

Mastering Generative AI

*From Word Embeddings to
Custom AI Agents*



TJC & DefineX

Bilge Yücel



Hi! 

-  Developer Relations Engineer at deepset
-  Open source LLM Framework: Haystack
-  Istanbul, Turkey

Bilge Yücel

Twitter: @bilgeycl

LinkedIn: [Bilge Yucel](#)

GitHub: [@bilgeyucel](#)

NLP?

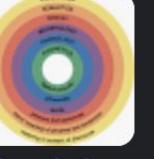
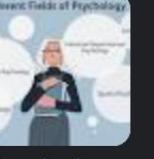
NLP?

Neuro-linguistic programming (Nörolinguistik Programlama)

[!\[\]\(4222901cb0b23d3b20e48e0aa550c263_img.jpg\)](#)

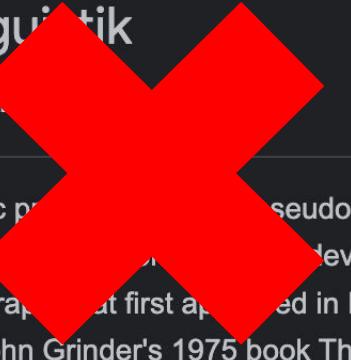
Neuro-linguistic programming is a pseudoscientific approach to communication, personal development and psychotherapy, that first appeared in Richard Bandler and John Grinder's 1975 book The Structure ... [Wikipedia](#)

People also search for [View 5+ more](#)

-  [Communication](#)
-  [Coaching](#)
-  [Linguistics](#)
-  [Psychology](#)

NLP?

Neuro-linguistic programming (Nörolinguistik Programmierung)

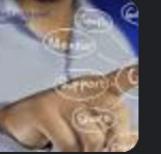
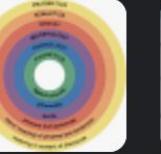


Neuro-linguistic programming (NLP) is a pseudoscientific approach to communication, personal development and psychotherapy that first appeared in Richard Bandler and John Grinder's 1975 book The Structure of Magic.

[... Wikipedia](#)

People also search for

View 5+ more

Communication Coaching Linguistics Psychology



Neuro-Linguistic Programming

NLP - Natural Language Processing

Natural language processing (Doğal dil işleme)



Natural language processing is an interdisciplinary subfield of computer science and linguistics. It is primarily concerned with giving computers the ability to support and manipulate human language. [Wikipedia](#)

People also search for

-  Artificial intelligence
-  Machine learning
-  Data science
-  Deep learning

View 10+ more



Neuro-Linguistic Programming

Natural Language Processing (Doğal Dil İşleme)

Agenda

- 01 - Text Embeddings
- 02 - Vector Databases
- 03 - Retrieval
- 04 - LLMs
- 05 - RAG
- 06 - Agents & Function Calling



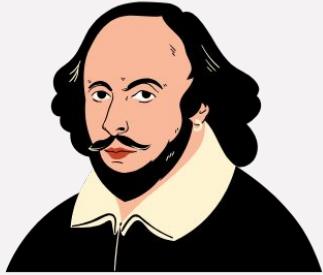
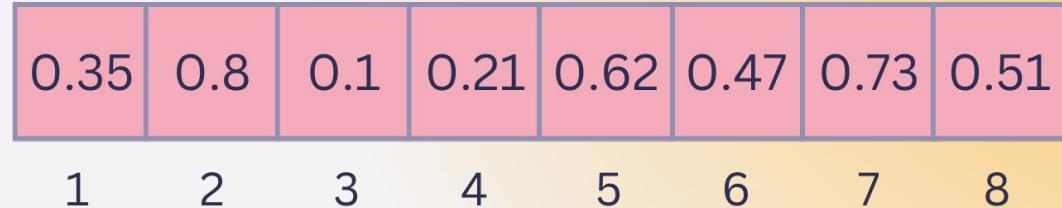
01

Text Embeddings



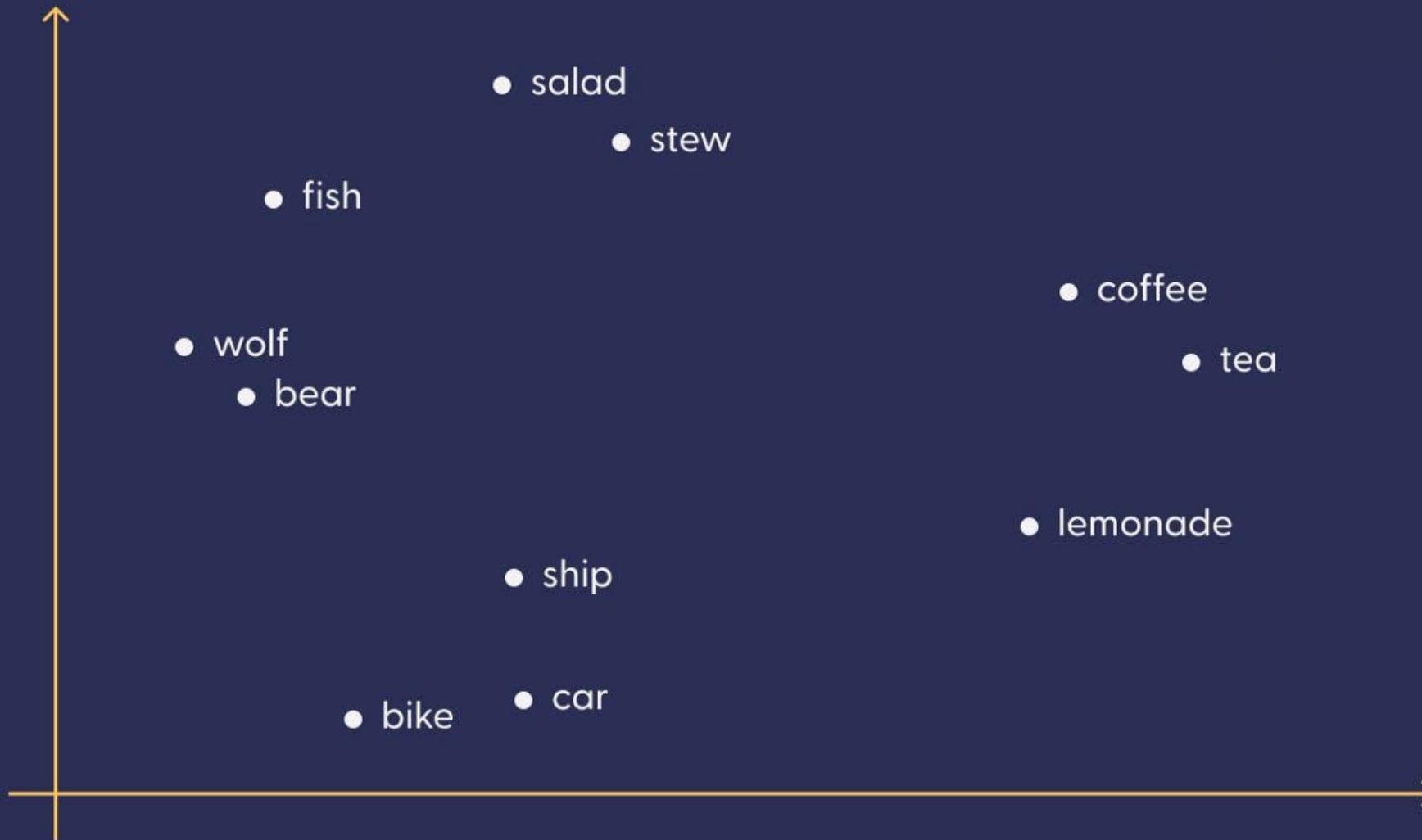
Text Embeddings/Text Vectors

to be or not to be

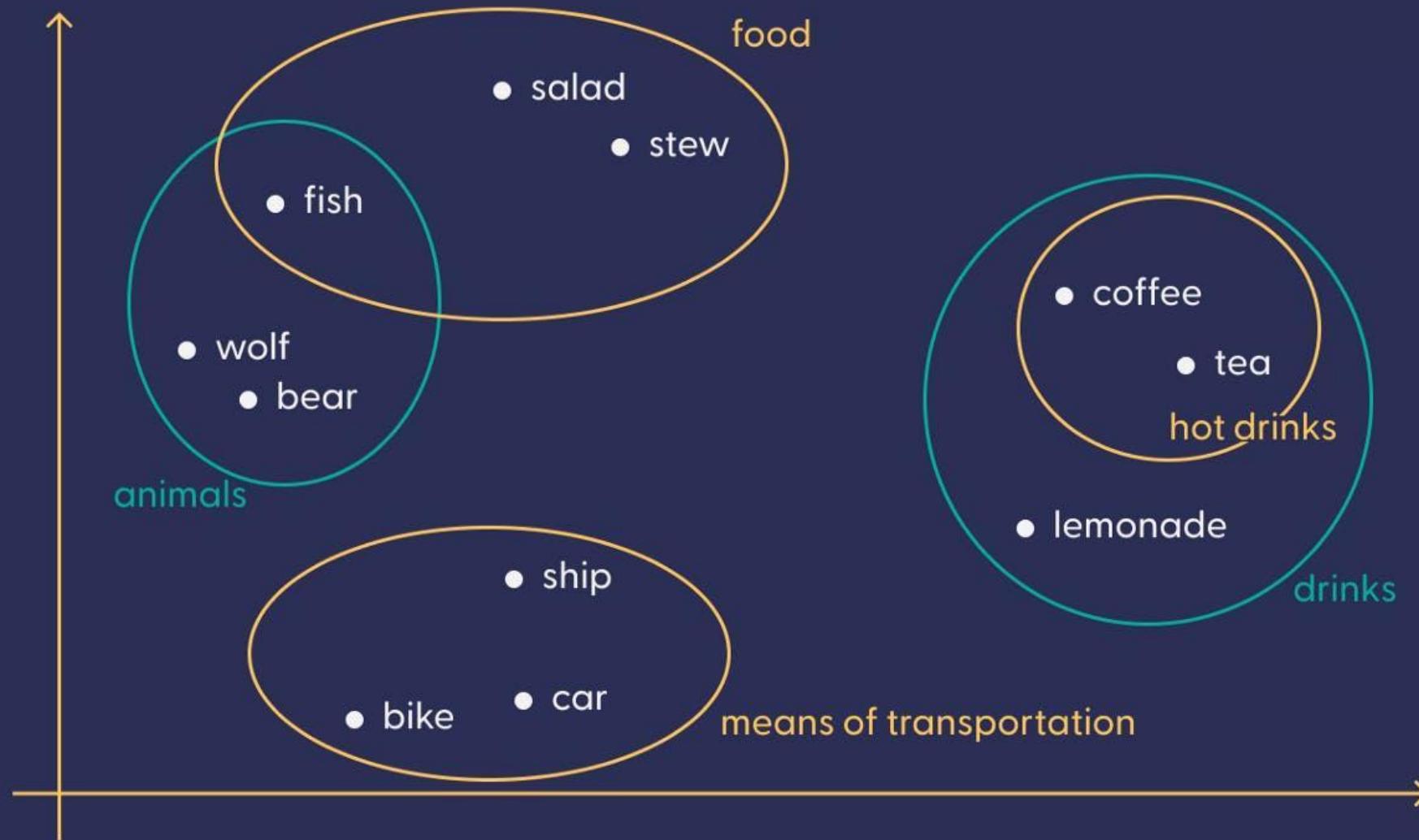


- Manageable by computers
- Different techniques:
 - Sparse: TF-IDF, BM25...
 - Dense: Trained models (Sentence Transformers, Cohere, OpenAI...)
- Come in different dimensions

Text Embeddings - 2 dimensional



Text Embeddings - Semantic Similarity

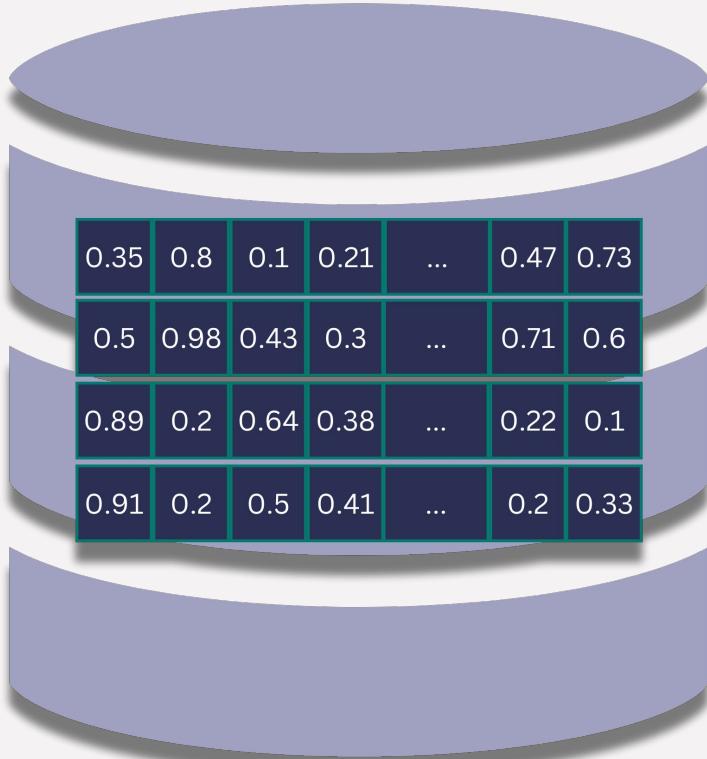


02

Vector Databases



Vector Databases



- Databases that store high-dimensional vectors
- Optimized for vectors:
 - Vector search
 - CRUD operations
 - Metadata filtering

Databases

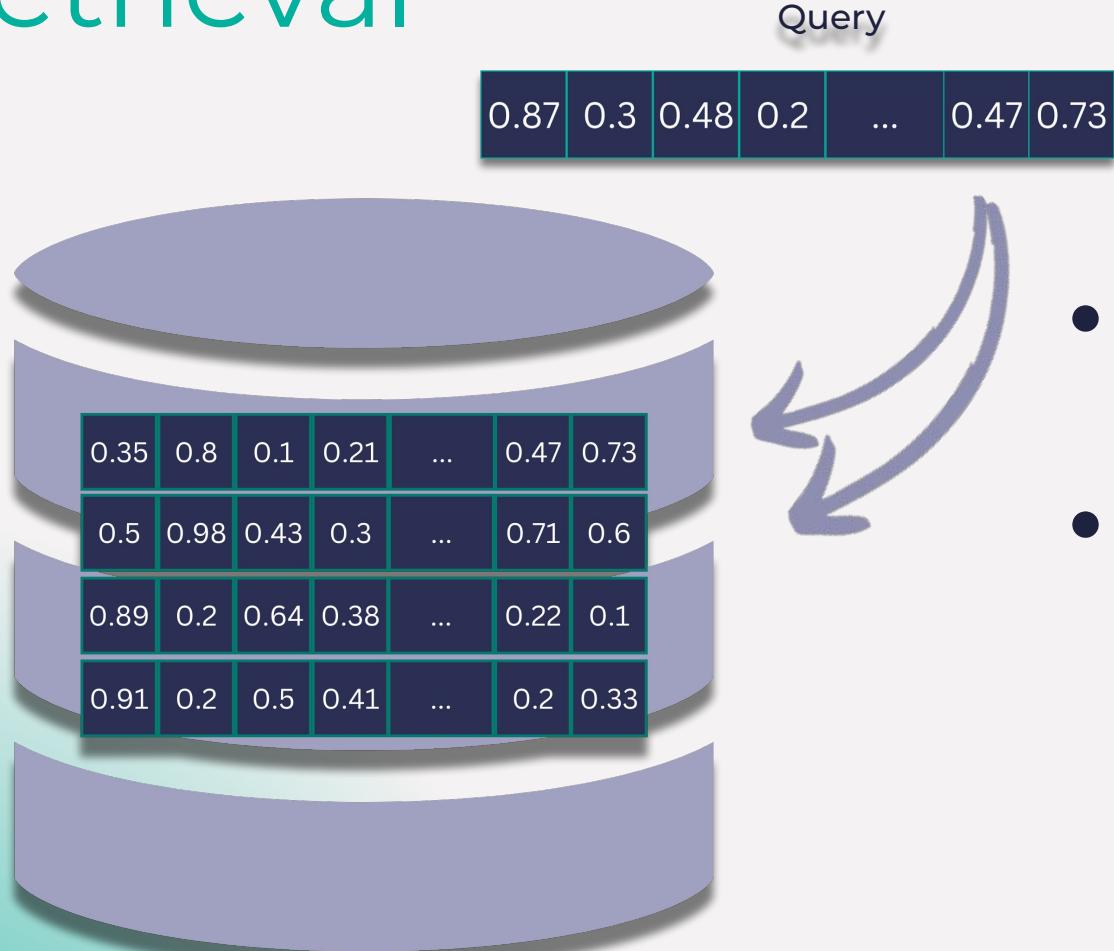
<p>Pure vector databases</p>      	<p>Full-text search databases</p>  
<p>Vector-capable SQL databases</p> <p>Pgvector for</p>  PostgreSQL	<p>Vector-capable NoSQL databases</p>   

03

Retrieval



Retrieval



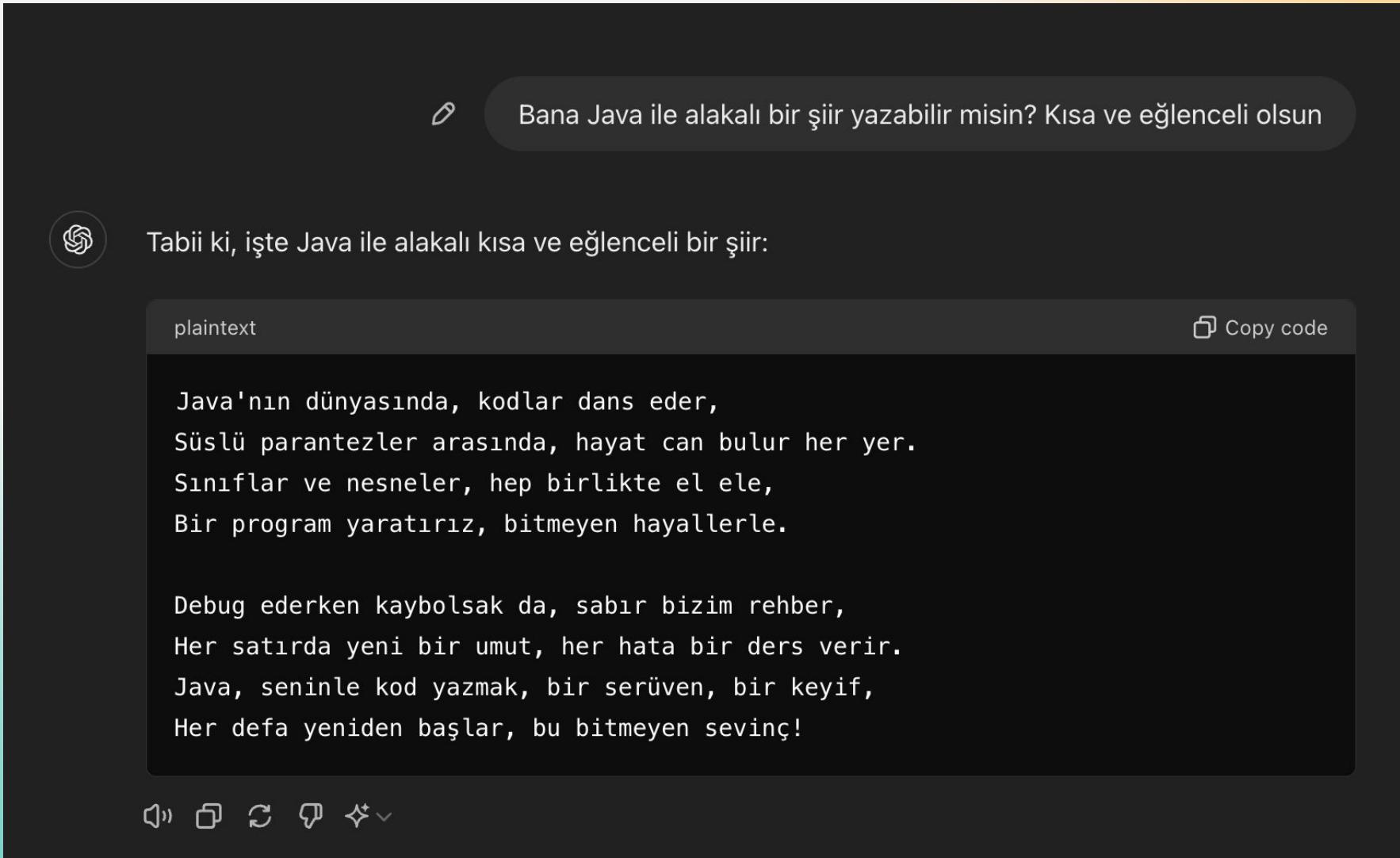
- Getting the most relevant information to the query
- Used for semantic search, question answering and more

04

LLMs



Large Language Models (LLMs)



Large Language Models (LLMs)

- Big language models
- More data & more parameter
- Prompt → Human-like output
- Text generation: summarization, generative QA, writing code, chat...

Model Providers - Which one?

ANTHROPIC

 OpenAI

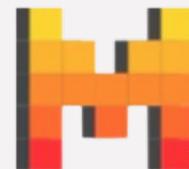
 cohere

 Gemini



Hugging Face

 Meta AI



MISTRAL
AI_



Amazon
Bedrock



Amazon
SageMaker



Azure

LLMs being LLMs

- Not conscious
- Use carefully



ChatGPT 4o >



Hadi taş kâğıt makas oynayalım,
sen başla



Tamam, ben başlıyorum: Taş. Sıra sende!

Benim seçimim kâğıt



Tebrikler, kâğıt taşı sarar! Bir daha
oynayalım mı? Eğer evet, ben tekrar
başlayayım mı?

Evet, oynamaya devam edelim



Tamam, ben başlıyorum: Makas. Sıra
sende!

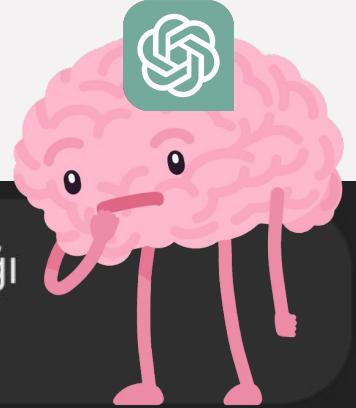
Taş



Tebrikler, taş makası kırar! Yine kazandın.

<https://x.com/hrrcnes/status/1793335665117044990>

LLMs know everything?



Bilge'nin Türkiye Java Community meetup'ında yapacağı konuşmanın konusu nedir?

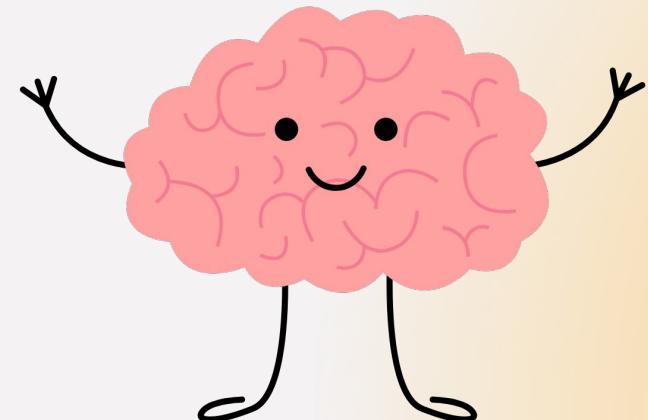


Bilge'nin Türkiye Java Community meetup'ında yapacağı konuşmanın konusu henüz belirlenmemiş gibi görünüyor. Konuya ilgili bir bilgi vermediği için net bir cevap veremiyorum. Konu hakkında daha fazla bilgi verebilirseniz, size yardımcı olmaktan mutluluk duyarım.



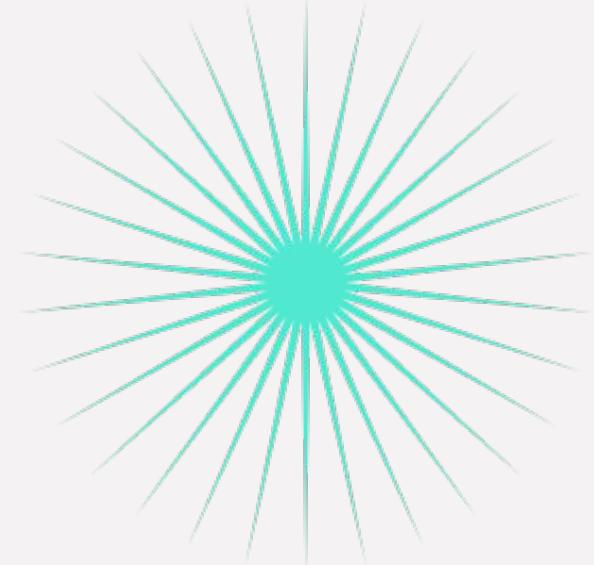
LLM: Limitations

- LLMs do not know the answer to everything
- But they are good at following instructions
- We can help them in their task by giving them the relevant context + instruction

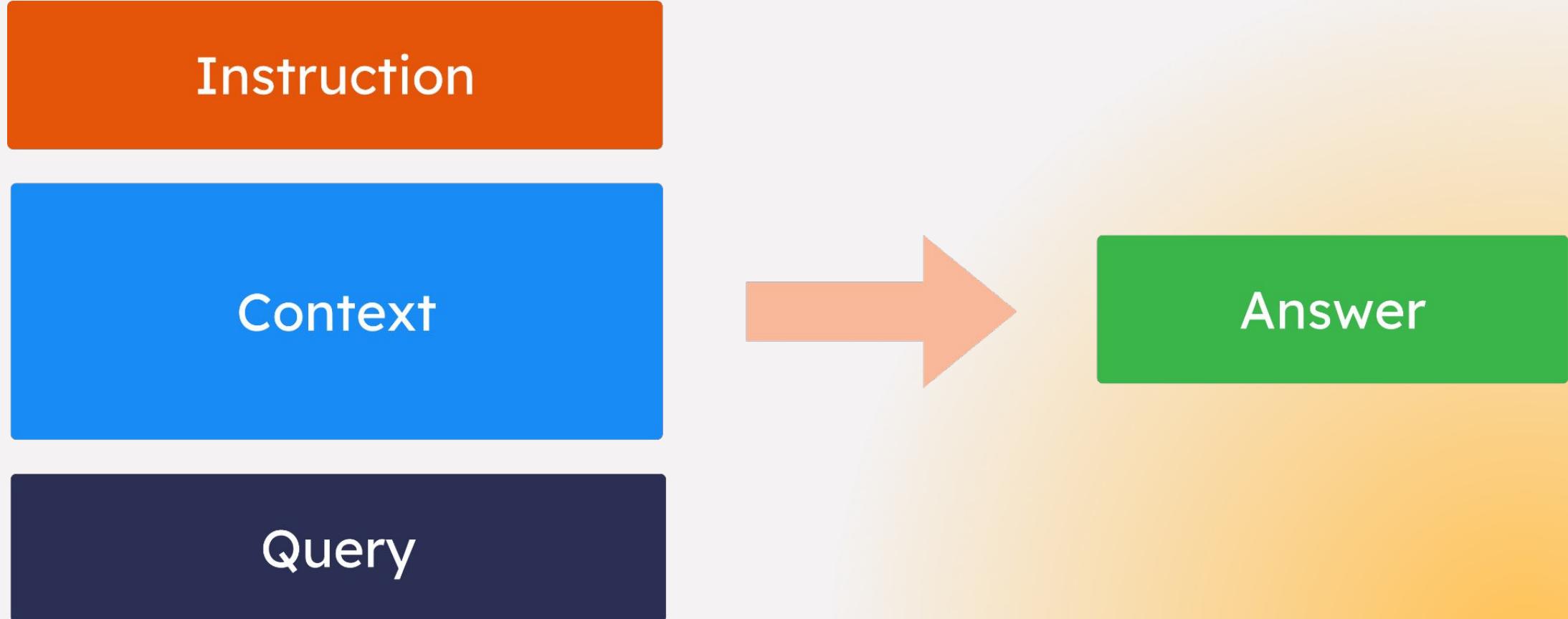


05

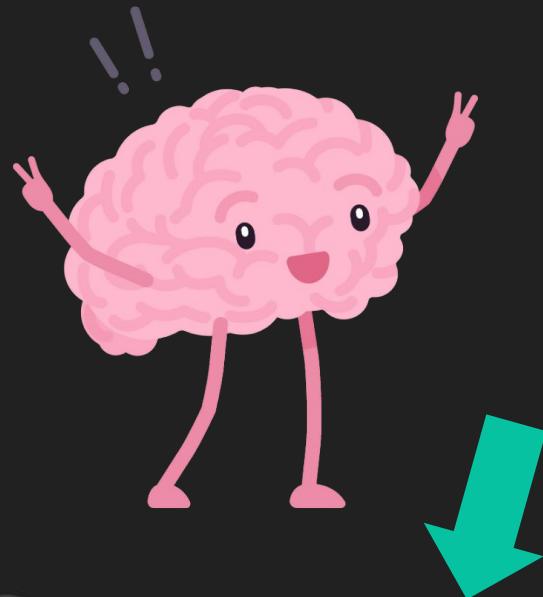
Retrieval Augmented Generation



Retrieval Augmented Generation



Retrieval Augmented Generation



Sana sorulan sorulara sadece verdiğim bilgiler doğrultusunda cevap ver, bilmediğinde bunu belirt.

Context: KONUSMACILAR + KONULAR

Question: Bilge'nin Türkiye Java Community meetup'ında yapacağı konuşmanın konusu nedir?

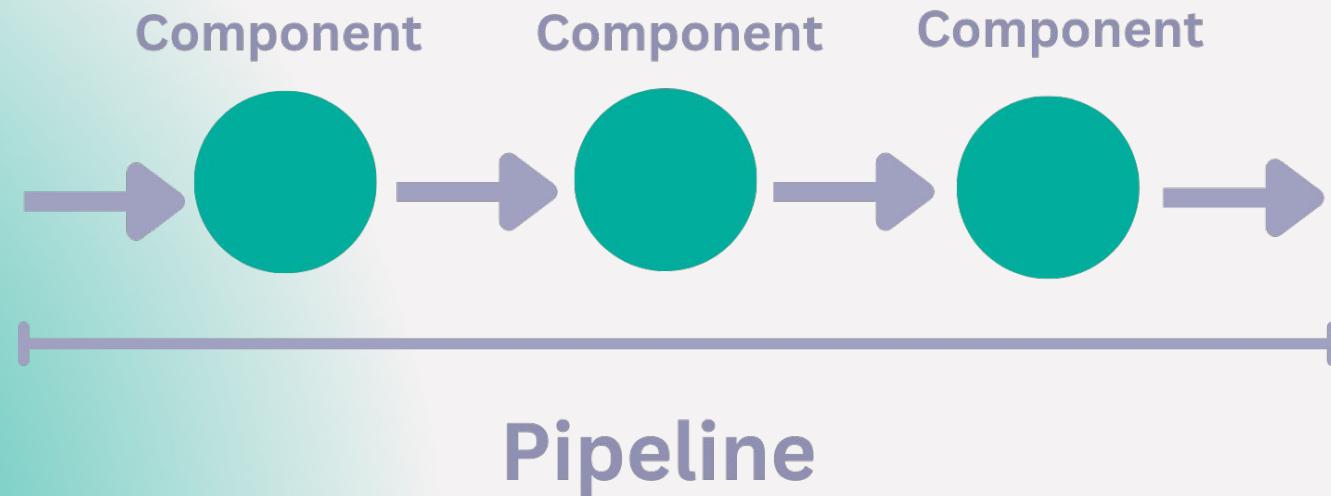


Bilge'nin Türkiye Java Community meetup'ında yapacağı konuşmanın konusu "Mastering Generative AI: From Word Embeddings to Custom AI Agents". Bu konuşmada, temel NLP kavramlarını, büyük dil modellerini ve prompting sanatını inceleyerek kendi AI ajanımızı nasıl uygulayabileceğimizi ve bu ajanı geliştirmek için kullanabileceğimiz teknikleri keşfedeceğiz.

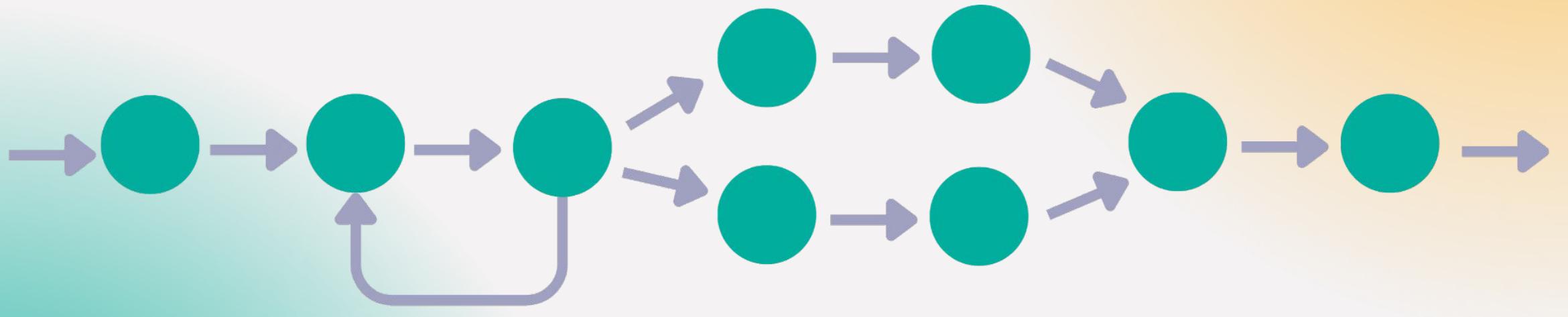
Haystack

Starred 14k

- Haystack is an open source Python framework for building production-ready LLM applications
- Prototyping, evaluation, deployment, monitoring...
- Building blocks: Components & Pipelines

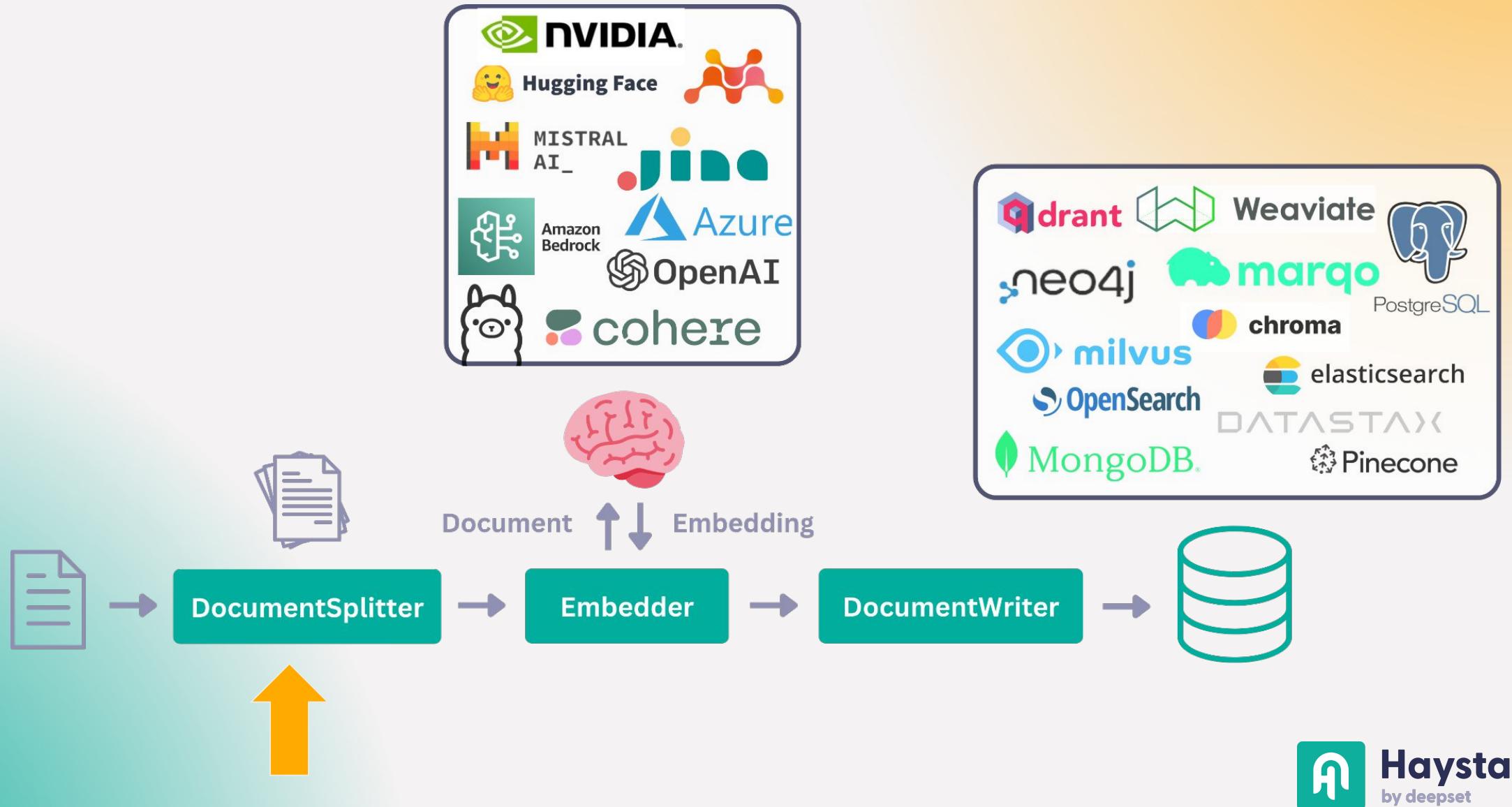


Haystack

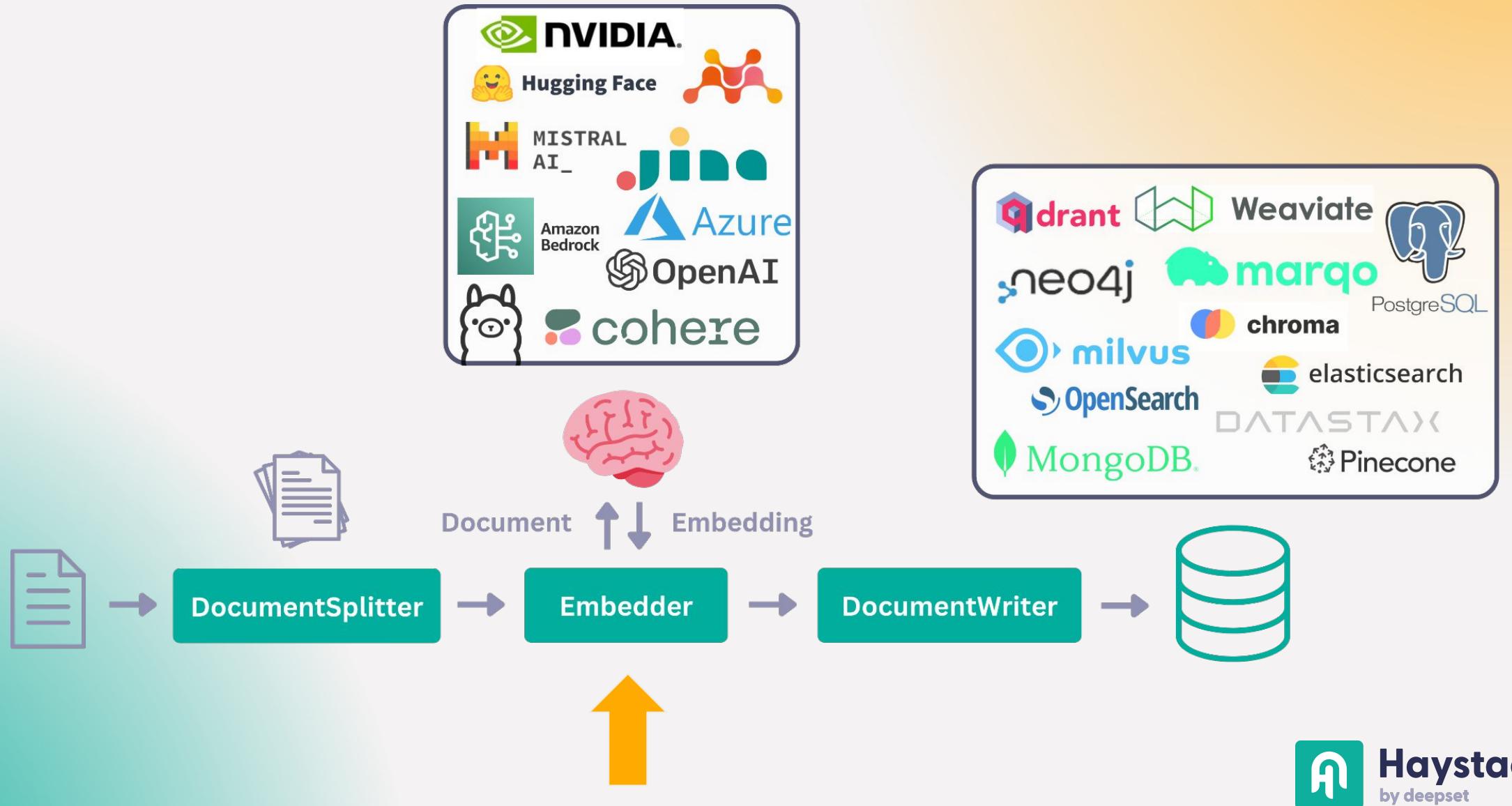


Custom Pipeline

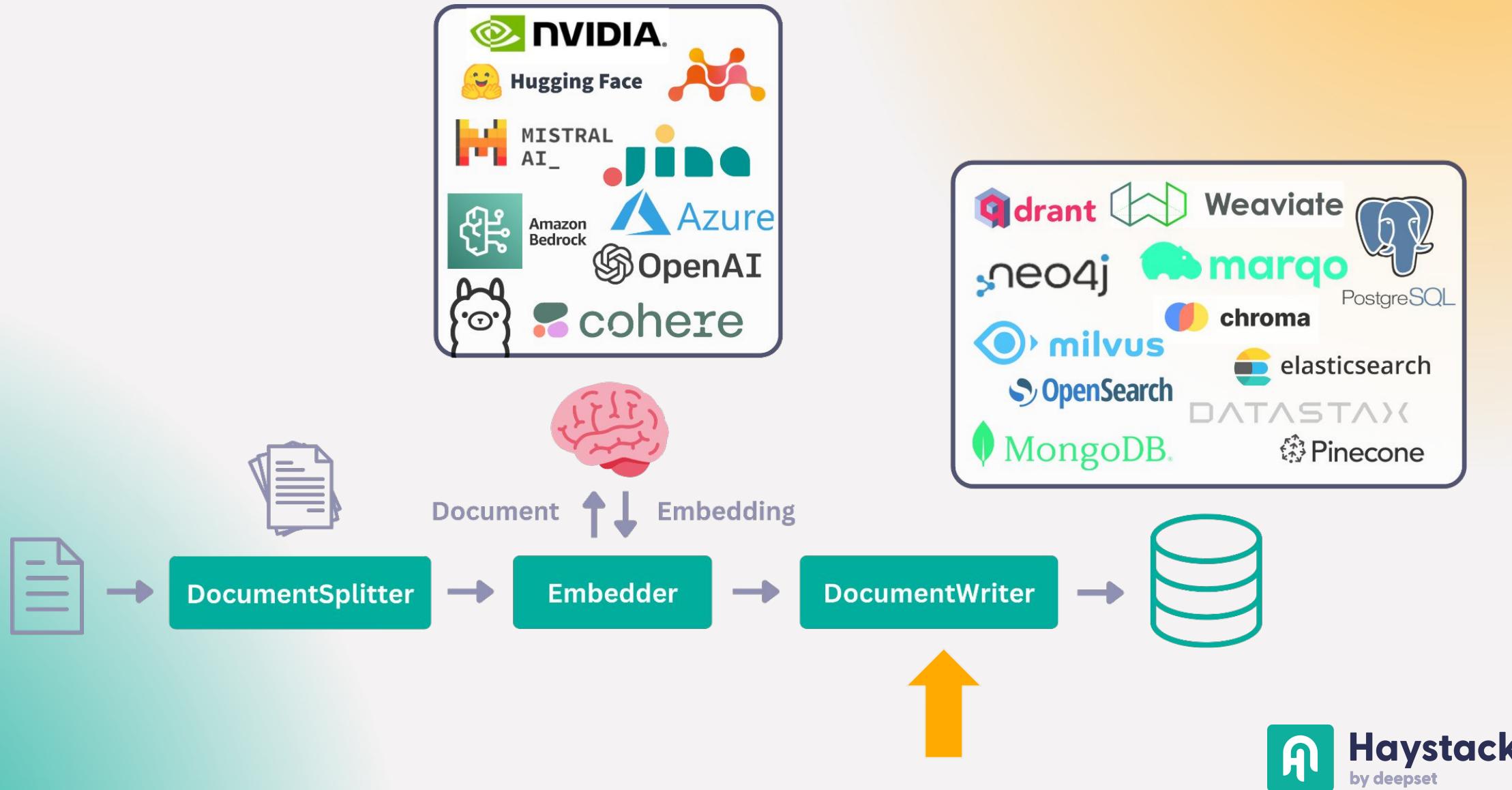
Indexing Pipeline



Indexing Pipeline



Indexing Pipeline



Indexing Pipeline



```
from haystack import Pipeline
from haystack.document_stores.in_memory import InMemoryDocumentStore
from haystack.components.embedders import SentenceTransformersDocumentEmbedder
from haystack.components.preprocessors import DocumentSplitter
from haystack.components.writers import DocumentWriter

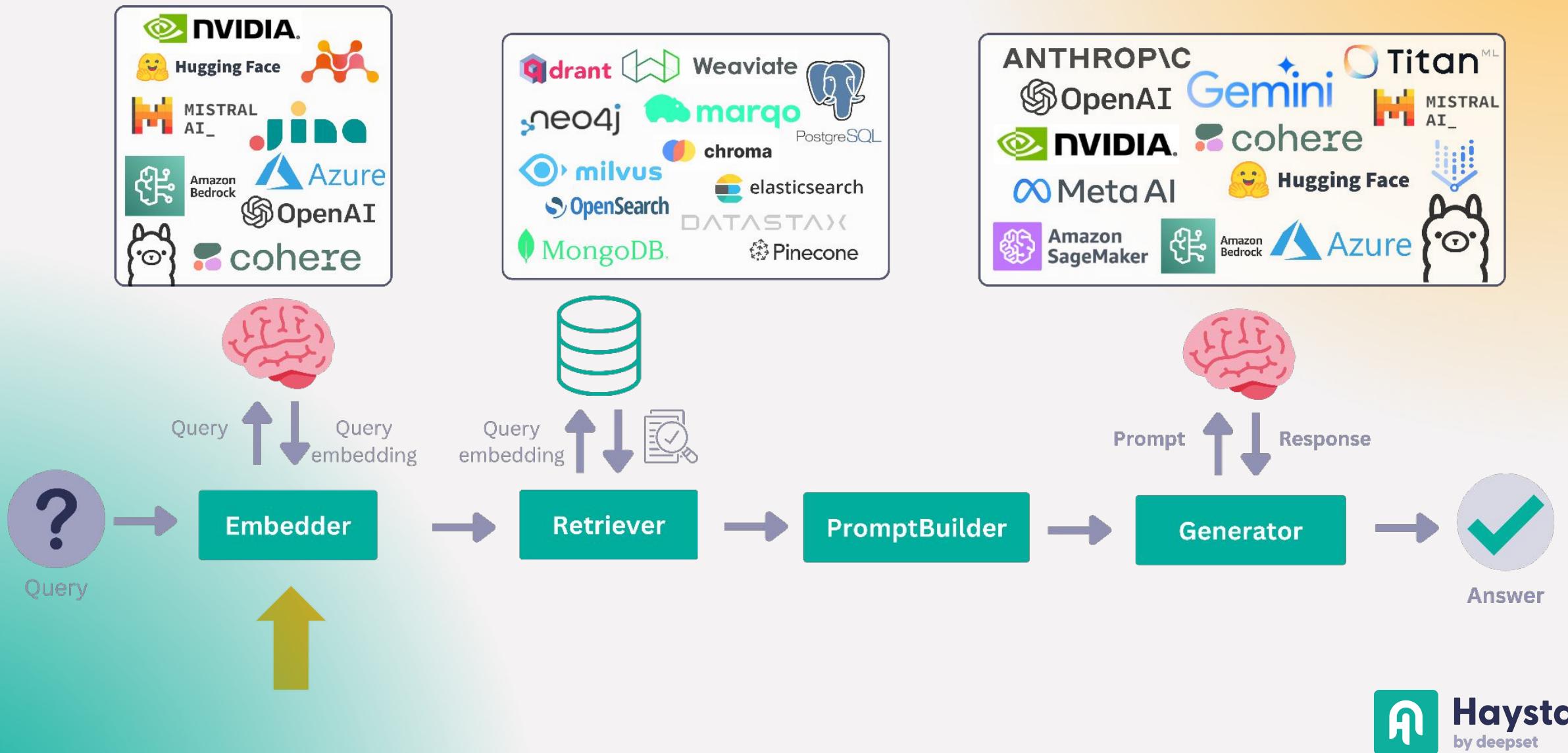
document_store = InMemoryDocumentStore()

indexing_pipeline = Pipeline()
indexing_pipeline.add_component("splitter", DocumentSplitter(split_by="word", split_length=450, split_overlap=50))
indexing_pipeline.add_component("embedder", SentenceTransformersDocumentEmbedder(model="sentence-transformers/all-MiniLM-L6-v2"))
indexing_pipeline.add_component("writer", DocumentWriter(document_store=document_store))

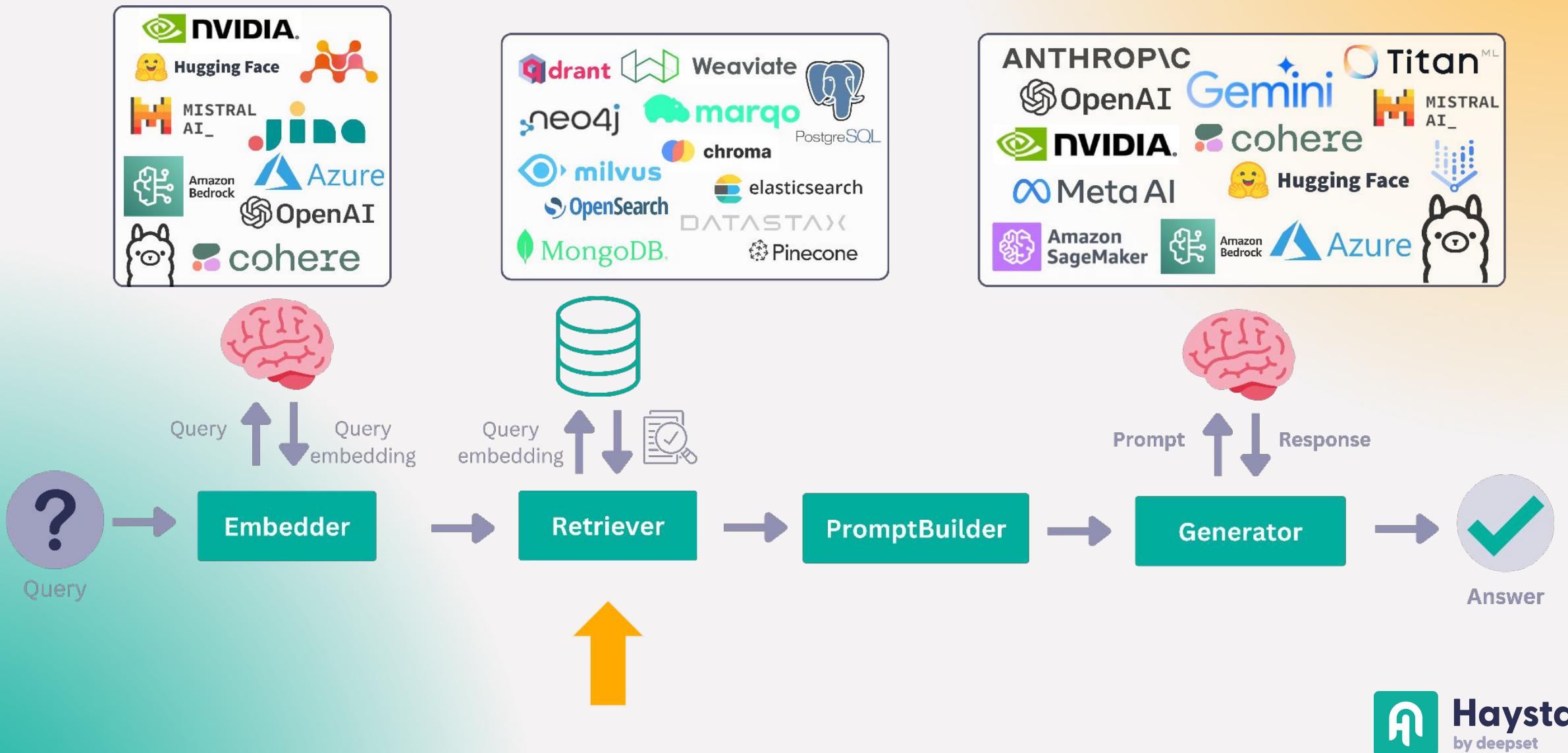
indexing_pipeline.connect("splitter.documents", "embedder")
indexing_pipeline.connect("embedder.documents", "writer.documents")

indexing_pipeline.run({"text_file_converter": {"sources": files}})
```

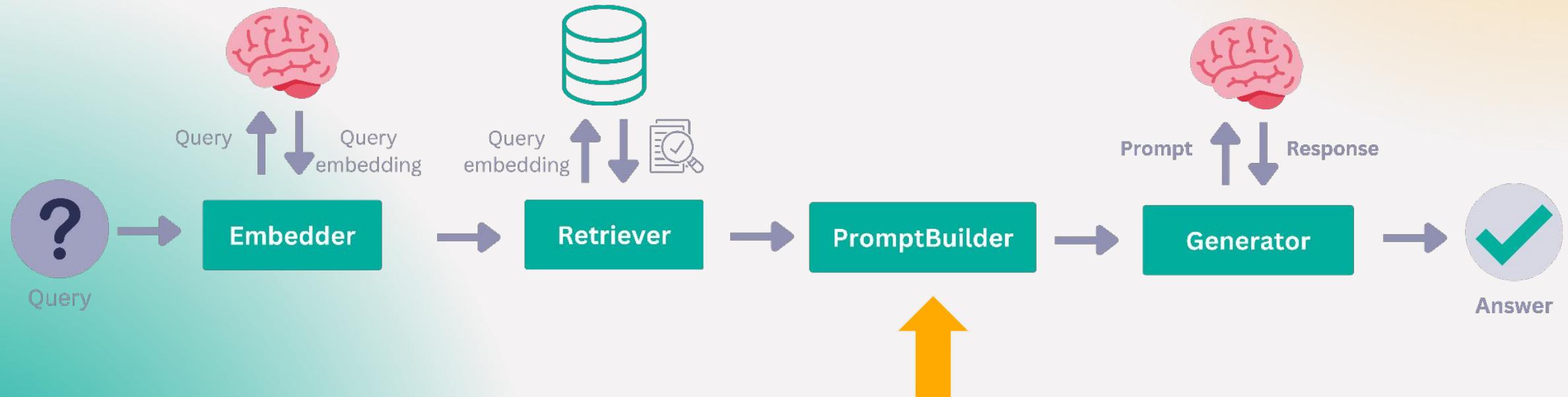
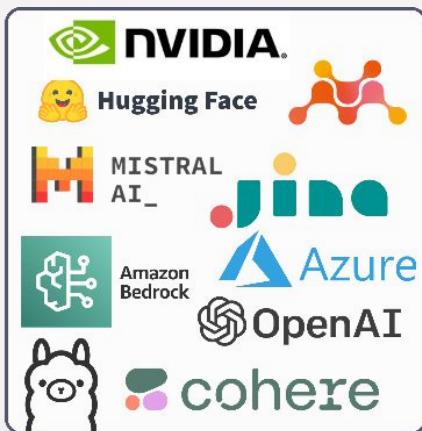
RAG Pipeline



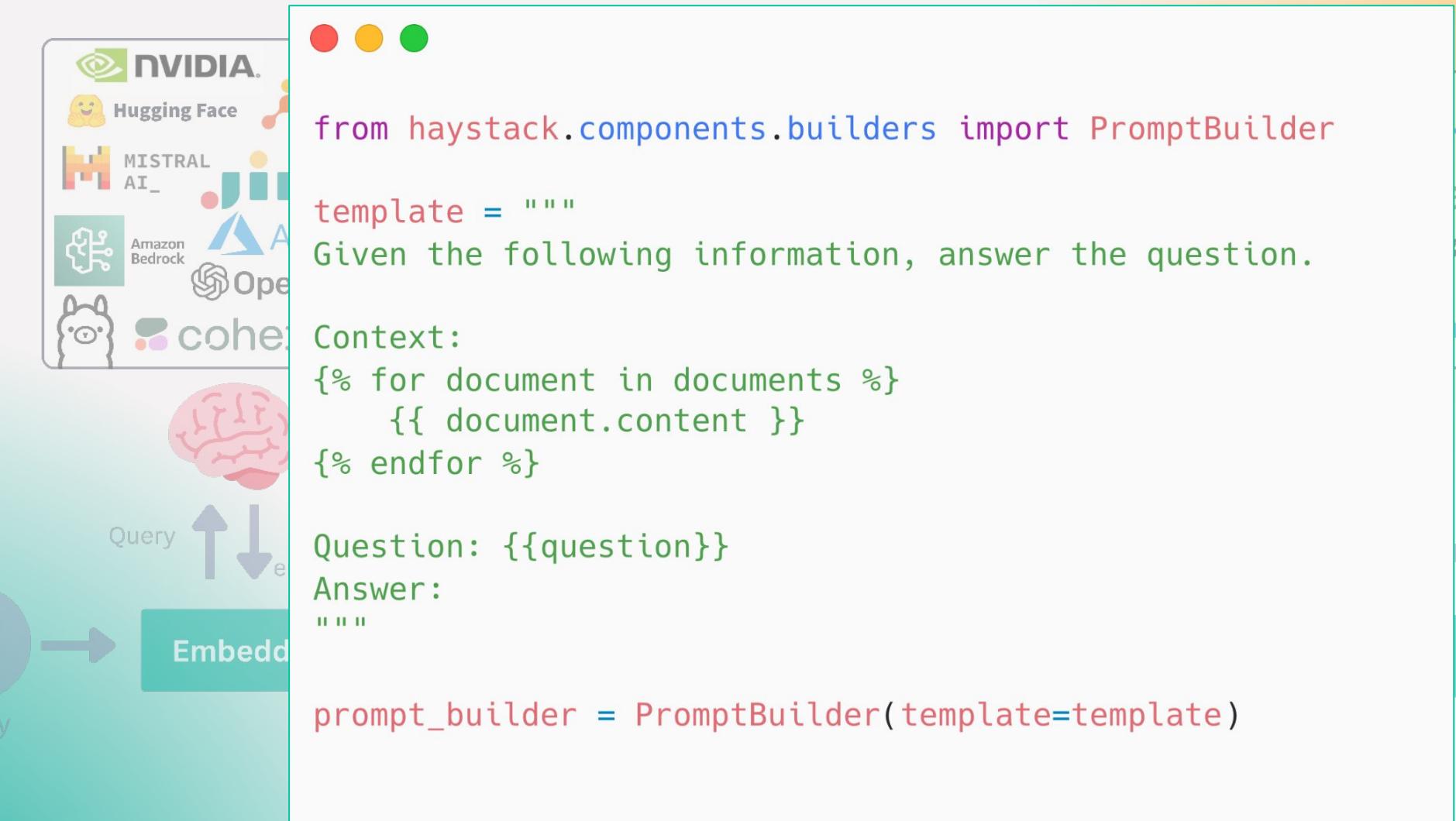
RAG Pipeline



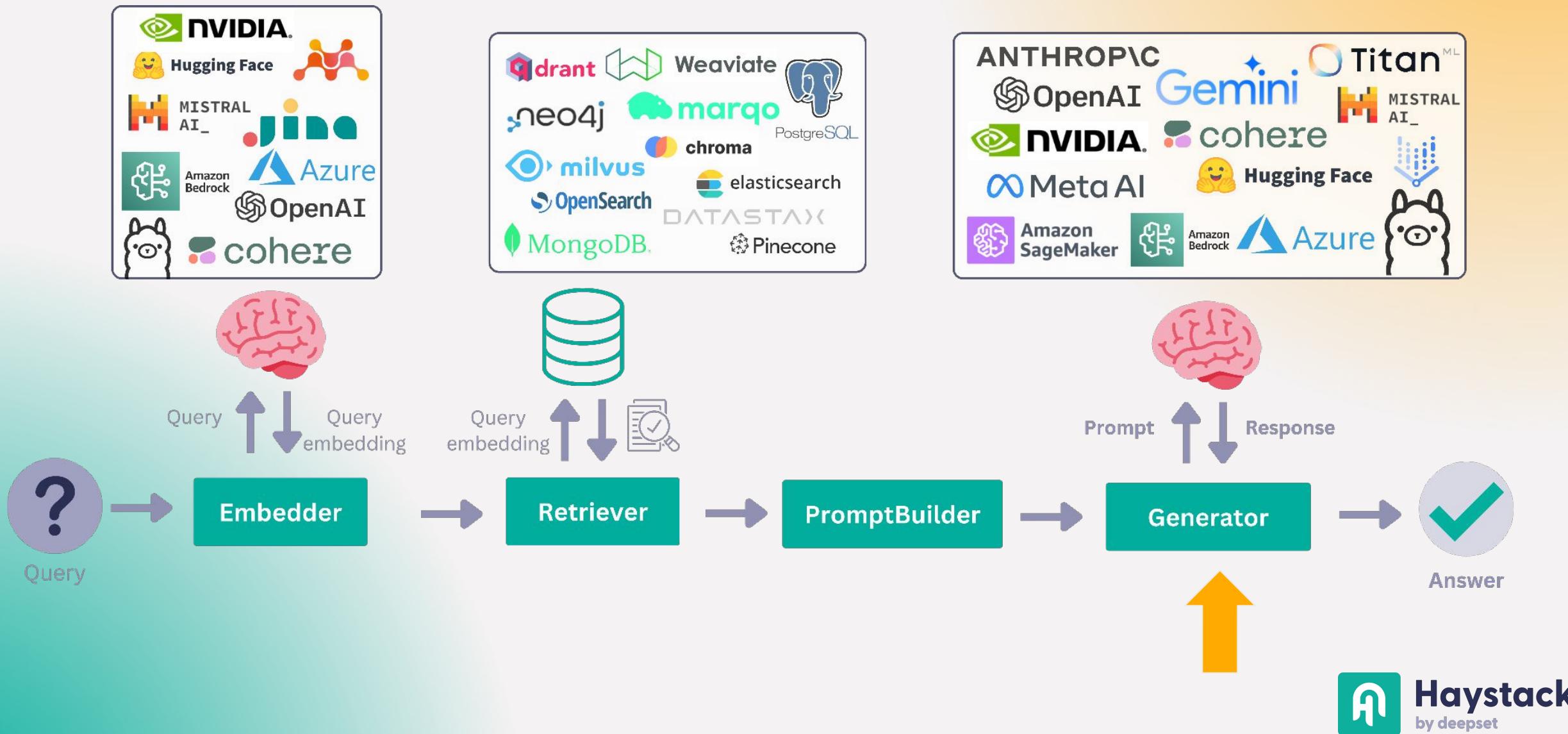
RAG Pipeline



RAG Pipeline



RAG Pipeline



RAG Pipeline



```
from haystack import Pipeline
from haystack.components.retrievers.in_memory import InMemoryEmbeddingRetriever
from haystack.components.embedders import SentenceTransformersTextEmbedder
from haystack.components.preprocessors import DocumentSplitter
from haystack.components.generators import OpenAIGenerator

rag_pipeline = Pipeline()
rag_pipeline.add_component("text_embedder", SentenceTransformersTextEmbedder(model="sentence-transformers/all-MiniLM-L6-v2"))
rag_pipeline.add_component("retriever", InMemoryEmbeddingRetriever(document_store=document_store, top_k=5))
rag_pipeline.add_component("prompt_builder", PromptBuilder(template=prompt_template))
rag_pipeline.add_component("generator", OpenAIGenerator(model='gpt-3.5-turbo'))

rag_pipeline.connect("text_embedder.embedding", "retriever.query_embedding")
rag_pipeline.connect("retriever", "prompt_builder.documents")
rag_pipeline.connect("prompt_builder", "generator")

query="What's the topic Bilge will talk about?"

rag_pipeline.run({"text_embedder":{"text": query}, "prompt_builder":{"query": query}})

>>> Bilge will talk about "Mastering Generative AI: From Word Embeddings to Custom AI Agents."
```



Congratulations!

06

Agents



Agents

- Complex systems (multiple LLMs)
- Prompting: Chain of Thought/ReAct
- Generate accurate responses to complex queries
- Might use tools for answers
- Use cases: chat bot, personal assistants



Agents - CoT

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. 

Agents - CoT

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. 

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. 

Agents - CoT

(1) Hotspot QA

Question: How many rooms are in the hotel that is home to the Cirque du Soleil show Mystere?

HotpotQA label: 2,664

Outdated 

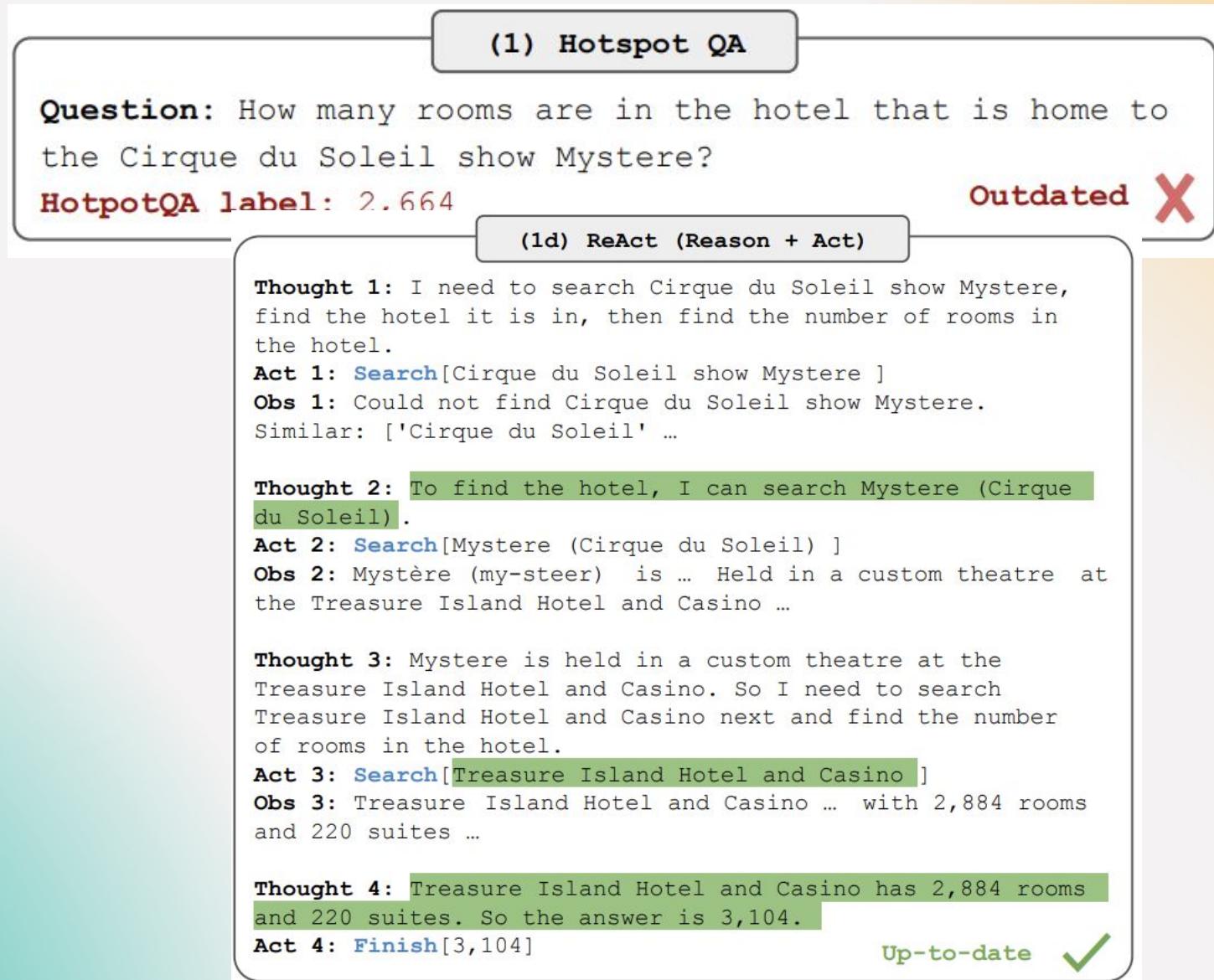
(1b) CoT (Reason Only)

Thought: Let's think step by step. The hotel that is home to the Cirque du Soleil show Mystere is Treasure Island. Treasure Island has 2,885 rooms, so the answer is 2,885.

Answer: 2,885



Agents - Reason + Act (ReAct)



Agents - ReAct Prompt



```
agent_prompt = """
```

In the following conversation, a human user interacts with an AI Agent. The human user poses questions, and the AI Agent goes through several steps to provide well-informed answers.

The AI Agent must use the available tools to find the up-to-date information. The final answer to the question should be truthfully based solely on the output of the tools. The AI Agent should ignore its knowledge when answering the questions.

The AI Agent has access to these tools:

```
{tool_names_with_descriptions}
```

AI Agent responses must start with one of the following:

Thought: [the AI Agent's reasoning process]

Tool: [tool names] (on a new line) Tool Input: [input as a question for the selected tool WITHOUT quotation marks and on a new line] (These must always be provided together and on separate lines.)

Observation: [tool's result]

Final Answer: [final answer to the human user's question]

When selecting a tool, the AI Agent must provide both the "Tool:" and "Tool Input:" pair in the same response, but on separate lines.

The AI Agent should not ask the human user for additional information, clarification, or context.

If the AI Agent cannot find a specific answer after exhausting available tools and approaches, it answers with Final Answer: inconclusive

```
Question: {query}
```

```
Thought: {transcript}
```

```
"""
```

06.5

Function Calling

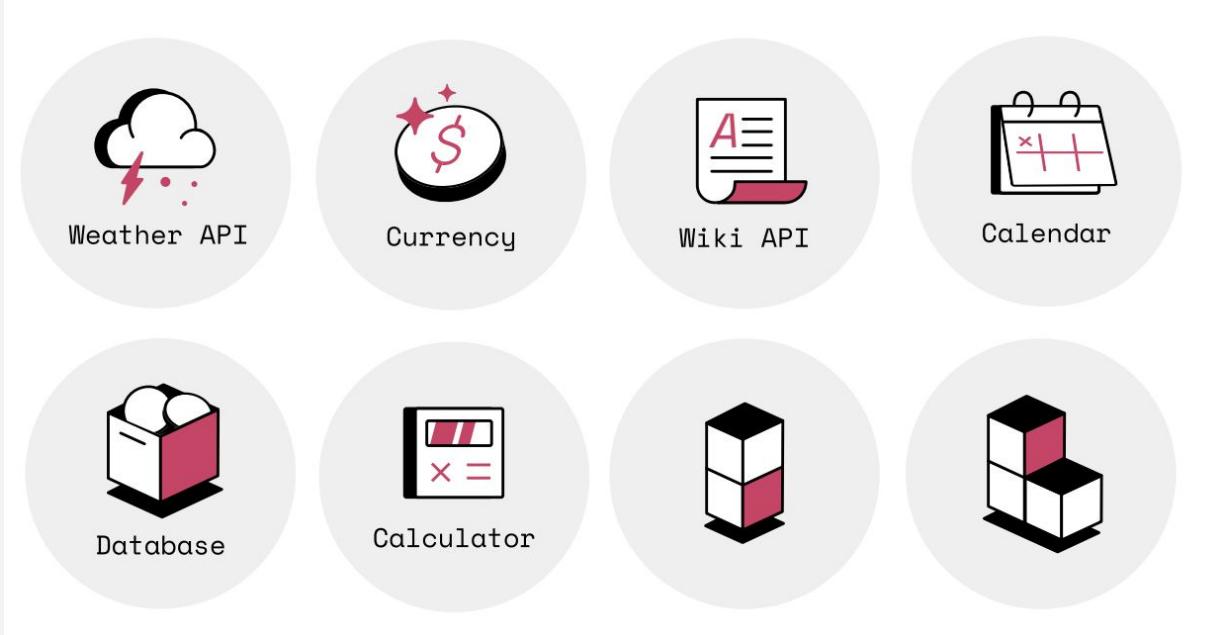


Function Calling

- Passing prompt + functions/tools (name, description, parameters)
- User query -> function name + JSON object for arguments
- Use arguments to call the function
- Voila! 🎉



Function Calling - Tools



<https://arxiv.org/pdf/2205.00445>



```
def get_current_weather(location: str, unit: str = "celsius"):  
    ## do something  
    return {"weather": "sunny", "temperature": 21.8, "unit": unit}
```

Function Calling - Tool definition

```
tools = [
  {
    "type": "function",
    "function": {
      "name": "get_current_weather",
      "description": "Get the current weather",
      "parameters": {
        "type": "object",
        "properties": {
          "location": {
            "type": "string",
            "description": "The city and state, e.g. San Francisco, CA",
          },
          "format": {
            "type": "string",
            "enum": ["celsius", "fahrenheit"],
            "description": "The temperature unit to use. Infer this from the users location.",
          },
        },
        "required": ["location", "format"],
      },
    },
  }
]
```



```
def get_current_weather(location: str, unit: str = "celsius"):
    ## do something
    return {"weather": "sunny", "temperature": 21.8, "unit": unit}
```

Function Calling

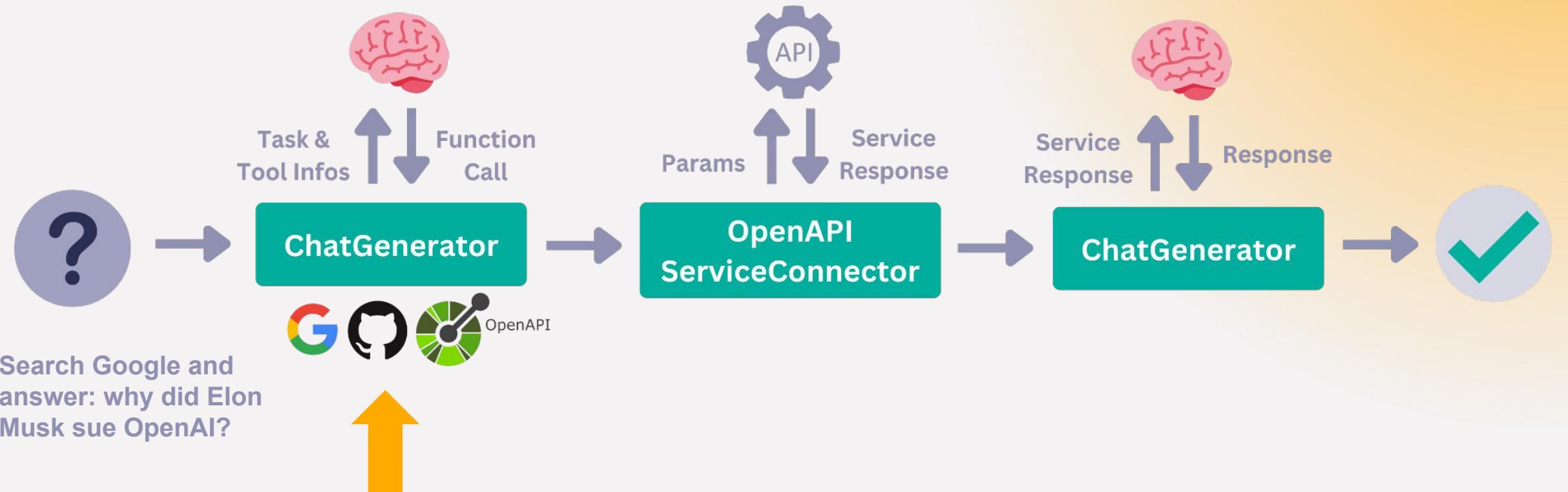


```
messages = []
messages.append({"role": "system", "content": "Don't make assumptions about what values to plug into
functions. Ask for clarification if a user request is ambiguous."})
messages.append({"role": "user", "content": "What's the weather like in Istanbul?"})

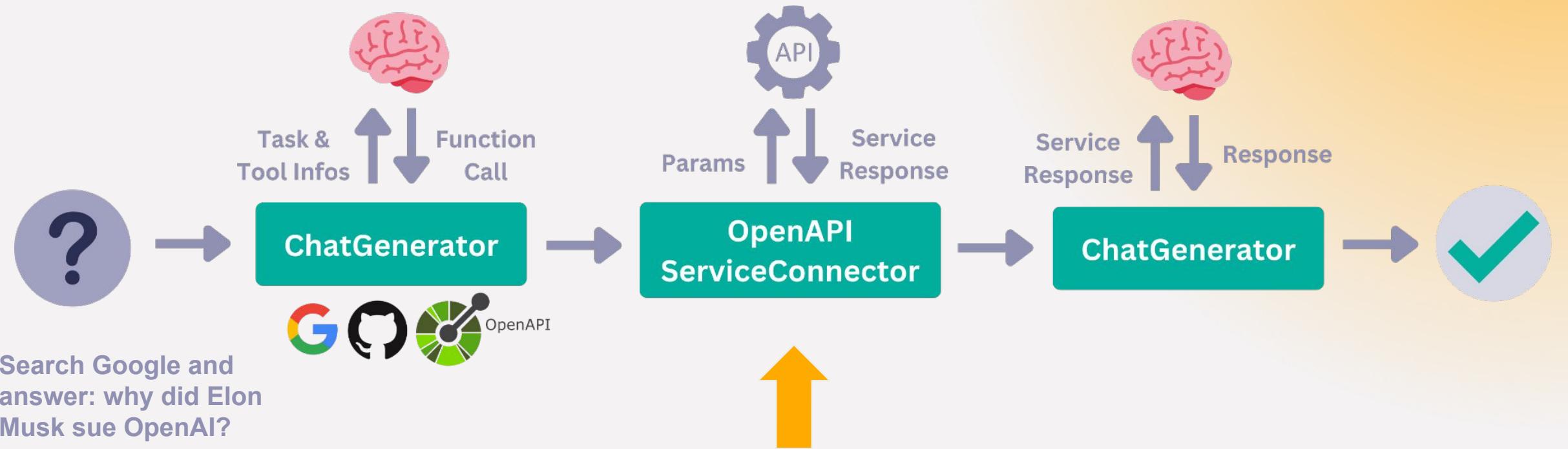
chat_response = chat_completion_request(
    messages, tools=tools
)

>>> ChatCompletionMessage(content=None, role='assistant', function_call=None, tool_calls=
[ChatCompletionMessageToolCall(id='call_...', function=Function(arguments='{"location":"Istanbul,
Turkey","format":"celsius"}', name='get_current_weather'), type='function')])
```

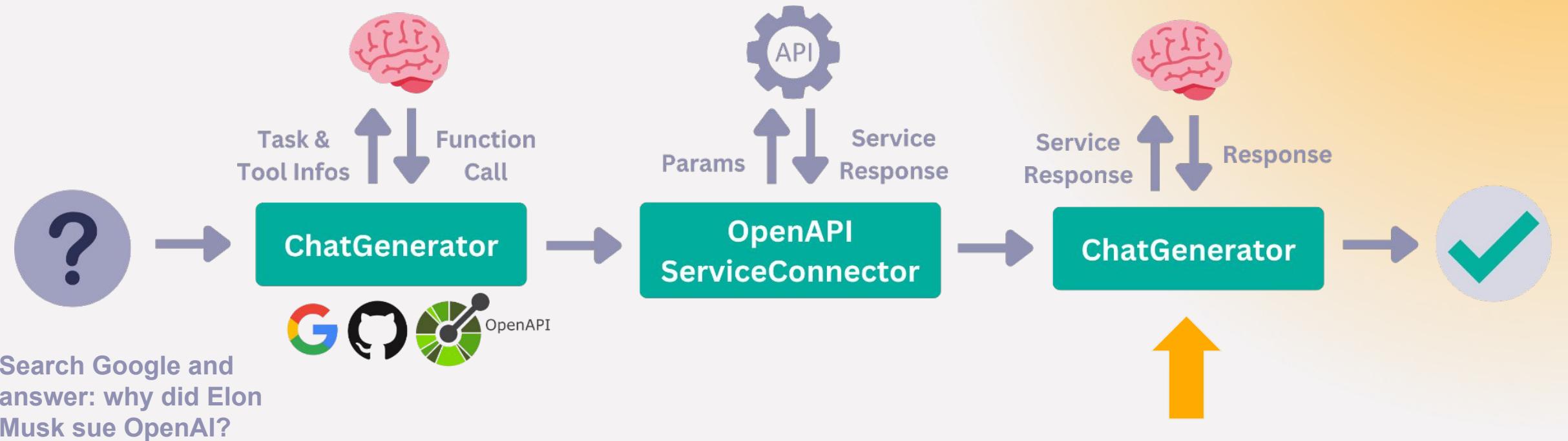
Agents



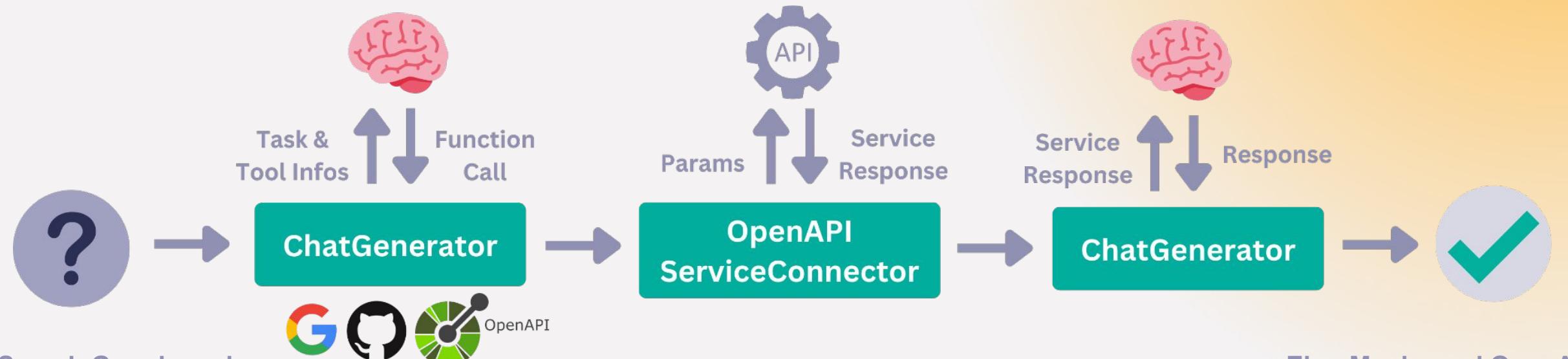
Agents



Agents



Agents



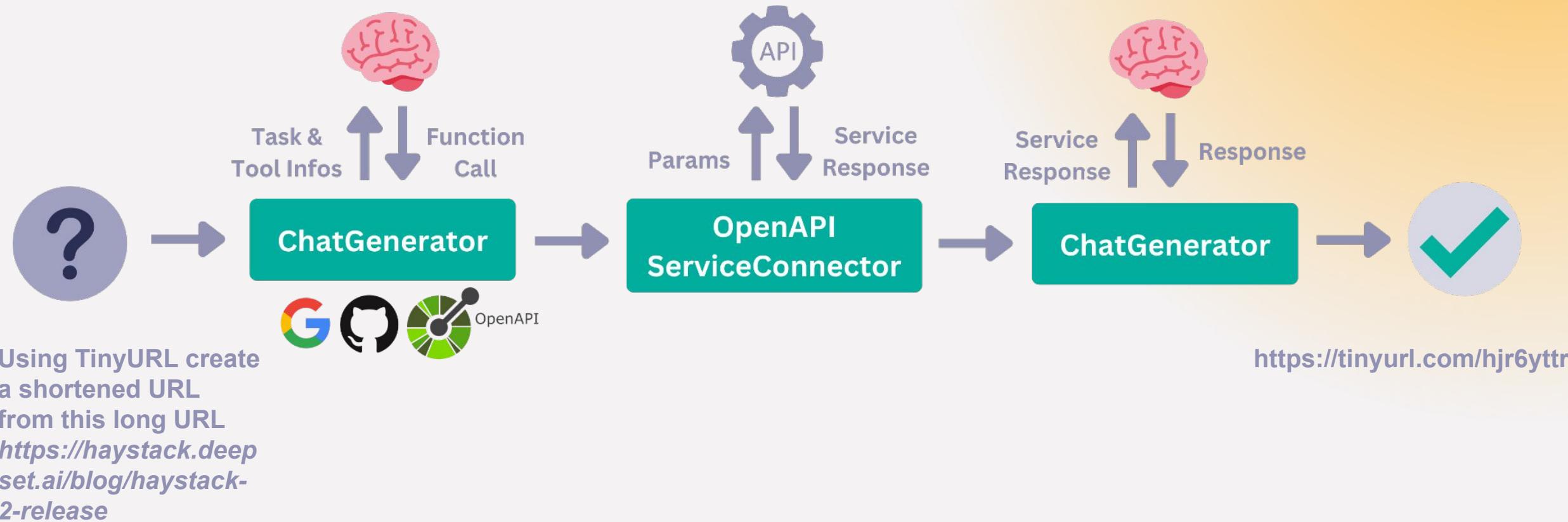
Search Google and answer: why did Elon Musk sue OpenAI?

Elon Musk sued OpenAI for breaching its founding agreement and diverging from its original nonprofit mission....

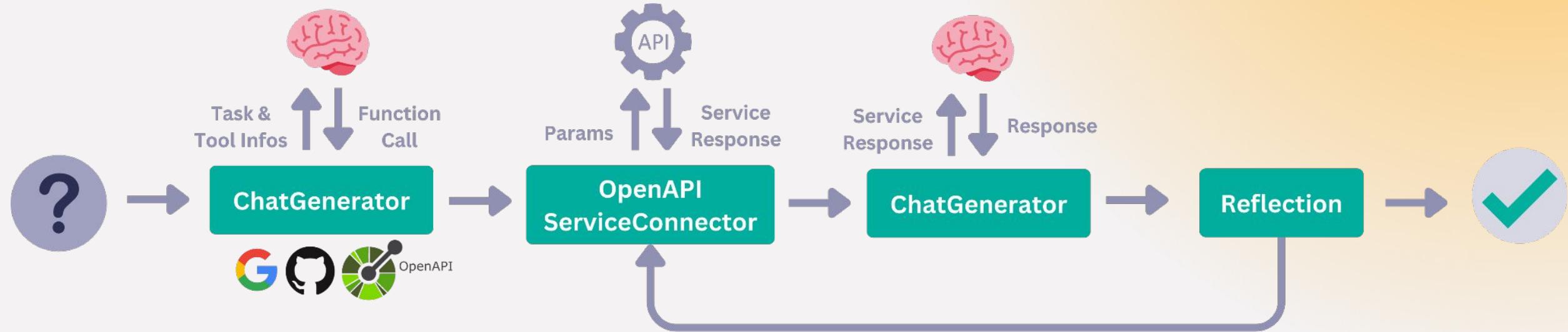


Haystack
by deepset

Agents



Agents



Haystack
by deepset

Recap

- 01 - Text Embeddings
- 02 - Vector Databases
- 03 - Retrieval
- 04 - LLMs
- 05 - RAG
- 06 - Agents & Function Calling



Thank you!



@bilgeycl



Bilge Yücel



bilgeyucel



haystack.deepset.ai