Context Engineering and Agentic AI development

Enrico Zimuel - Tech Lead & Principal Software Engineer

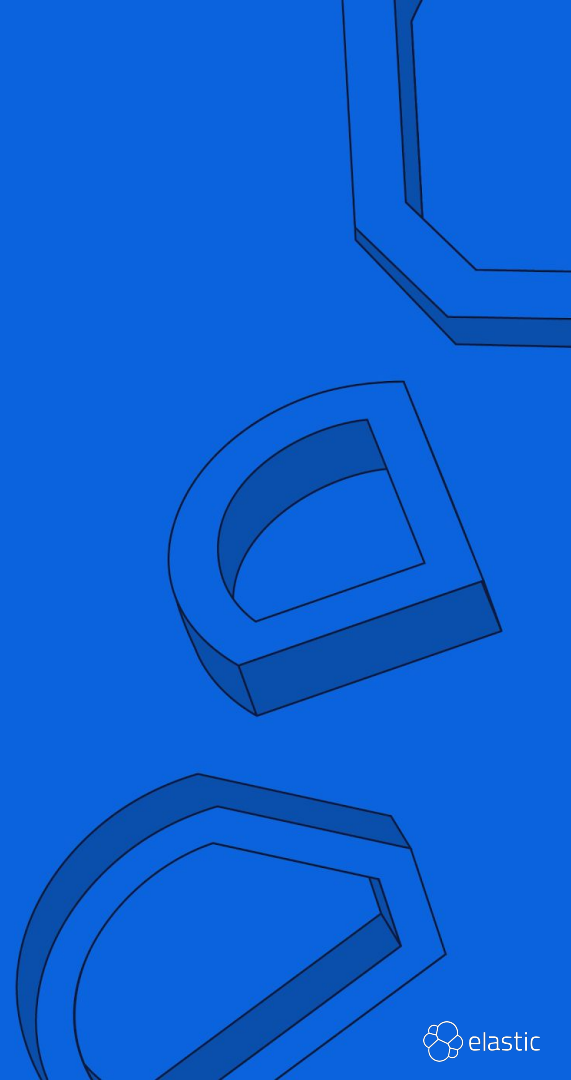
elastic | The Search AI Company

CloudConf

Agenda

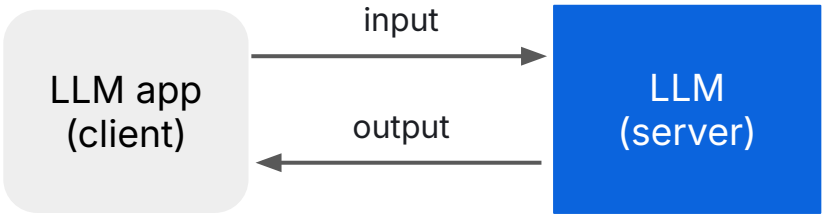
- Context Engineering
- LLM properties:
  - In-Context Learning
  - Tool calling
- Agentic AI applications
- It’s all about search

Context Engineering



LLM apps

- An application that uses an LLM is a **client** that interacts with a backend service
- Send an input request (**prompt**) and obtains an output response
- The size of the prompt is called **context window**
- It's measured in **token**, about ¾ of an English word
- The cost ($) of an interaction is **token input + token output**

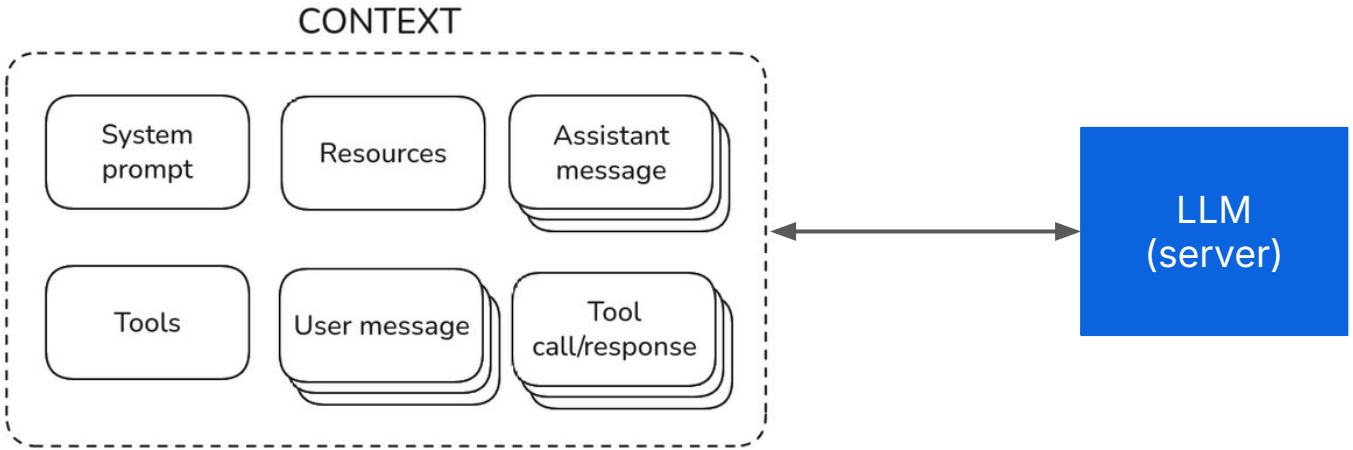


Context Window

- The management of the **context window** is a key part of an AI application
- For instance:
  - In a chat application is crucial to remember previous conversations (**memory**)
  - In a RAG architecture, the most important part is the **accuracy** of the context. If the retrieved information is pertinent to the query, the answer will be accurate
  - In an Agentic AI application the context is the **router of the flow**

Context Engineering

- **Context Engineering** is the emerging discipline of designing, managing, and optimizing the entire **informational environment** (instructions, retrieved data, tools, and history) that a Large Language Model (LLM) uses to make decisions



In-Context Learning

- **In-Context Learning** is an “emergent” property of LLMs that allows them to “learn” the pattern of a task from the context they are given and apply it immediately
- It allows an LLM to adapt to a task using only examples or instructions provided in the prompt, without updating its model weights
- In-context learning is more like **understanding** and using the context provided in the prompt without permanently learning from it

Example

- **Prompt**
  - *Assign each request to one category: Billing, Technical Support, or Sales.*
    Request: I want to upgrade my subscription.
    Category: Sales
    Request: My app keeps crashing.
    Category: Technical Support
    Request: I was charged twice this month.
    Category: Billing
    Request: I need help resetting my password.
    Category:
- **Response**
  - *Technical Support*

Tool calling (or function call)

- Tool calling is the **ability of LLM to recognize the need to execute external functions** (tools) as part of its reasoning process
- An LLM can determine whether a request requires invoking an external tool, referred to in the prompt as a resource (tool)
- If it does, the LLM returns the tool's name and the parameters to include in the function call
- The client—rather than the LLM—is responsible for executing the function, and this step is typically overseen by a human
- The function's result is then provided back to the LLM as context, enabling it to generate the final answer

Tool calling in OpenAI

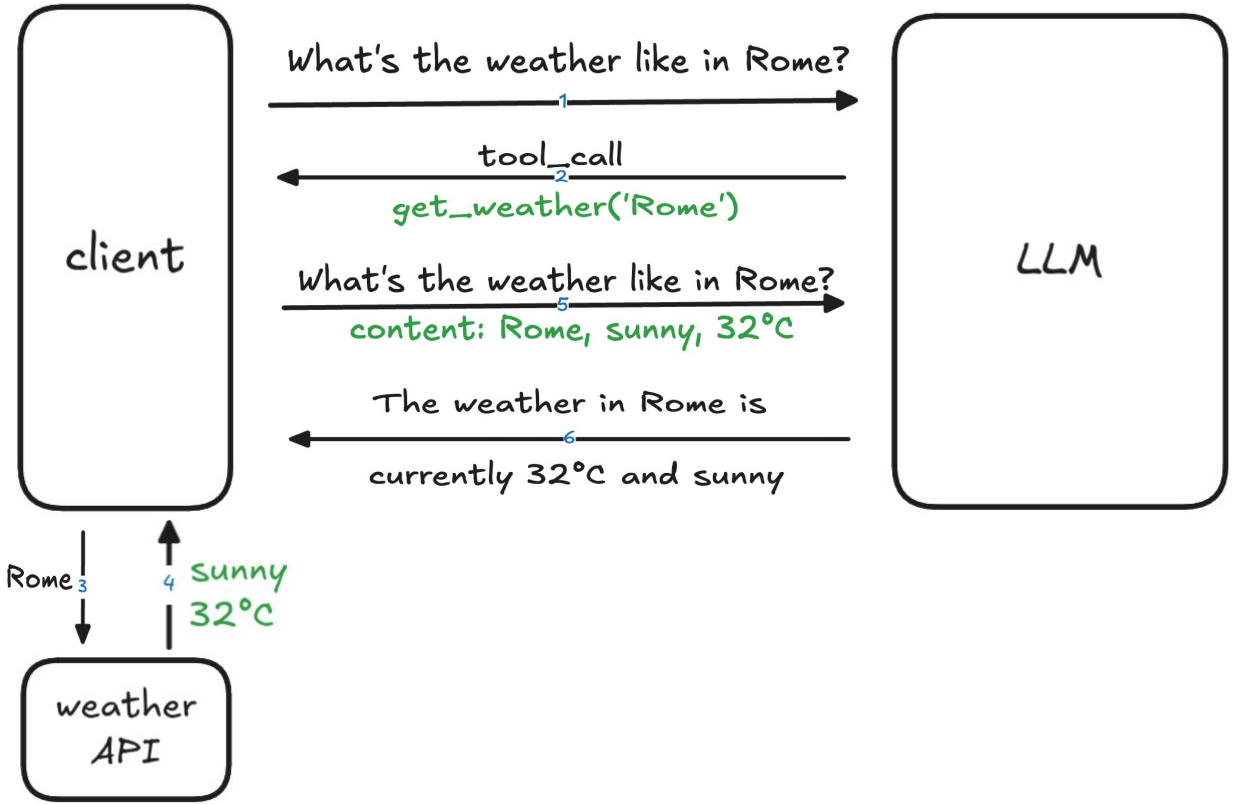
POST https://api.openai.com/v1/chat/completions

```
{
  "model": "gpt-4.1",
  "messages": [
    {
      "role": "user",
      "content": "What is the weather like in Rome today?"
    }
  ],
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "get_weather",
        "description": "Get current temperature for a given location.",
        "parameters": {
          "type": "object",
          "properties": {
            "location": {
              "type": "string",
              "description": "City and country e.g. Rome, Italy"
            }
          },
          "required": [
            "location"
          ],
        },
      }
    }
  ]
}
```

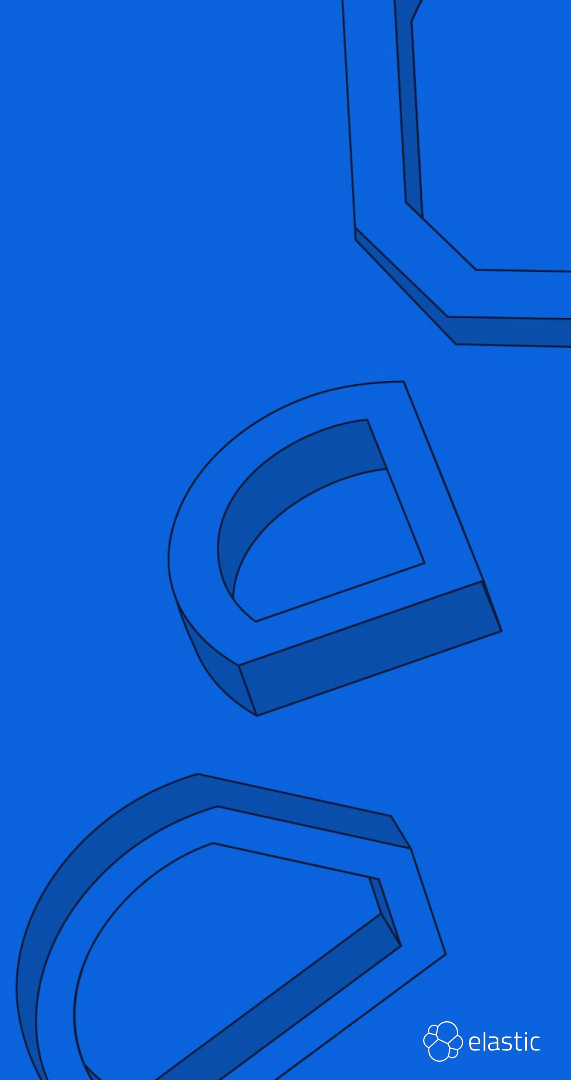
Response

```
[{
  "id": "call_12345xyz",
  "type": "function",
  "function": {
    "name": "get_weather",
    "arguments": "{\"location\":\"Rome, Italy\"}"
  }
}]
```

A diagram of Tool calling



Agentic AI



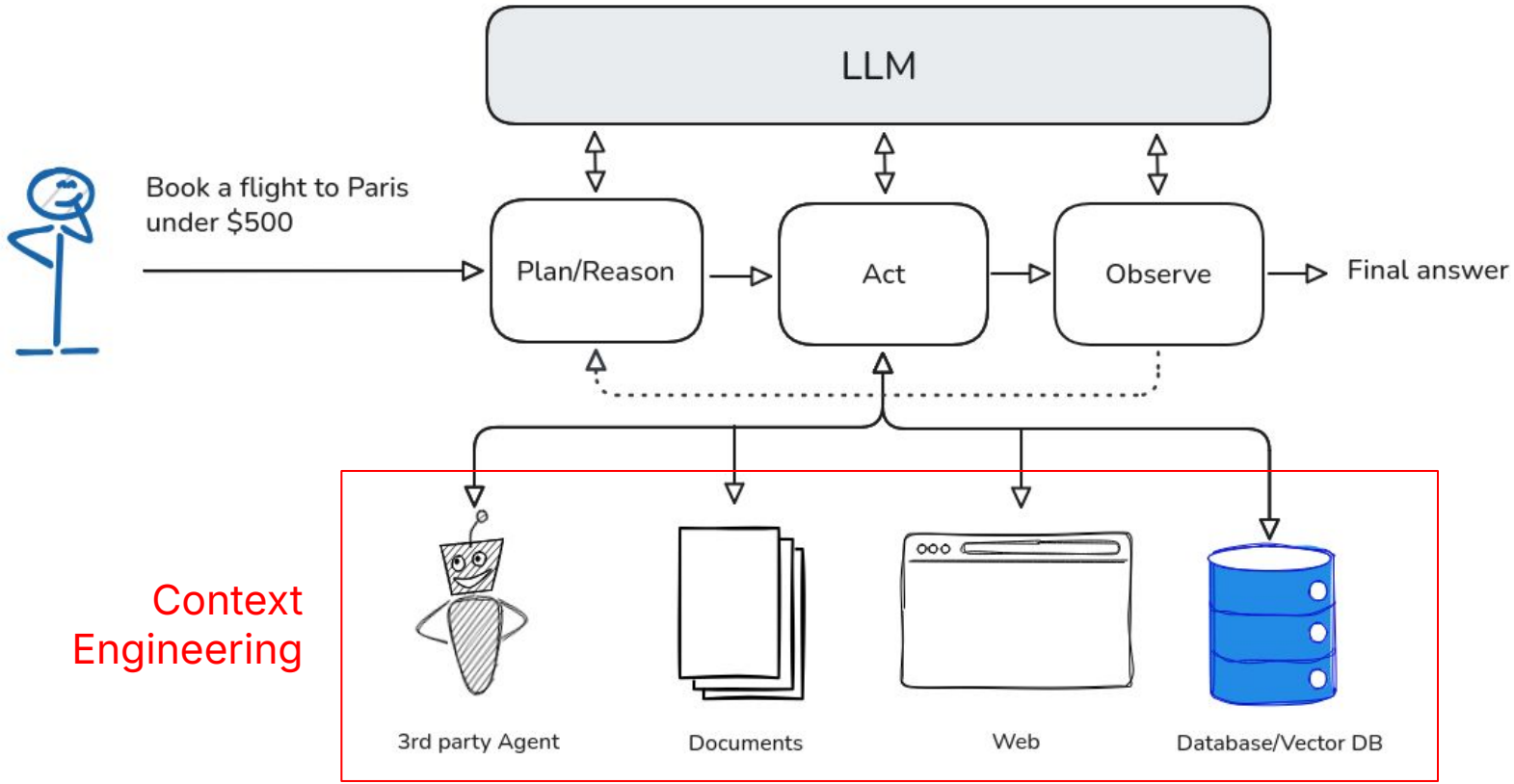
Agentic AI

- **Agentic AI** refers to AI systems that can act autonomously to achieve goals, making decisions and taking actions without direct human intervention
- An **agent** is usually a software that makes decisions and takes actions often by calling tools or APIs in a loop, based on its current understanding of the world and its objective
- Key features of an agent:
  - **Goal-oriented**
  - **Autonomous**
  - **Interactive**
  - **Iterative**: reason → act → observe → repeat
  - **Memory** (optional)

Agent workflow example

- **Goal:** Book a flight to Paris under $500.
  - **Plan (reason)**: “Check available flights.”
  - **Act**: Calls search_flights(to="Paris", max_price=500)
  - **Observe**: Gets a list of flights.
  - **Reason**: “There’s a flight on Tuesday under budget. Book it!”
  - **Act**: Calls book_flight(flight_id)
  - **Finish**: Reports the result back to the user.
- The agent monitors its own process, deciding what to do next based on each result

Agent workflow diagram

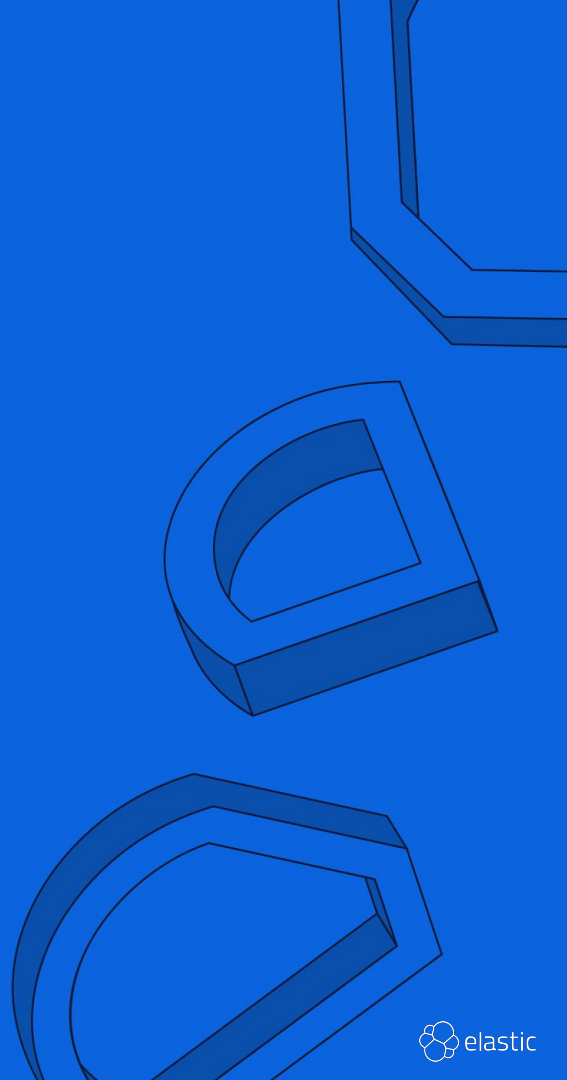


Limitations

- **Execution time:** not really fast since it requires many steps
- **Expensive**: agents typically use about 4x more tokens than chat interactions, multi-agent use about 15x more tokens than chat
- **Complexity**: multi-agent are good with parallel tasks but not so good to manage many dependencies between agents*
- **Unpredictability**: agent interactions can lead to unexpected outcomes. Needs of Guardrail systems.
- **Evaluation**: it's difficult to measure the collective success or alignment of multiple agents

* How we built our multi-agent research system, Anthropic

Searching for context



Search for AI

- In Context Engineering we need to manage multiple source to retrieve the information for a specific LLM session
- Elasticsearch is a NoSQL distributed datastore, a powerful search and analytics engine and a vector database
- It’s released with 3 licenses:
  - AGPLv3 open source license
  - Server Side Public License, v1
  - Elastic license

Try Elasticsearch

- Local (using start-local script):
  - `curl -fsSL https://elastic.co/start-local | sh`
- In Cloud (free trial):
  - https://www.elastic.co/cloud

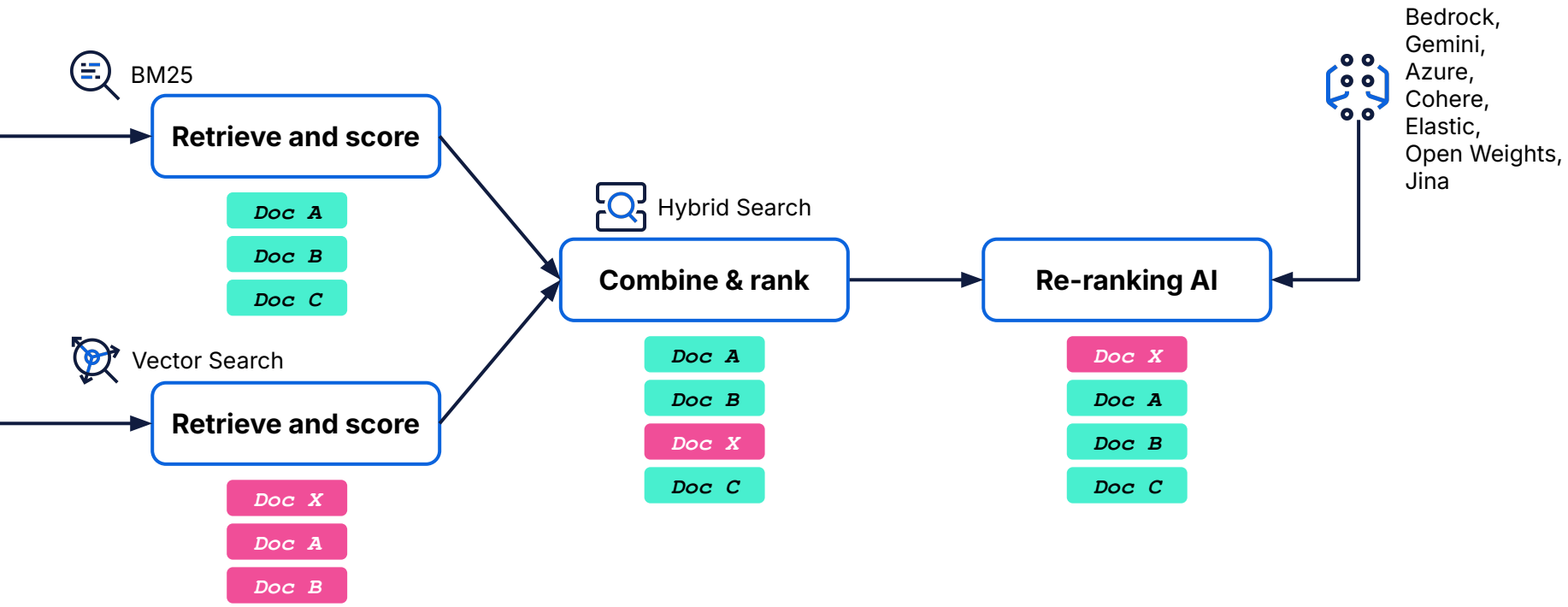
From raw to searchable data

- Transforming, enriching, and restructuring data during ingestion help prepare raw data into clean, searchable, and LLM-friendly context
- This improves retrieval quality, reduces noise, and makes the generated answers more accurate and grounded
- Elasticsearch offers:
  - Analyzer, tokenizer, token filter, normalizer
  - Example: Normalization, Lemming, Stop Words, parse data
  - Support of over 30 languages
  - Scripting (painless)

It’s all about Search

- **Lexical search** (BM25), matches exact terms between the query and documents using inverted indexes
- **Vector search** (HNSW), converts text into embeddings and finds documents with similar meaning
- **Hybrid search**, combines Lexical + Vector search to improve the relevance of the context
- **Reranking**, reorder the results using a relevance model, such as a semantic reranker or cross-encoder

Bringing it all together



References

- What is Context Engineering?, Elastic web site
- Tomás Murúa, Context engineering vs. prompt engineering, Elastic blog
- Shane Connelly, Practical BM25 - Part 1: How Shards Affect Relevance Scoring in Elasticsearch, Elastic blog
- Shane Connelly, Practical BM25 — Part 2: The BM25 Algorithm and its variables, Elastic blog
- Florian Bernd, Enrico Zimuel, First to hybrid search: with Elasticsearch and Semantic Kernel, Elastic Search labs
- Huage Chen, Yazid Akadiri, Evaluating RAG systems, EAH 2025 presentation
- Panagiotis Bailis, Hybrid search revisited: introducing the linear retriever!, Elastic Search labs
- Adam Demjen, Nick Chow, Semantic reranking in Elasticsearch with retrievers, Elastic Search labs
- Enrico Zimuel, Software Engineering in the Age of Generative AI, 2025
- Enrico Zimuel, Tools calling in Agentic AI, 2025

Thanks!

Contact: enrico.zimuel@elastic.co