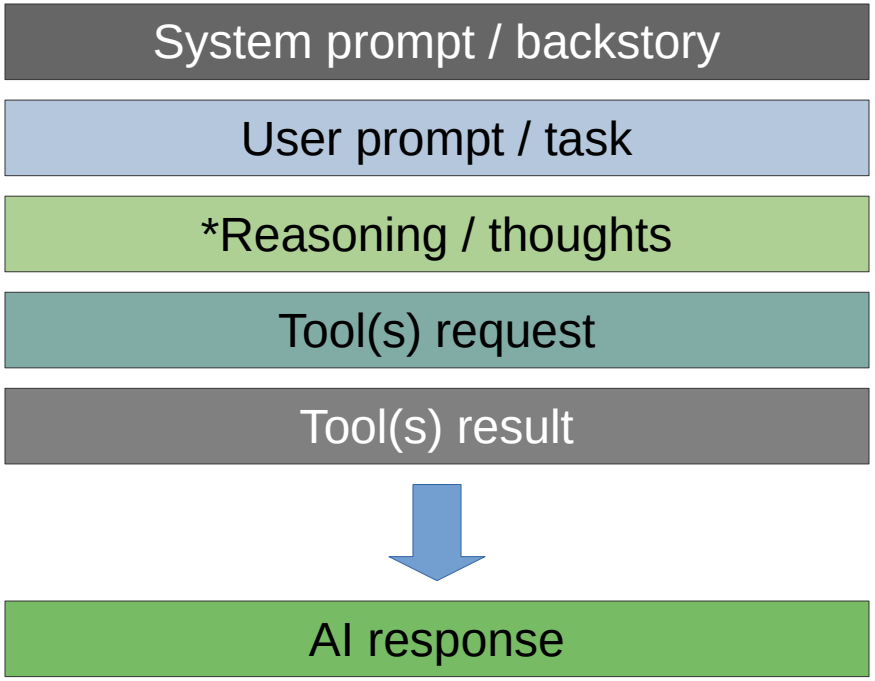
LLM Workflow

- Example shows a common basic flow
- Reasoning is optional

Grayish – programmed / system

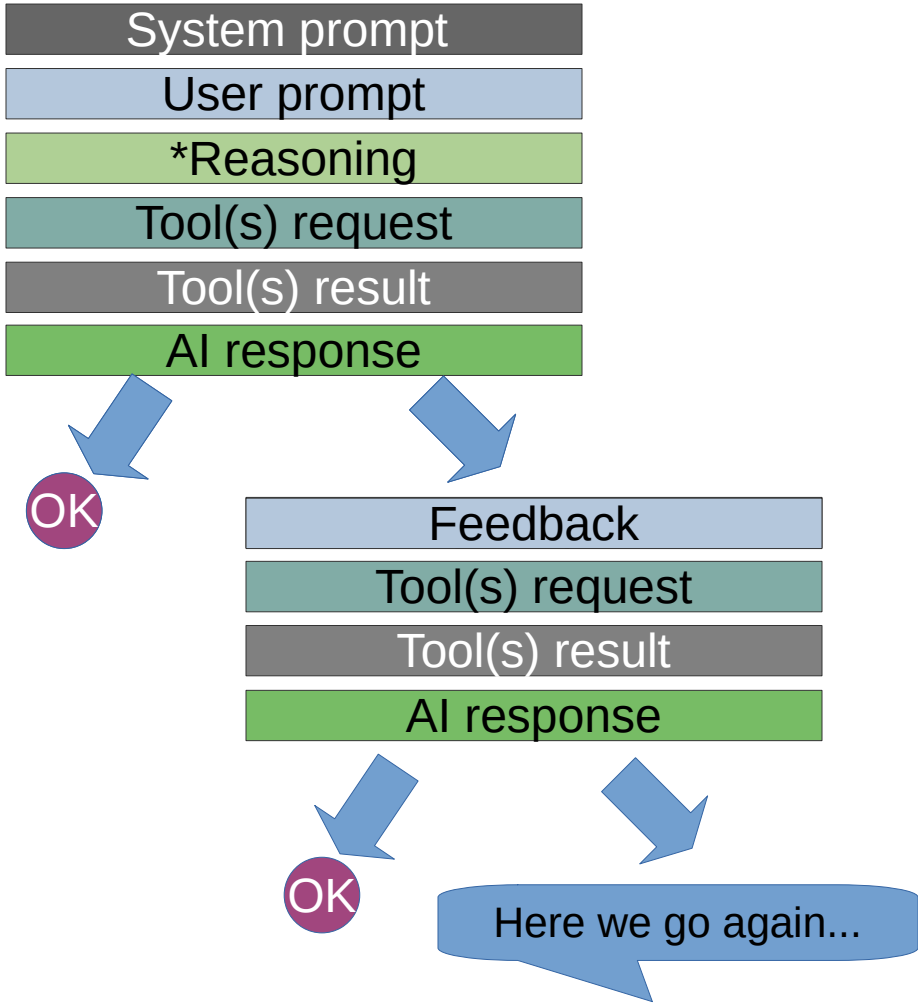
Blueish - user

Greenish - AI



Feedback / branching

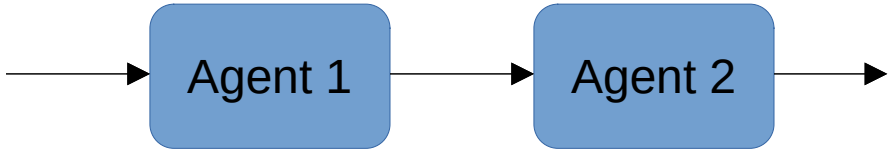
- We can iterate on the results
- Manual check and critique – human-in-loop
- Automated check (for example compilation errors)
- Other branching possibilities depending on tool results or LLM decisions
- And the context window grows...



Multi agent systems

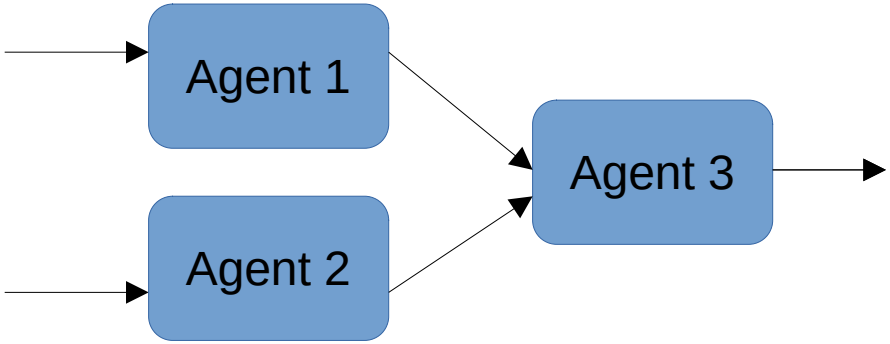
- Divide and conquer
- Simpler prompts – less hallucinations
- Smaller context window if you don’t choose to share the context window between agents
- Splitting job into tasks makes it easier to debug and monitor

Strategy: Chain agents



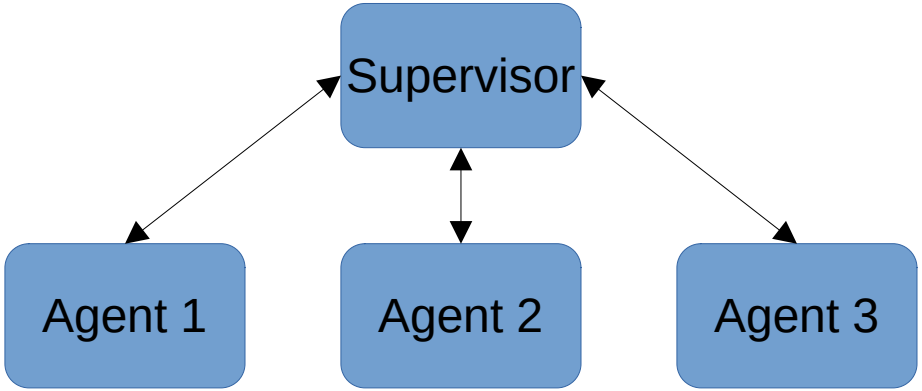
- Different tasks when chatting with user (agent 1 confirm id and agent 2 performs the interview)
- Agent 2 may do summary of agent 1 interactions

Strategy: Merge flows



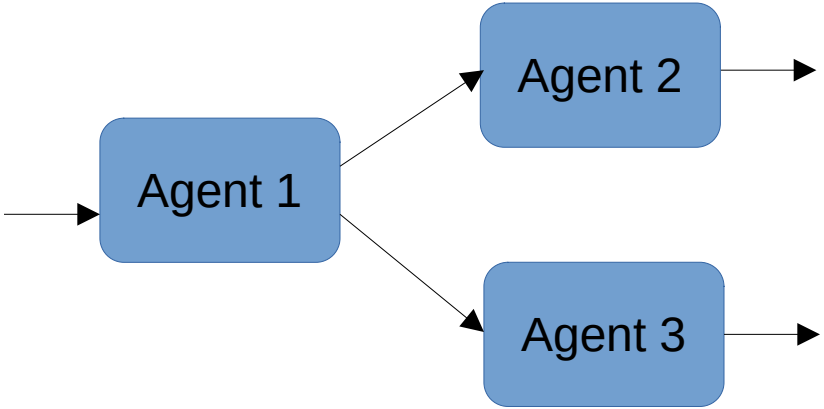
- Useful when compiling data from very different knowledge bases
- Agents 1 and 2 provide a summary of findings to the agent 3

Strategy: Supervisor agent



- Just let the supervisor agent decide which agents to employ to perform the task

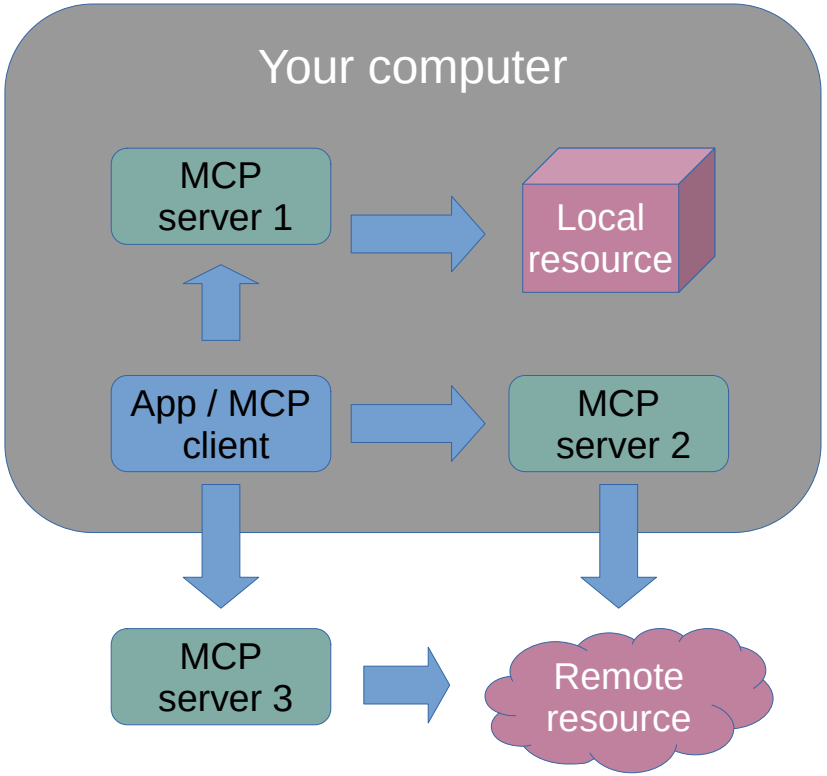
Strategy: Diverge flows



- Similar to supervisor, but the agent 1 is used to classify the user input not to respond
- Flow is later dispatched to one or the subsequent agents (or none)
- We can use that to prevent an agent from receiving request not related to it’s setup

Model Context Protocol

- Open sourced by Anthropic in Q4 2024
- LLM tool interface standardization
- Tool definition and implementation on the server side
- MCP server may be online – pass url and credentials
- MCP server may be local – pass a startup command, args and env
- Transport layers:
  - STDIO
  - Streamable HTTP with SSE



MCP

- MCP divides server contents into categories
- Protocol is under rapid development
- Multitude of servers, clients and libraries
- Microsoft announced exposing many Windows functionalities via MCP

Primitive	Description
Resources	Resources are a core primitive in the Model Context Protocol (MCP) that allow servers to expose data and content that can be read by clients and used as context for LLM interactions.
Tools	Tools are a powerful primitive in the Model Context Protocol (MCP) that enable servers to expose executable functionality to clients. Through tools, LLMs can interact with external systems, perform computations, and take actions in the real world.
Prompts	Prompts enable servers to define reusable prompt templates and workflows that clients can easily surface to users and LLMs. They provide a powerful way to standardize and share common LLM interactions.

Resource Augmented Generation

So what’s the problem?

- “Find me all the documents on this or similar subject”
- Word search?
- What about synonyms and homonyms?
- Similar subject??
- Wasn’t that problem solved before (internet search engines)?

Clustering to the rescue

- So before LLMs data scientists came up with clustering
- It also works for NLP problems
- Actual results are in a multidimensional space
- Each piece of text processed by the model gets a vector representation
  (0-1 scale)
- Do not confuse with classification
- Single dimensions are not useful to us. The difference (distance) between vectors matter.

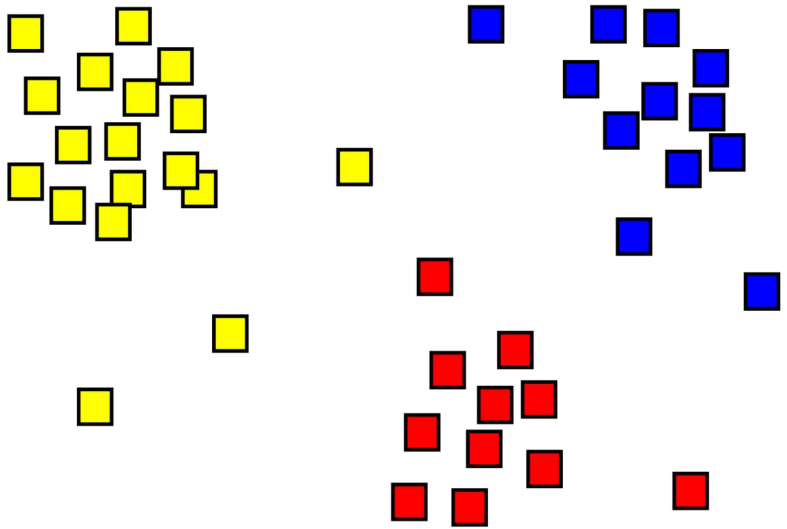


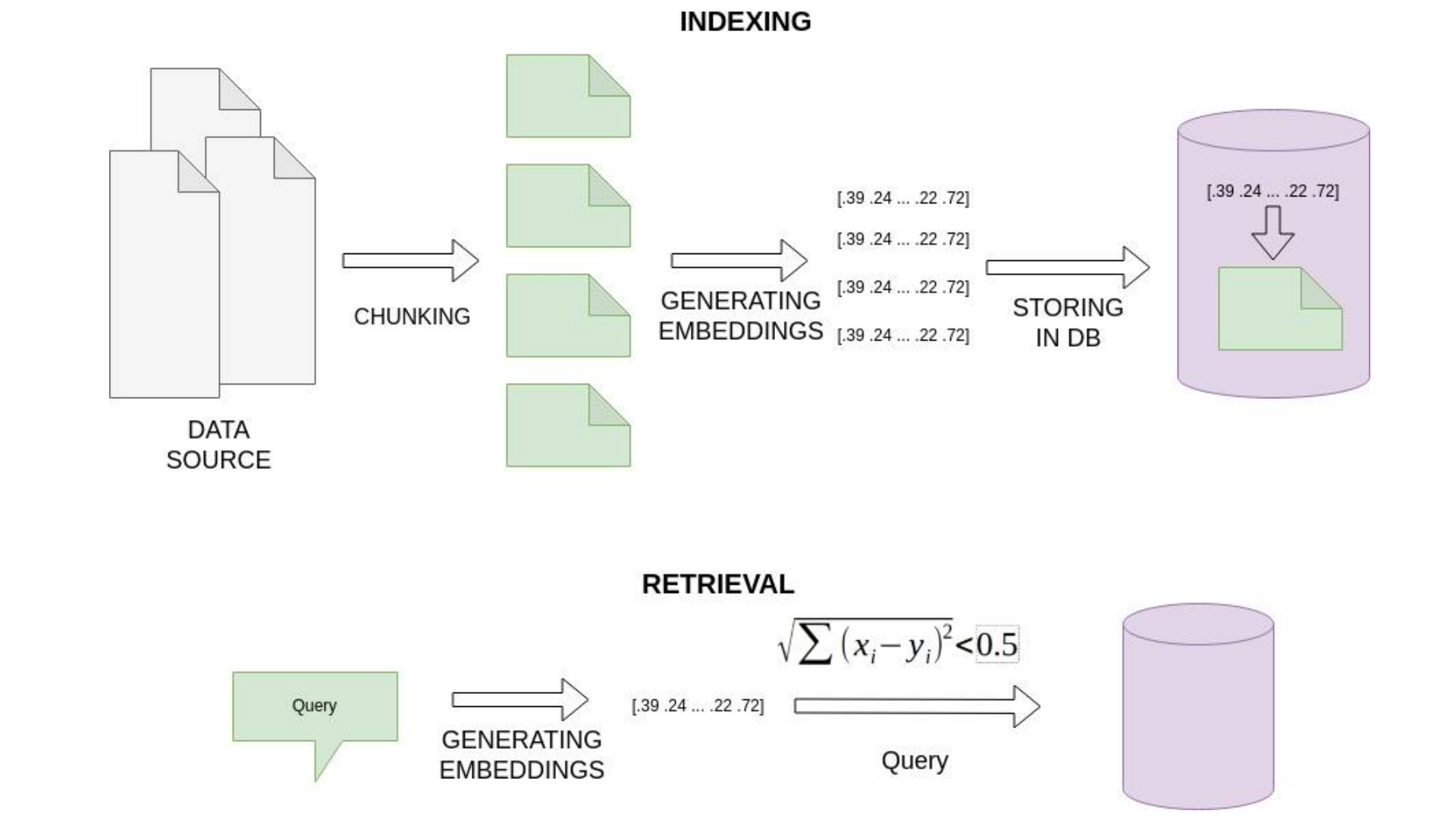
*Image from : https://en.wikipedia.org/wiki/Cluster_analysis

Embeddings model

- Unsupervised training requires a massive amount of data
- There are ready to use universal models available
- So how many dimensions does an ML model need to cluster the reality? :)
- Those are not LLM models. Inference needs much less resources!
- You have to use the same model version for indexing and retrieval!

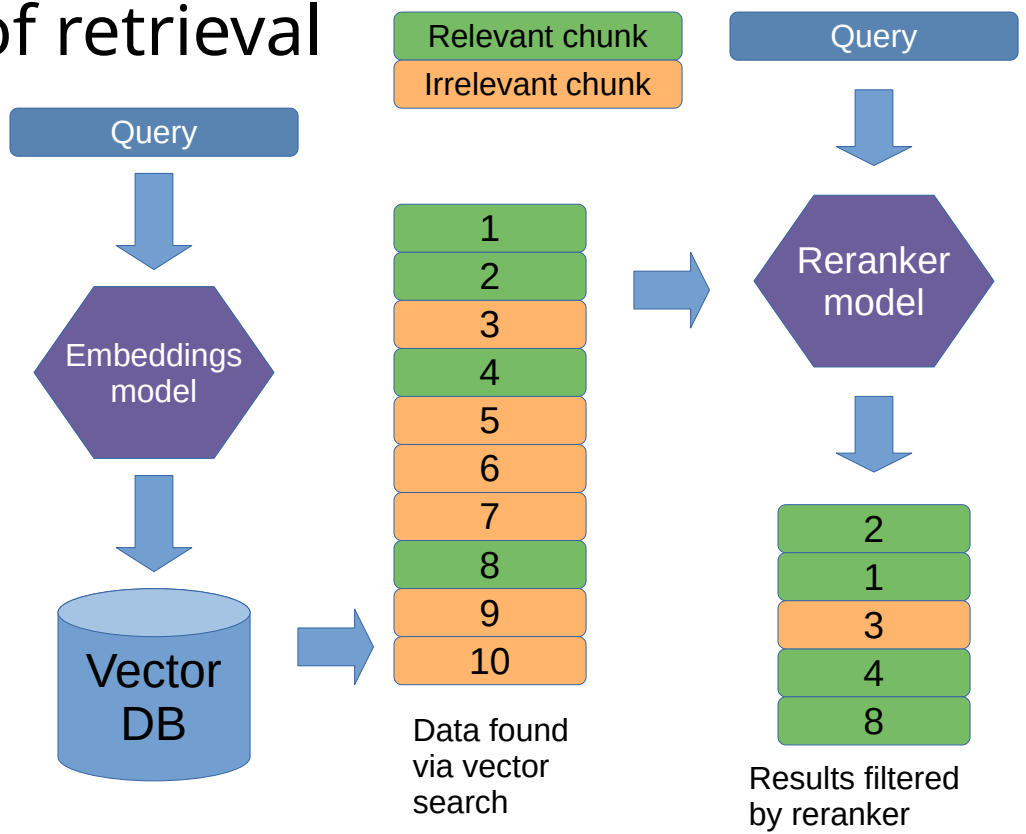
*Data from: https://www.datastax.com/blog/best-embedding-models-information-retrieval-2025

Model	Output dimensions	Licence
OpenAI text-embedding 3-large	3072	Proprietary
OpenAI text-embedding 3-large	1536	Proprietary
Gemini text-embedding	768	Proprietary
Cohere Embed v3	1024	Proprietary
ModernBERT Embed Base	768	Apache 2
ModernBERT	1024	Apache 2
Jina Embeddings v3	1024	Cc-by-nc-4.0
Stella 400M v5	1024	MIT
Stella 1.5B v5	1024	MIT


INDEXING
DATA SOURCE
CHUNKING
GENERATING EMBEDDINGS
[.39 .24 ... .22 .72]
[.39 .24 ... .22 .72]
[.39 .24 ... .22 .72]
[.39 .24 ... .22 .72]
STORING IN DB
[.39 .24 ... .22 .72]
RETRIEVAL
Query
GENERATING EMBEDDINGS
[.39 .24 ... .22 .72]
$\sqrt{\sum (x_i - y_i)^2} < 0.5$
Query

Rerankers - 2nd stage of retrieval

- Embedding models are far from perfect
- Each chunk may contain many pieces of information, how to represent it as one vector?
- Reranker models cross-encode query and chunk against each other to verify similarity
- Reranker may be based on LLM or not
- Same as with embeddings there are open source models and MaaS APIs



Vector DBs

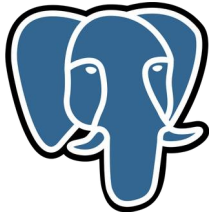
- Vector DB adds only a new type of index!
- Allows least distance vector search
- Several new DBMS has emerged on the RAG hype
- Meanwhile all the major players have introduced vector search in their products

Chroma

Pinecone

VS

elastic



Data chunking

- Embeddings models have limited input size (8k tokens for example)
- Split into pieces that make sense
- You may enrich the data if you think it may help
- Structuring data is a good idea too
- Keep different types of data in different tables (like source code and documentation) to avoid bias.
- There some libraries for chunking popular data formats (like Langchain Text Splitters)

RAG pipelines

- These are just data workflows
- Decide if you need to index the knowledge on a regular basis
- Decide if you need a workflow orchestration like Apache Airflow, Luigi or Prefect
- Or maybe just run it as a job in your CI/CD service
- Or use whatever you already have in your organization

Apache Airflow

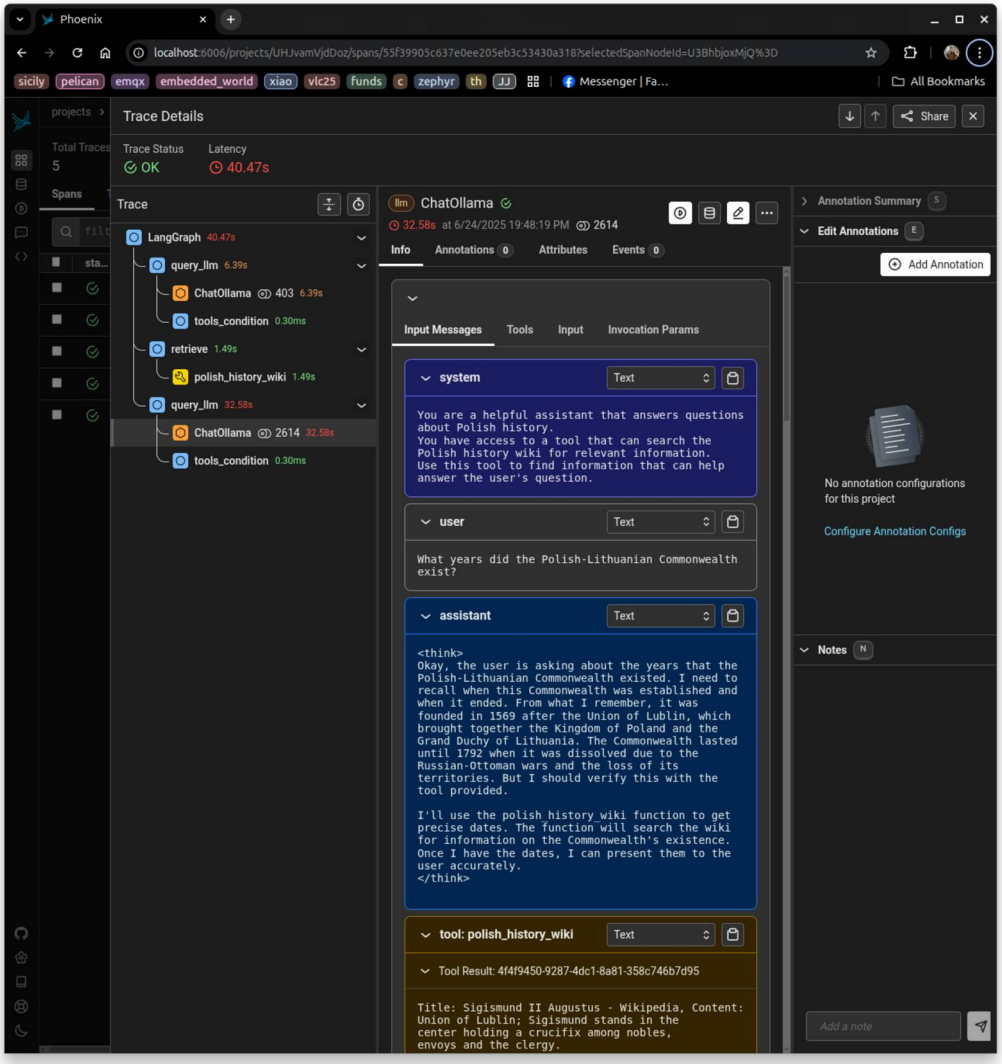
PREFECT

GitLab

Quality

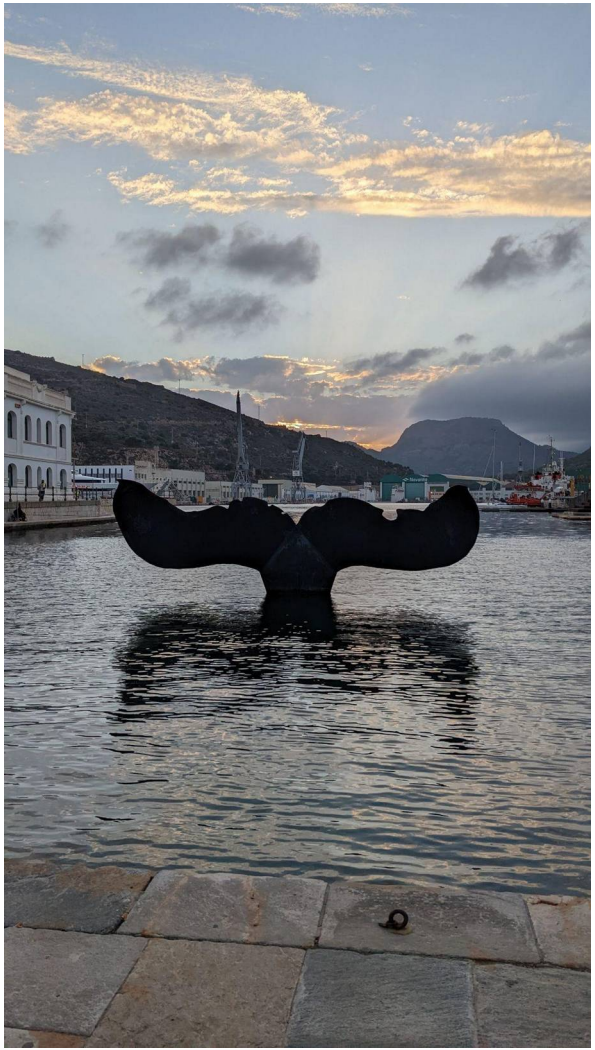
Monitoring

- Open Telemetry / Open Inference
- Arize Phoenix
- LangSmith for LangChain/Graph



Security

- Don’t do authorization/authentication on the llm level
- If your chat is interacting with users use guardrails:
  - Inappropriate content and offensive language filters
  - Prompt injection protection
  - Sensitive data scanning
- Some popular tools:
  - https://www.guardrailsai.com/
  - https://www.nvidia.com/en-us/ai-data-science/products/nemo/
  - All major cloud providers have their own set of guardrail tools
- You usually have to configure the guardrails yourself



How to tackle hallucinations

- Experiment with prompts – asking explicitly that model says “I cannot deliver it with current data” may actually help
- Make sure that you cover all corner cases in your prompt
- Try splitting job between agents
- Try changing parameters responsible for randomness (like temperature for ChatGPT)
- Feedback loop (human in the loop or automated if possible)
- Check the data you provide with tools
- Saying ‘please’ in the system prompt does not help