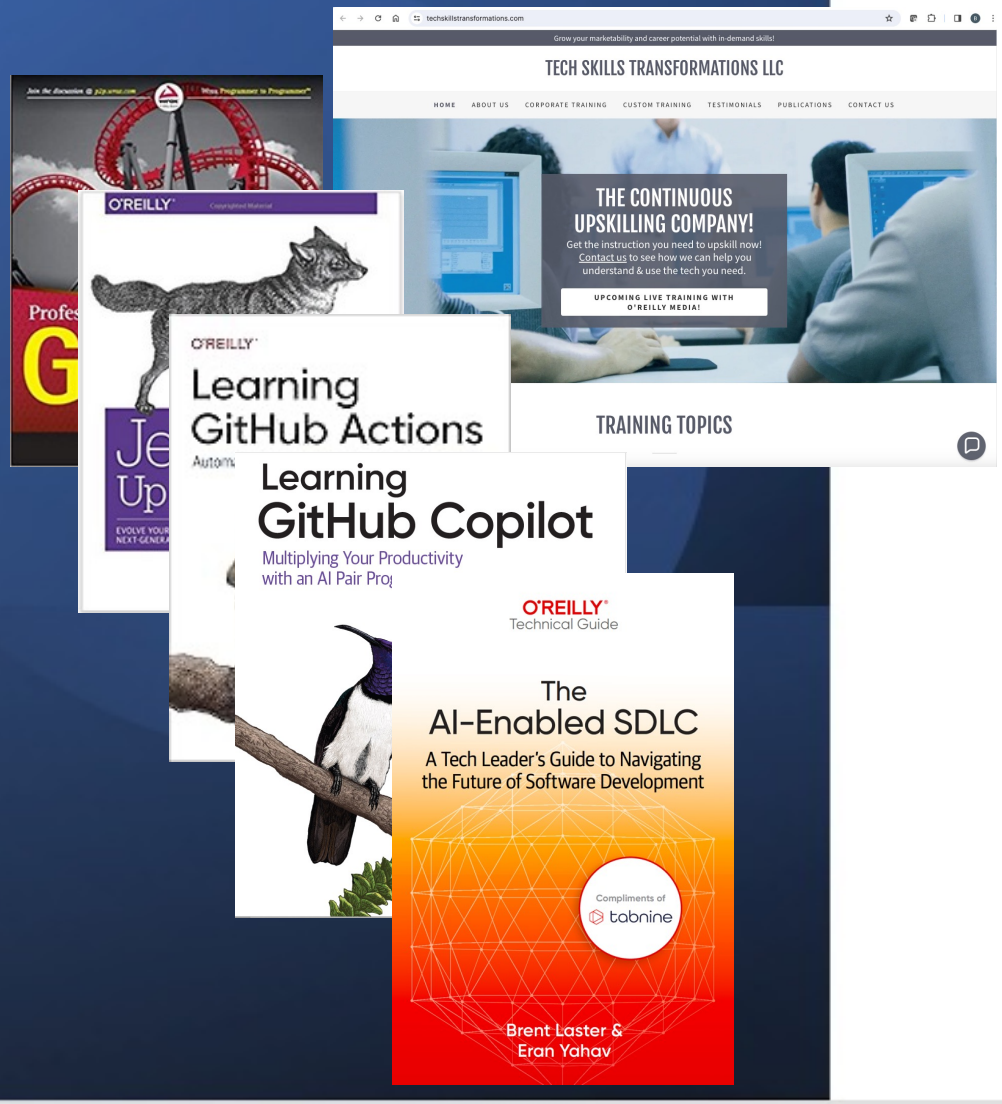
AI 3-in-1: Agents, RAG, and Local Models

Presented by Brent Laster &

Tech Skills Transformations LLC

TECHUPSKILLS ®

About me



- Founder, Tech Skills Transformations LLC
- https://getskillsnow.com
- info@getskillsnow.com

- Long career in corporate as dev, manager, and director in DevOps and other areas

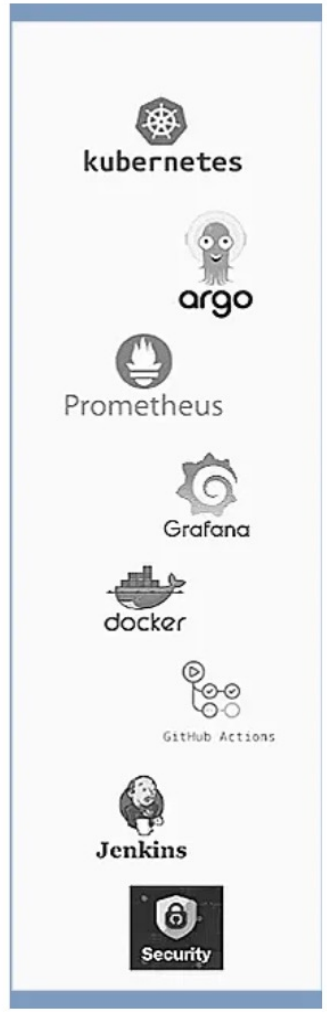
- Author
  - O'Reilly "reports"
  - Books
    - Professional Git
    - Jenkins 2 – Up and Running
    - Learning GitHub Actions
    - Learning GitHub Copilot
    - AI-Enabled SDLC
- Speaker

- Social media

- ❑ **LinkedIn: brentlaster**
- ❑ **X: @BrentCLaster**
- ❑ **Bluesky: brentclaster.bsky.social**
- ❑ **GitHub: brentlaster**

LISTED BELOW ARE A FEW OF THE TECHNOLOGIES FOR WHICH WE OFFER HANDS-ON TRAINING. THESE CAN BE CUSTOMIZED FOR ANY SIZE TEAM FROM 1-100 AND FOR ANY LEVEL FROM BEGINNER TO ADVANCED.

- ArgoCD
- Containers
- Docker
- Gerrit
- Git
- GitHub Actions
- GitHub Codespaces
- GitHub Copilot
- GitHub Foundations
- GitHub Security
- GitLab
- GitOps
- Gradle
- Grafana
- Helm
- Jenkins
- Kubernetes
- Kustomize
- LLMs
- Prometheus
- Tekton
- VS Code

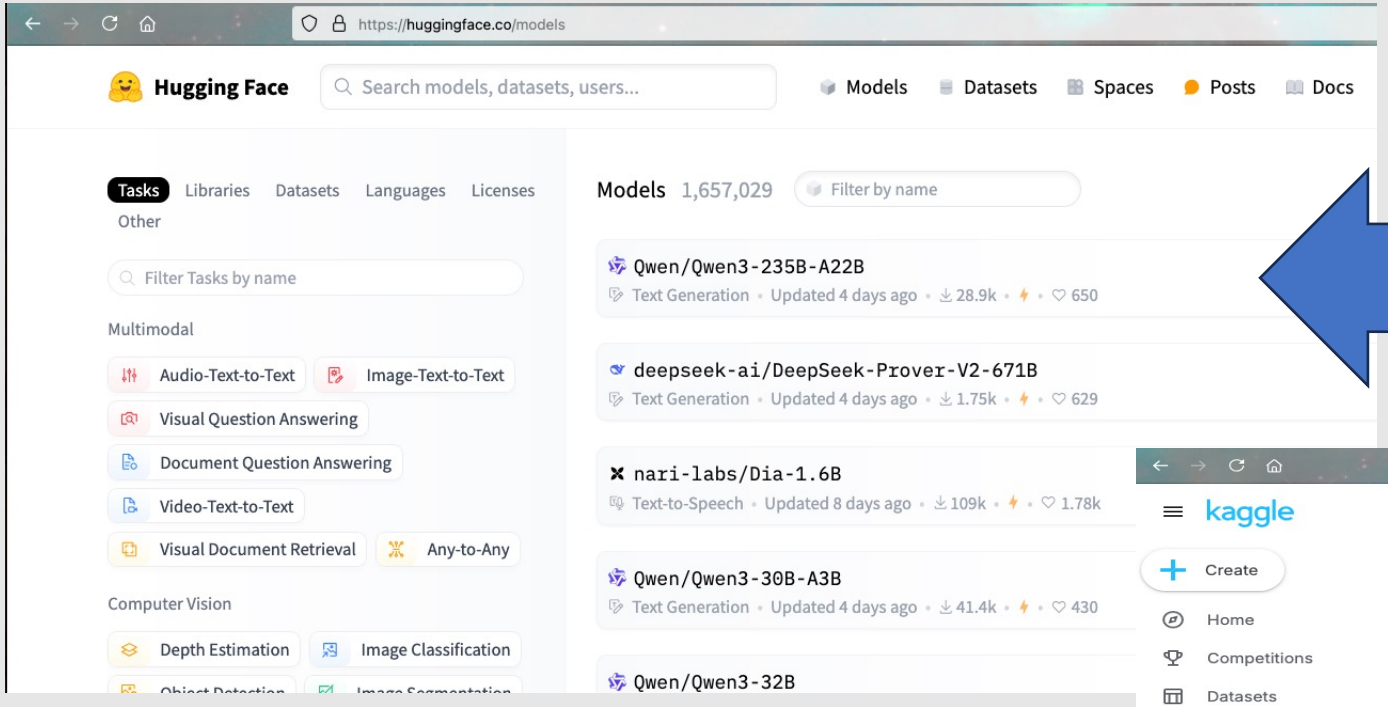


Running models locally

Why run models locally?

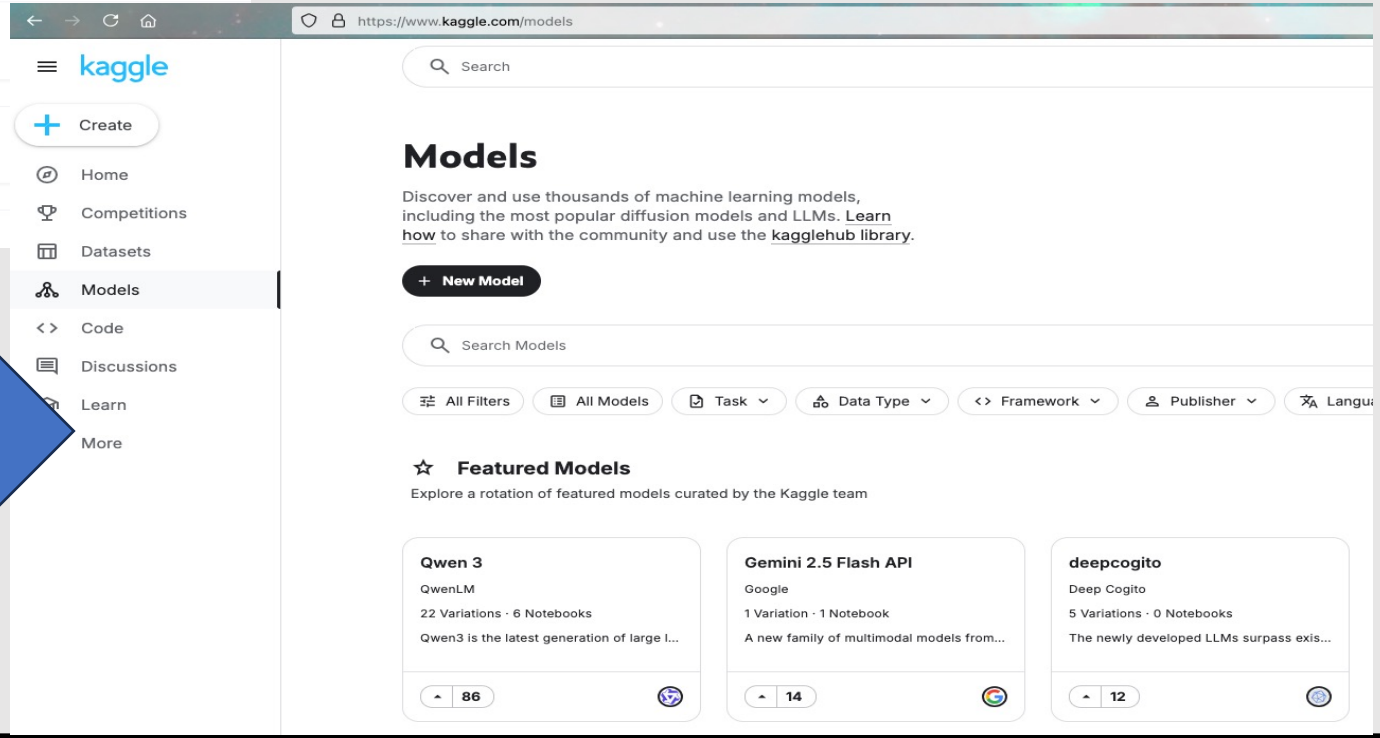
- Privacy - no need to share data
- Gives you control over setup, configuration, and customization options
  - Can tailor LLM to your needs, experiment with settings, integrate into your infra
- Can easily swap between different models for different tasks
- Work in offline mode
- Cost savings
  - No charges for subscriptions or API calls
- No censoring of results

Where to get models +



http://huggingface.co/models

http://kaggle.com/models



Options for running LLMs locally

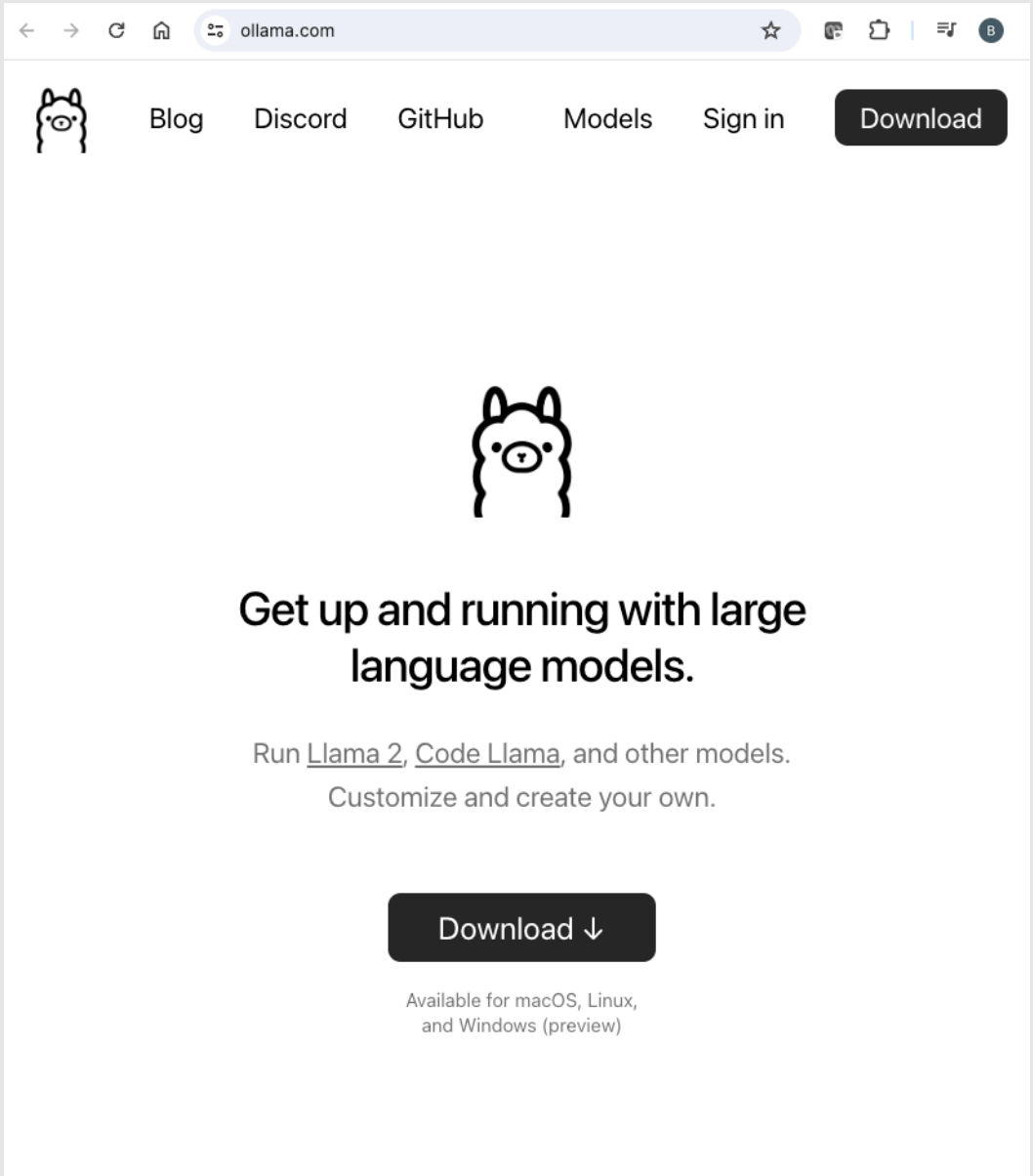
- GPT4All - https://github.com/nomic-ai/gpt4all
- LM Studio - https://lmstudio.ai
- Jan AI - https://jan.ai
- llama.cpp - https://github.com/ggerganov/llama.cpp
- LlamaFile - https://github.com/Mozilla-Ocho/llamafile
- Ollama - https://ollama.com/
- HuggingFace Transformers - https://huggingface.co/docs/transformers
- More!

LlamaFile
Local Executable GPT

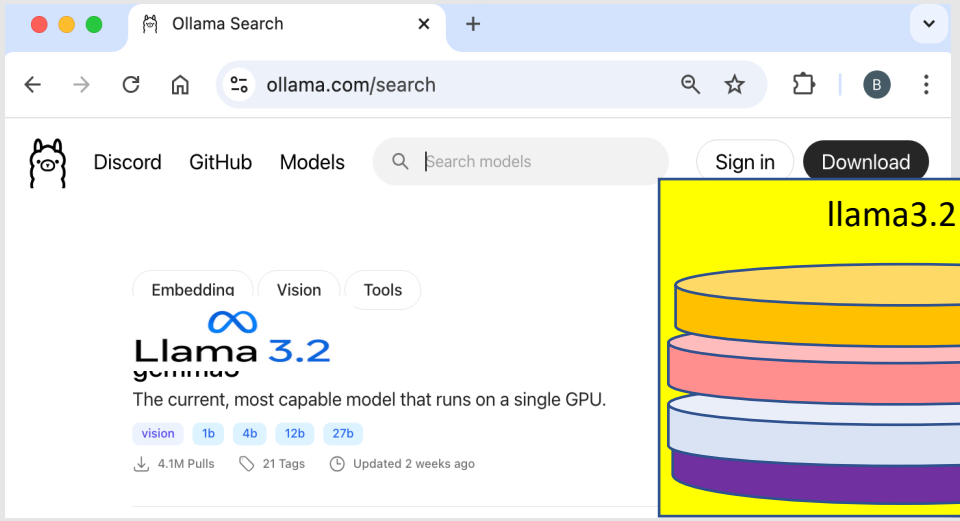
Transformers

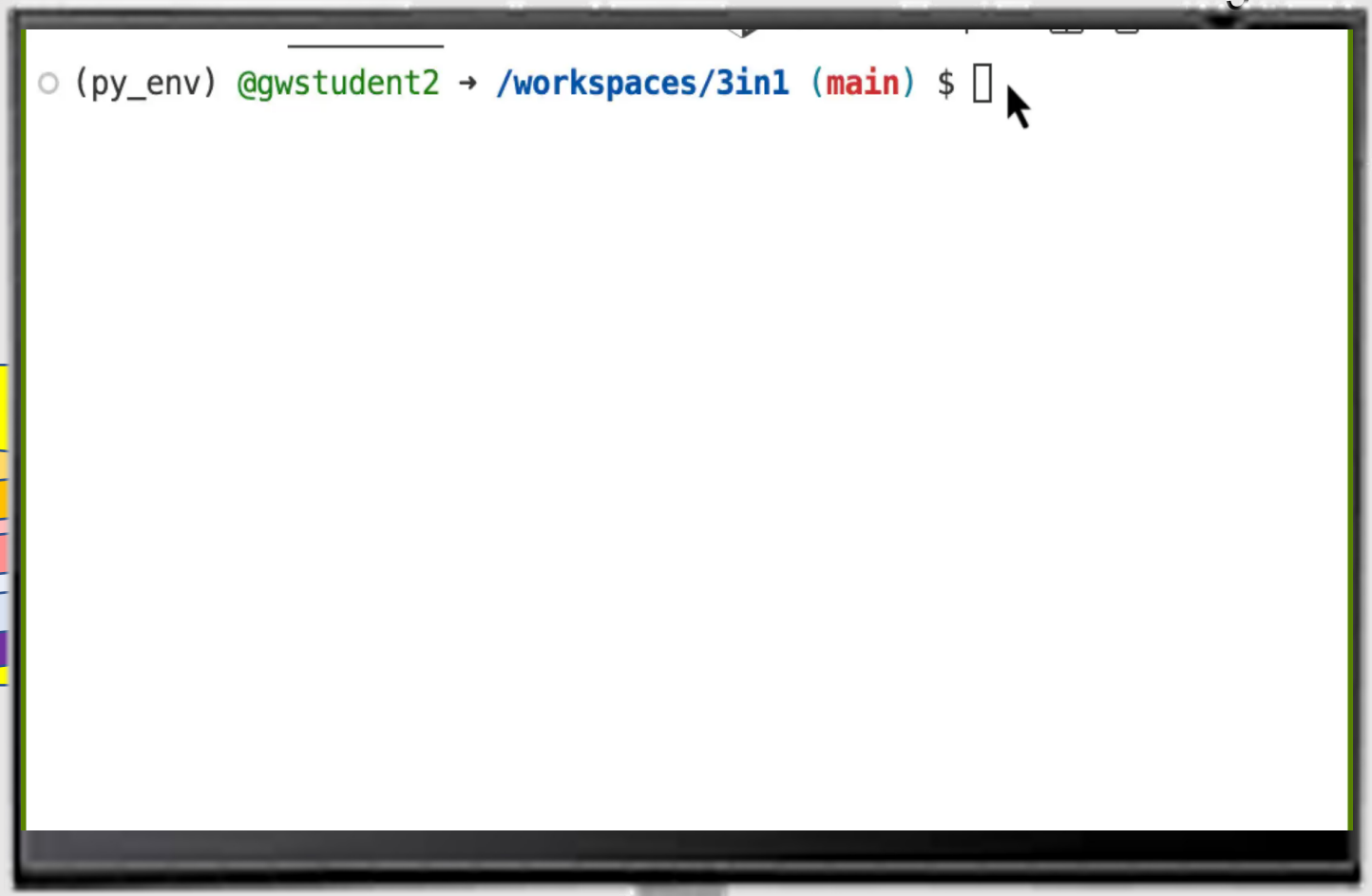
Ollama

- Command line tool for downloading, exploring and using LLMs on local machine
- open source
- supports most of Hugging Face's popular models
- allows uploading new ones
- Links:
  - main site: https://ollama.com
  - GitHub: https://github.com/ollama/
- Advantages
  - speeds up and simplifies
    - » model selection and download
    - » configuring endpoints
    - » integration with Python or JavaScript codebase



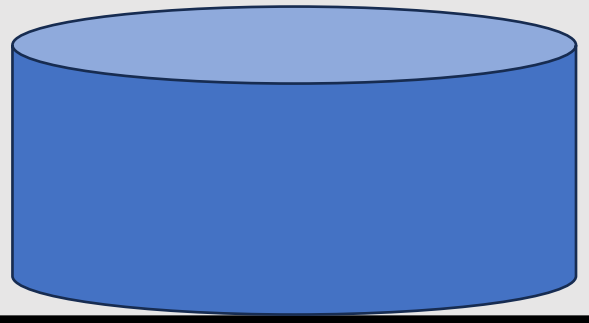
Working with Ollama #1



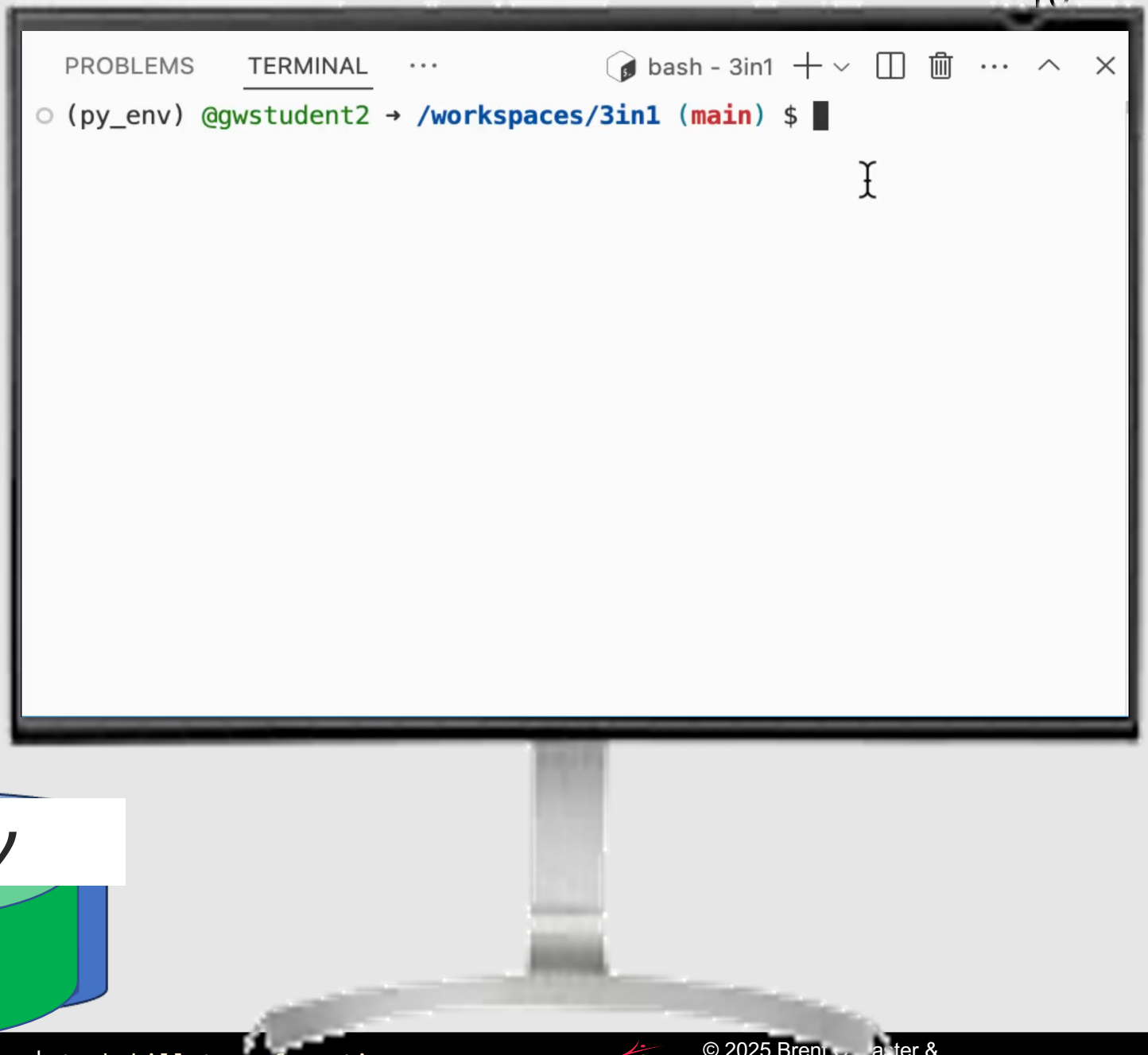


ollama pull

Ollama

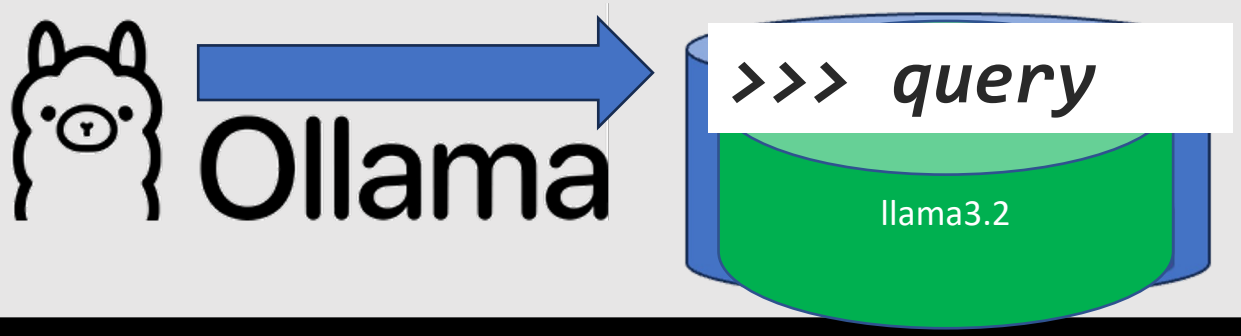


Working with Ollama #2



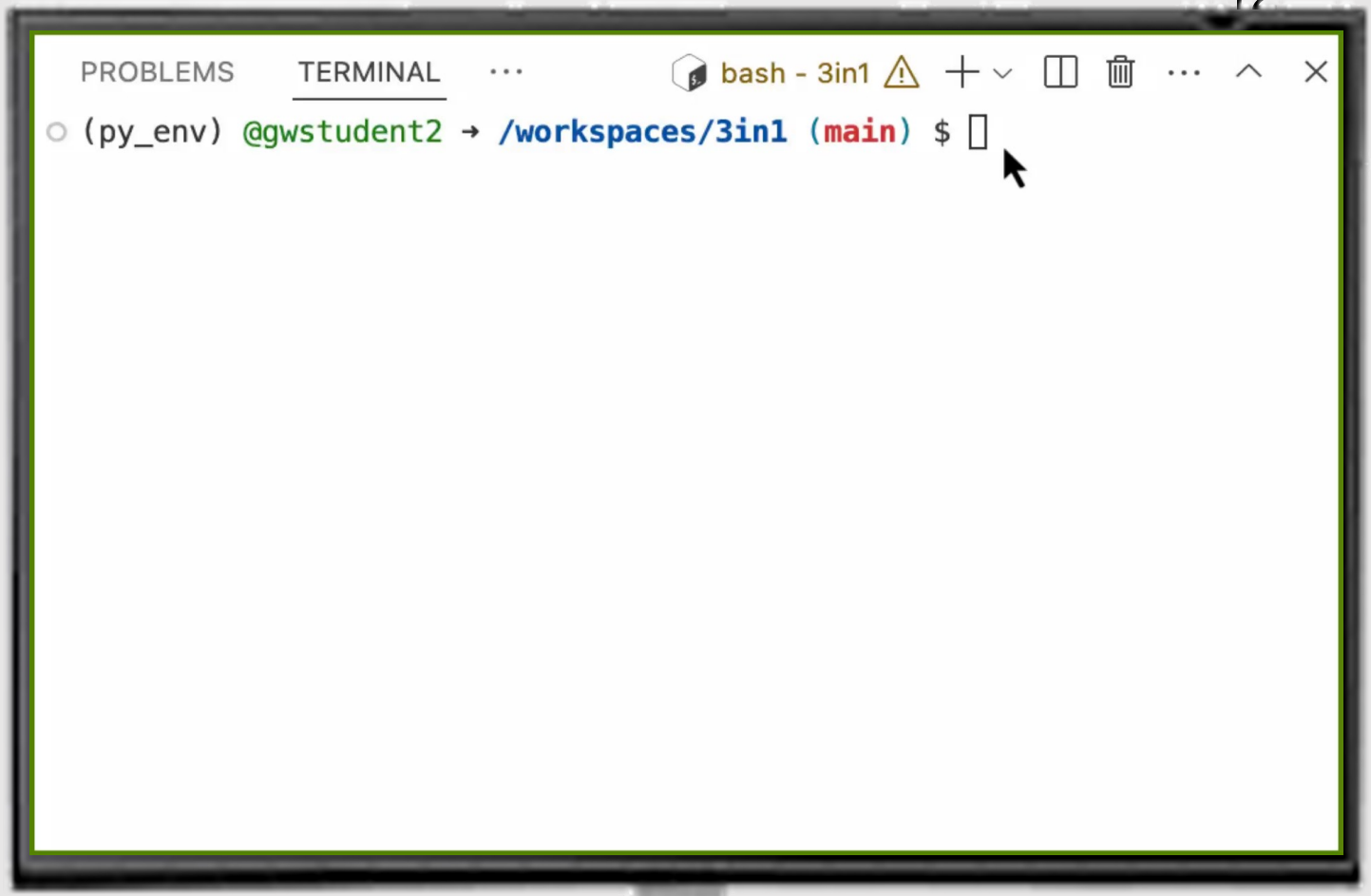

```
>>> Briefly explain what
an AI model is
```

`ollama run`

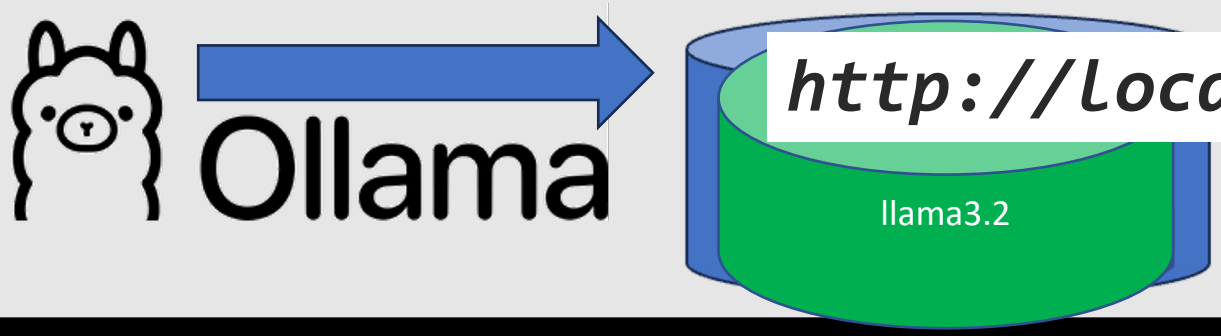


Working with Ollama #3

```python
from openai import OpenAI

# Initialize the client to talk to your local Ollama server
client = OpenAI(
    base_url="http://localhost:11434/v1",
    api_key="n-a"
)

# Send a chat completion request
completion = client.chat.completions.create(
    model="llama3.2",
    messages=[
        {"role": "system", "content": "Always answer in 3 bullet points."},
        {"role": "user",   "content": "Tell me what AI is."}
    ],
    temperature=0.7,
)

# Print out the assistant's reply
print(completion.choices[0].message.content)
```



ollama serve

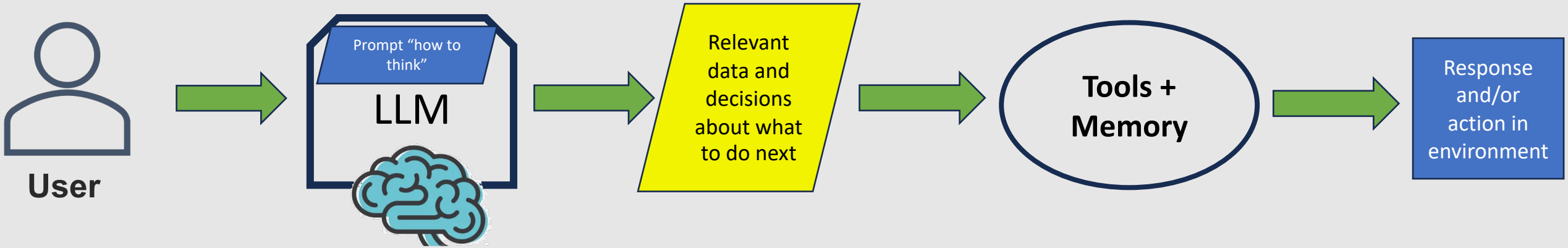


Demo #1 – Simple program to work with local model

Agents

What is an AI Agent?

- A **system** that operates within an **environment** by using **sensors** to **perceive** information, a **decision-making mechanism to process and reason about the data**, and **actuators to take actions that influence or update/respond to the environment**
- This interaction enables the agent to achieve specific goals autonomously while continuously learning and adapting over time
- Agents use LLMs to identify key data, drive decisions, and communicate naturally

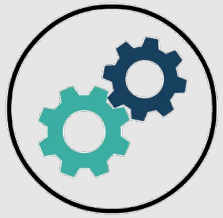


Architectural Features of AI Agents

Planning

- AI autonomously outlines and executes a logical series of steps for accomplishing a given objective.
- Provides the AI with a way to dynamically adapt its approach based on real-time data and feedback..
- Might employ reflection to evaluate and improve responses
- Example: A research agent plans search → summarize → generate report.

Tool Use



- AI agents interact with external APIs, databases, and functions.
- Enhances LLMs by providing access to real-world knowledge.
- Reduces hallucinations by using retrieval-augmented generation (RAG).
- Example: Calling a Python function to perform complex calculations.

Memory

- Short-term handles tasks; long term stores knowledge and experience
- Memory ensures consistency and efficiency in multi-step decisions
- Memory recalls preferences to enhance personalization and user experience
- Example: Storing user preferences for future reference or personalized responses

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.

You have access to the following tools:

Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string

You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.

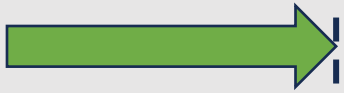
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:""""

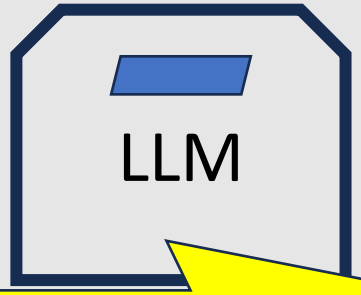
Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:"""

What's the weather in Paris?

User





Chain of Thought – Step 1: Interpret User Query
Thought: "The user is asking about the weather in Paris. I need to extract 'Paris' as the location.
Action: Extracted location = "Paris"

AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:"""

What's the weather in Paris?

User


```
AIResponse(
   tool_calls=[{
     name: "find_weather"
     parameters: {
        latitude: "48.8566",
        longitude: "2.3522",
     },
     id: "call_tool123",
     type: "tool_invoke"
  }]
)
```

Chain of Thought – Step 2: Decide to use tool
Thought: "I need real-time data, so I will call the '**find_weather**' tool. First, I need to get the latitude and longitude for the tool call.

Weather Search Tool

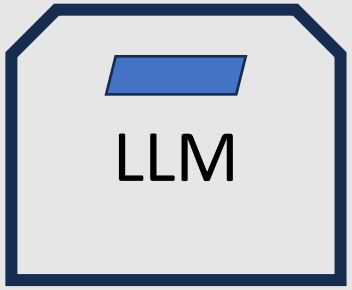
AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:"""

What's the weather in Paris?

User




```
AIResponse(
    tool_calls=[{
      name: "find_weather"
      parameters: {
        latitude: "48.8566",
        longitude: "2.3522",
      },
      id: "call_tool123",
      type: "tool_invoke"
    }]
)
```

Agent parses LLM output identifies JSON tool call, parses it, forms it into actual tool call

```
{
    name: "find_weather"
    parameters: {
        latitude: "48.8566",
        longitude: "2.3522",
    },
    id: "call_tool123",
    type: "tool_invoke"
}
```

Weather Search Tool

AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:"""

What's the weather in Paris?

User

LLM

```
AIResponse(
    tool_calls=[{
      name: "find_weather"
      parameters: {
         latitude: "48.8566",
         longitude: "2.3522",
      },
      id: "call_tool123",
      type: "tool_invoke"
   }]
)
```

Agent executes tool call

```
{
   name: "find_weather"
   parameters: {
      latitude: "48.8566",
      longitude: "2.3522",
   },
   id: "call_tool123",
   type: "tool_invoke"
}
```

Weather Search Tool

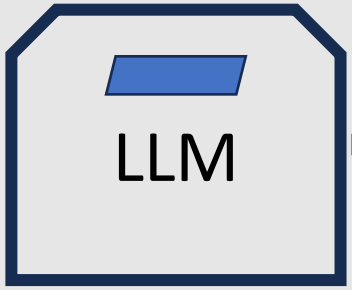
AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:""""

User

What's the weather in Paris?




```
AIResponse(
   tool_calls=[{
      name: "find_weather"
      parameters: {
         latitude: "48.8566",
         longitude: "2.3522",
      },
      id: "call_tool123",
      type: "tool_invoke"
   }]
)
```

```
{
   name: "find_weather"
   parameters: {
      latitude: "48.8566",
      longitude: "2.3522",
   },
   id: "call_tool123",
   type: "tool_invoke"
}
```

Weather tool returns result

Weather Search Tool

```
ToolResponse(
   content="53 and rainy",
   name="find_weather",
   tool_invoke_id: "call_tool123"
)
```

AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:""""

User

What's the weather in Paris?

LLM

```
AIResponse(
    tool_calls=[{
      name: "find_weather"
      parameters: {
         latitude: "48.8566",
         longitude: "2.3522",
      },
      "call_tool123",
      e: "tool_invoke"
```

Agent includes tool output in message/prompt back to model

```
{
   name: "find_weather"
   parameters: {
      latitude: "48.8566",
      longitude: "2.3522",
   },
   id: "call_tool123",
   type: "tool_invoke"
}
```

Weather Search Tool

```
ToolResponse(
   content="53 and rainy",
   name="find_weather",
   tool_invoke_id: "call_tool123"
)
```

AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:""""

What's the weather in Paris?

User

LLM

```
AIResponse(
   tool_calls=[{
     name: "find_weather"
     parameters: {
        latitude: "48.8566",
        longitude: "2.3522",
     },
     id: "call_tool123",
     type: "tool_invoke"
   }]
)
```

Chain of Thought – Step 3 : Interpret JSON Response
Thought: "The tool returned weather data for Paris. I will summarize the information concisely.

```
   name="find_weather",
   tool_invoke_id:
"call_tool123"
)
```

Weather Search Tool

```
{
   name: "find_weather"
   parameters: {
      latitude: "48.8566",
      longitude: "2.3522",
   },
   id: "call_tool123",
   type: "tool_invoke"
}
```

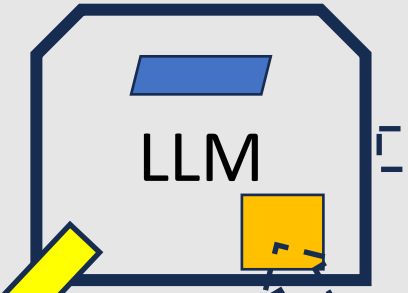
AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:"""

User

What's the weather in Paris?




```
AIResponse(
    tool_calls=[{
      name: "find_weather"
      parameters: {
        latitude: "48.8566",
        longitude: "2.3522",
      },
      id: "call_tool123",
      type: "tool_invoke"
    }]
)
```

```
{
    name: "find_weather"
    parameters: {
        latitude: "48.8566",
        longitude: "2.3522",
    },
    id: "call_tool123",
    type: "tool_invoke"
}
```

Weather Search Tool

```
ToolResponse(
    content="53 and rainy",
    name="find_weather",
    tool_invoke_id: "call_tool123"
)
```

```
AIFinalResponse(
    content="The current weather in Paris is 53 degrees Celsius with light rain."
)
```

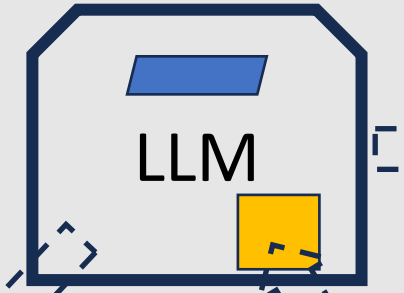
AI Agent

Agent Example

system_message="""You are an AI assistant designed to help users accurately and efficiently. Your primary goal is to provide precise, helpful, and clear responses.
You have access to the following tools:
Tool Name: find_weather, Description: Get weather for a location., Arguments: latitude: float, longitude: float, Outputs: string
You should think step by step in order to fulfill the objective with a reasoning process divided into Thought/Action/Observation. This cycle can repeat multiple times if needed.
You should first reflect with "Thought: {your_thoughts}" on the current query, then (if necessary), call a tool with the proper JSON formatting "Action: {JSON_BLOB}", or else print your final answer starting with the prefix "Final Answer:"""

User

What's the weather in Paris?




```
AIResponse(
    tool_calls=[{
      name: "find_weather"
      parameters: {
         location: "Paris",
      },
      id: "call_tool123",
      type: "tool_invoke"
   }]
)
```

```
{
    name: "find_weather"
    parameters: {
       latitude: "48.8566",
       longitude: "2.3522",
    },
    id: "call_tool123",
    type: "tool_invoke"
}
```

Weather Search Tool

```
ToolResponse(
    content="53 and rainy",
    name="find_weather",
    tool_invoke_id: "call_tool123"
)
```

```
AIFinalResponse(
    content="The current weather in Paris is 53 degrees Celsius with light rain."
)
```

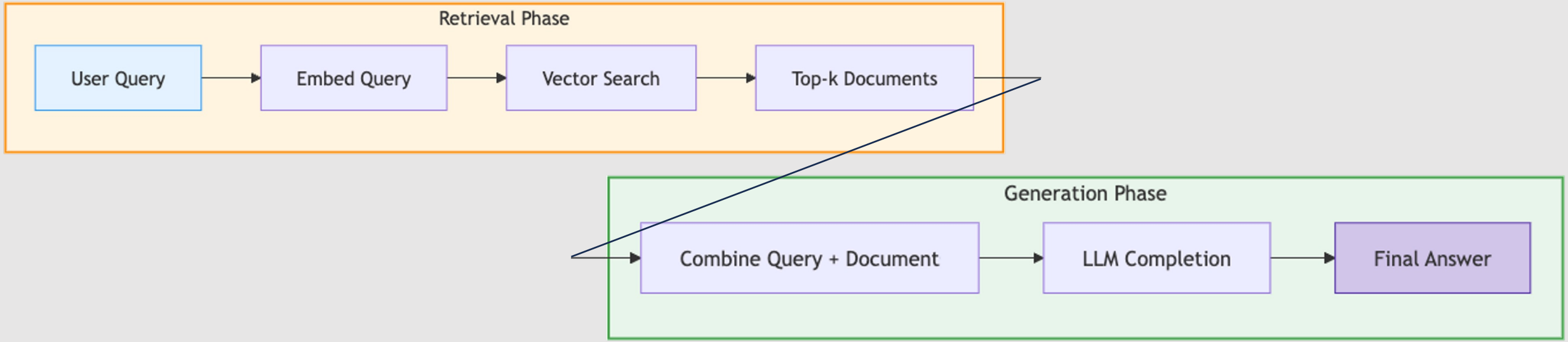
AI Agent

Demo #2 – Adding agency to our code

RAG

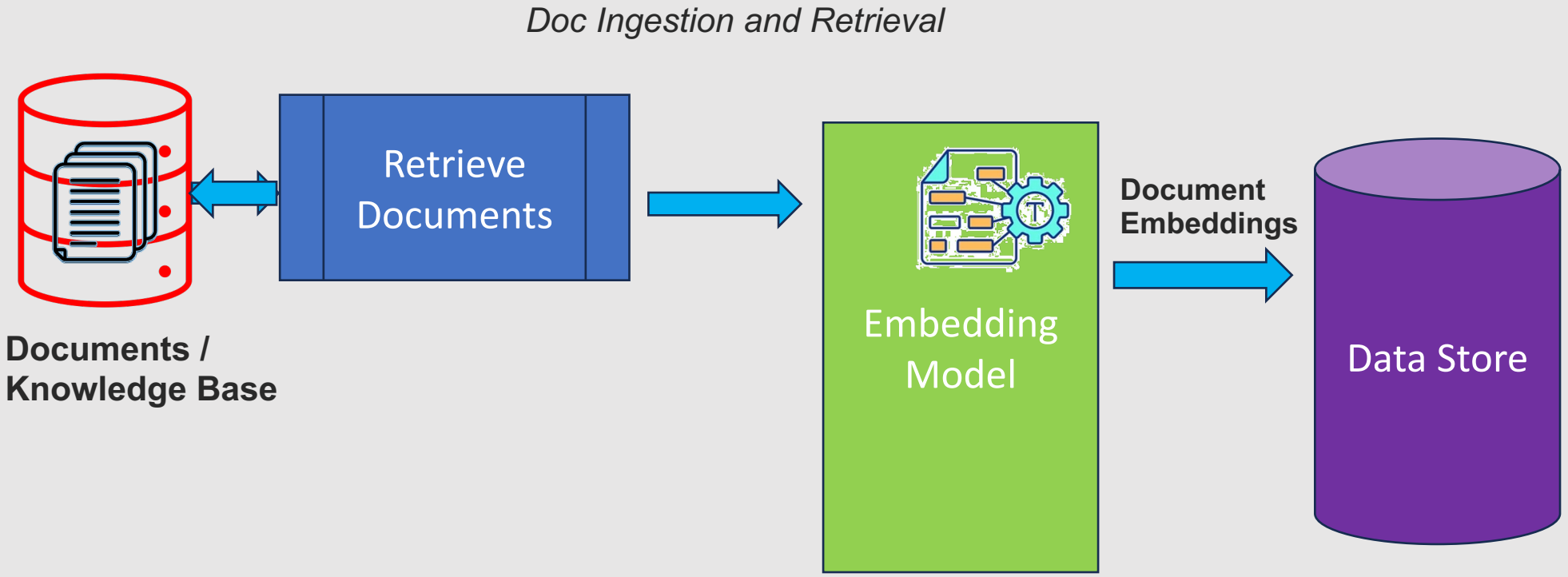
What is RAG and how does it work?

- **Combination of retrieval and generation**: RAG combines information retrieval (like a search engine) with text generation (like a language model).
- **Uses external knowledge**: Instead of relying solely on pre-trained knowledge, RAG retrieves relevant documents or data from an external source (like a database or private knowledge bases) to generate more accurate and up-to-date responses.
- **Improves factual accuracy**: By pulling in real-time data or documents, RAG reduces the risk of generating factually incorrect or outdated information.
- **Two-step process**:
  - **Retrieve**: The model searches for relevant information from a knowledge source.
  - **Generate**: It then uses the retrieved data to create a coherent, contextually accurate answer.



Source: https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/

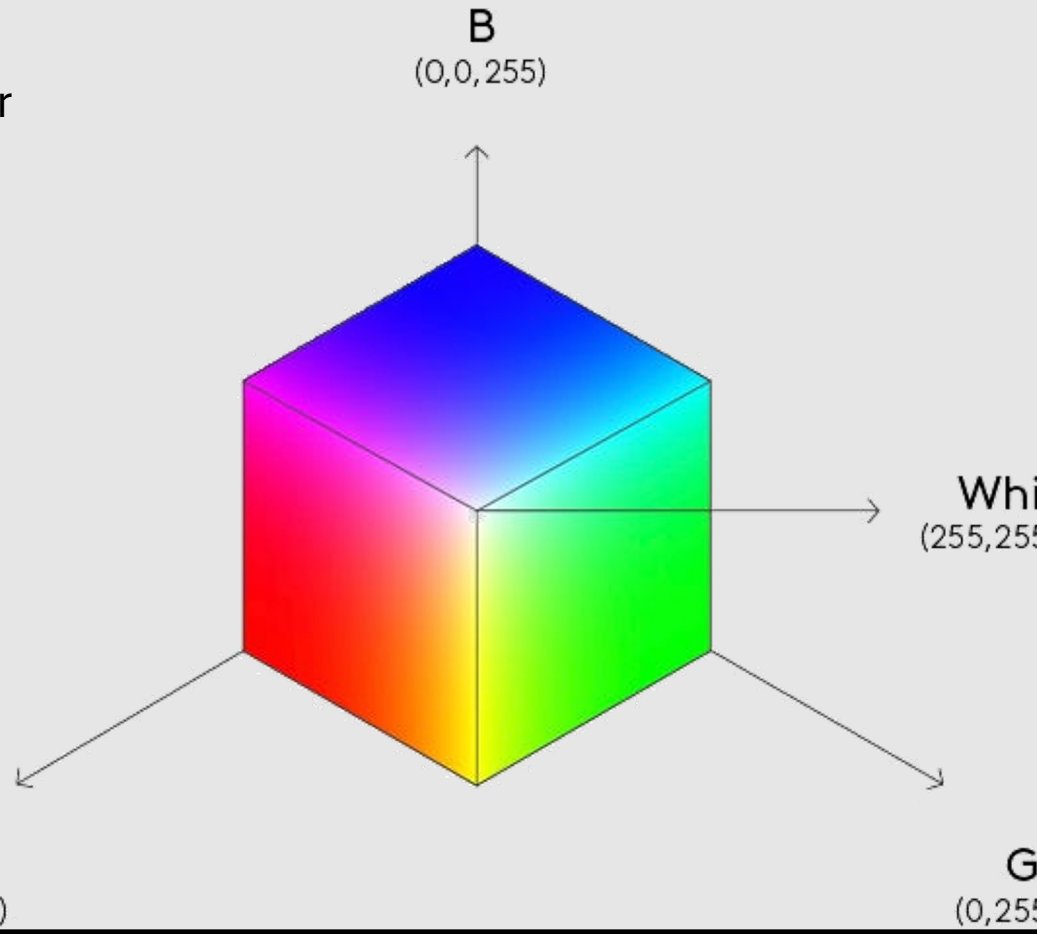
How is RAG setup?



- You provide data sources and point application to them
- Info is retrieved from the data sources and tokenized, embedded and stored in a data store
- For queries/prompts, application gathers results (most relevant ones) from the vector database with your data

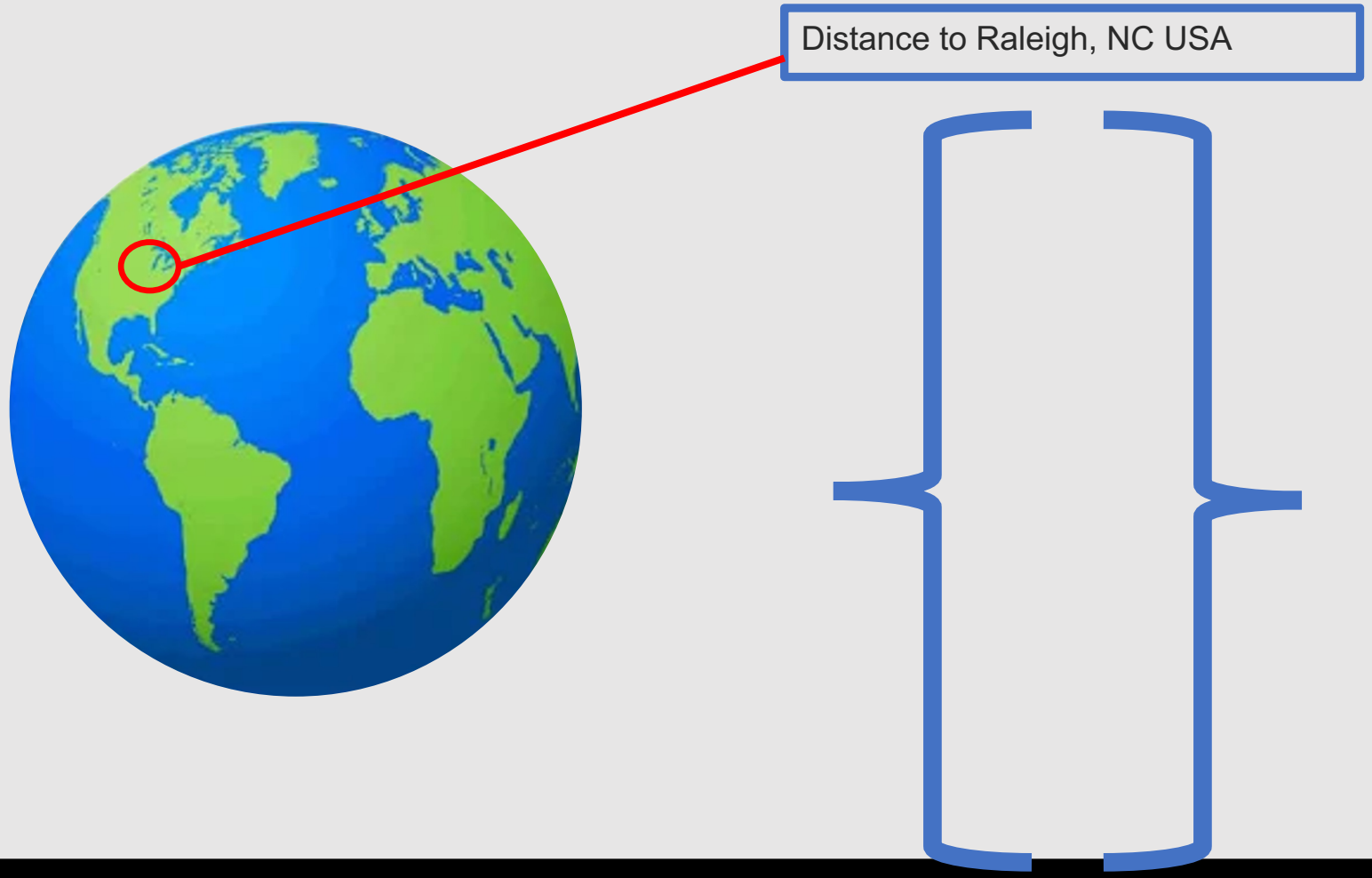
Embeddings

- Embeddings represent text as sets of numeric data - tensors (lots of dimensions)
- Each dimension stores some info about the text's meaning, context, or syntactical aspects
- Words or sentences with similar meanings are stored closer together in the vector space
  - If two pieces of text are similar syntactically, they will have similar embeddings (smaller distance between their vectors)
- During training, models learn to place text with similar meanings closer together in the embedding space
- Common pre-trained models used for generating embeddings include BERT and variants (RoBERTa, DistilBERT)
- Once you have embeddings, you can use them for NLP tasks like semantic search, text classification, sentiment analysis



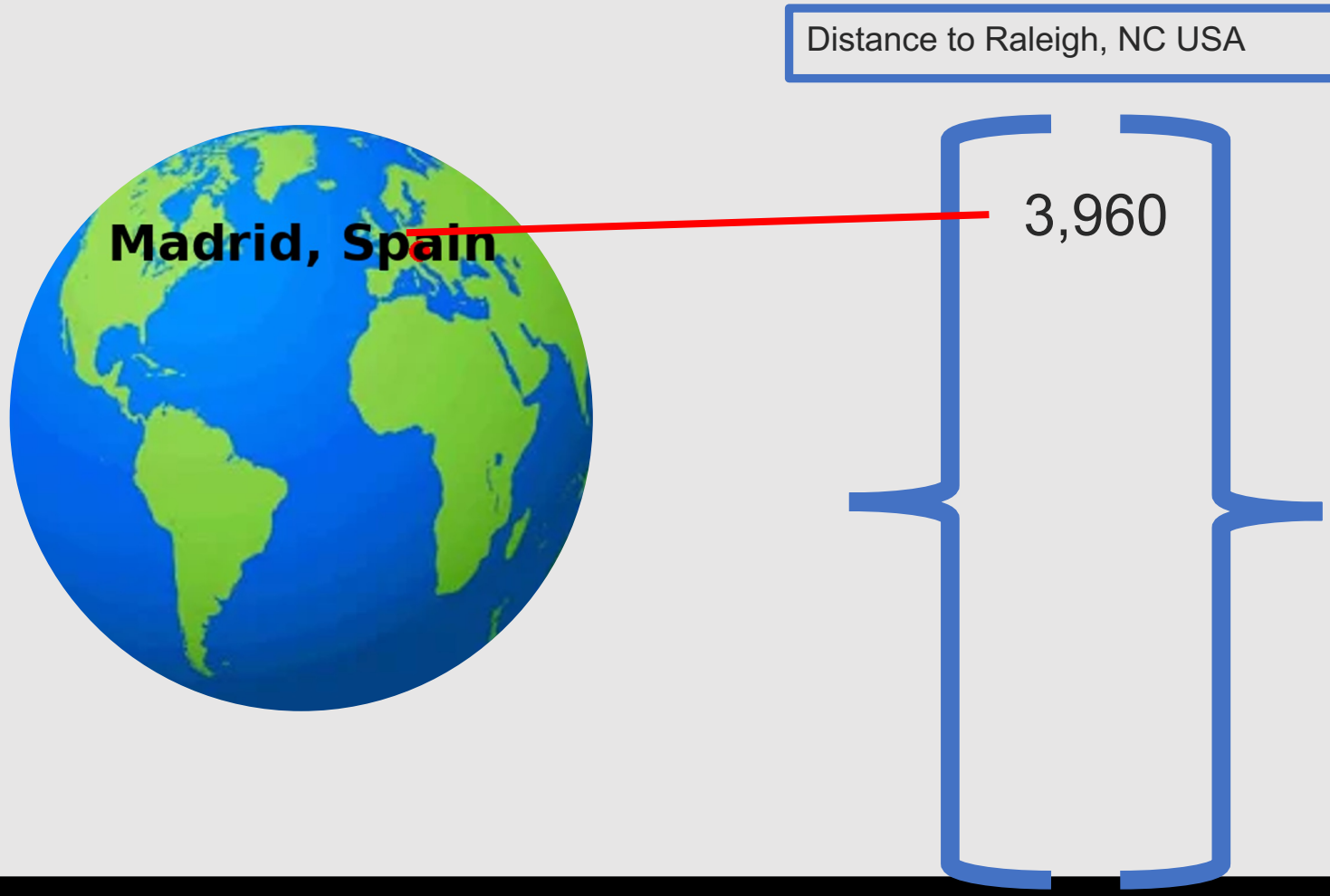
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



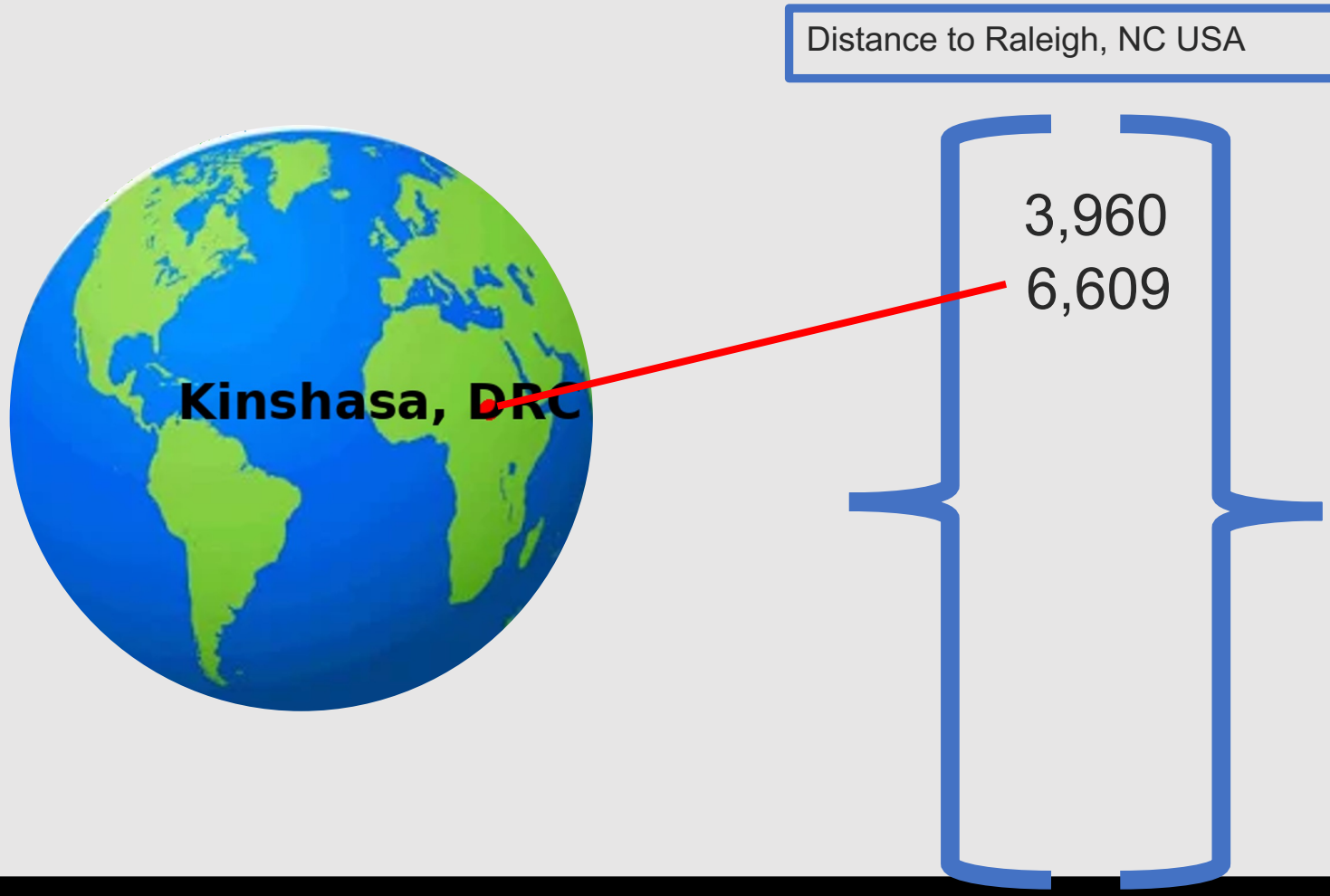
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



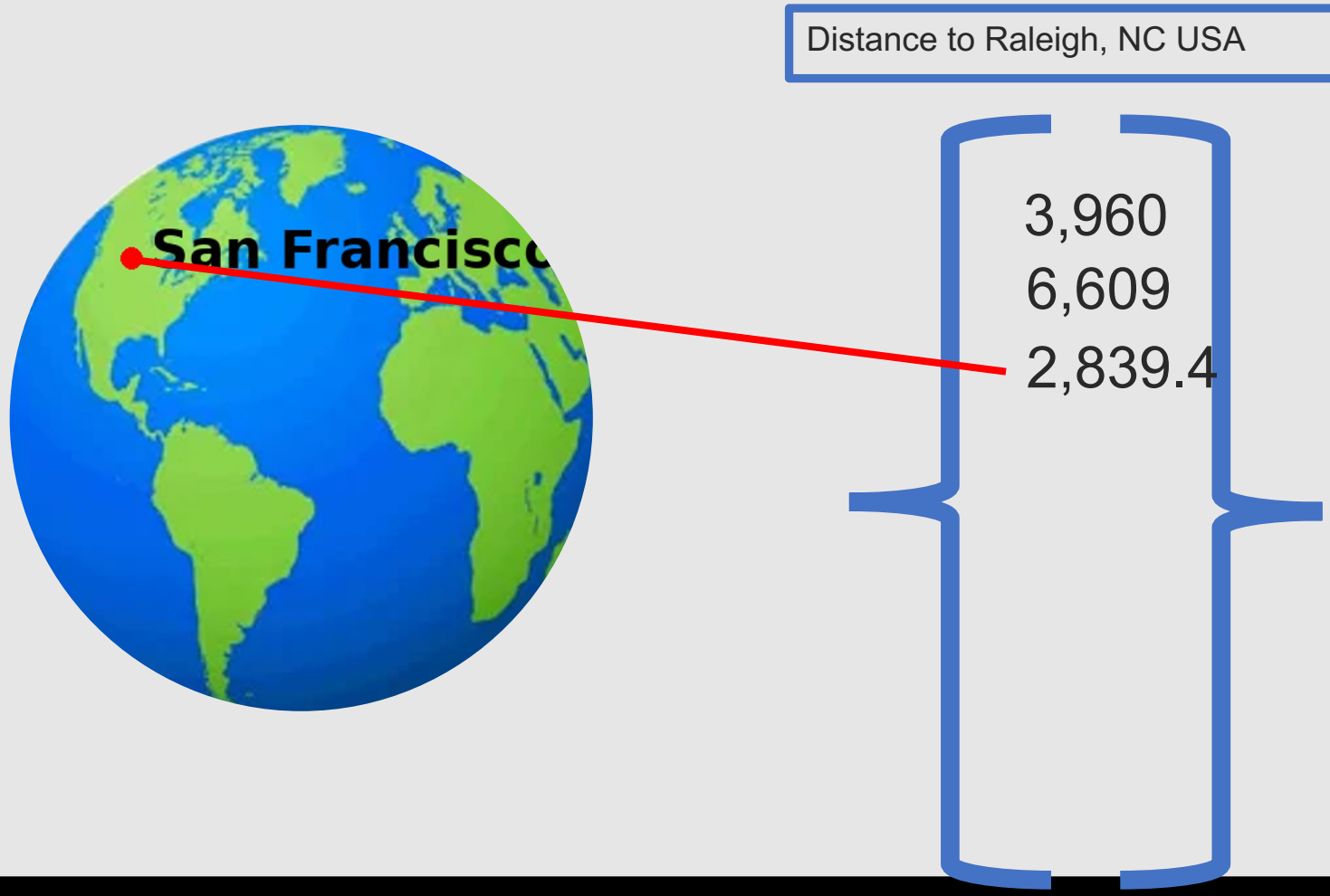
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



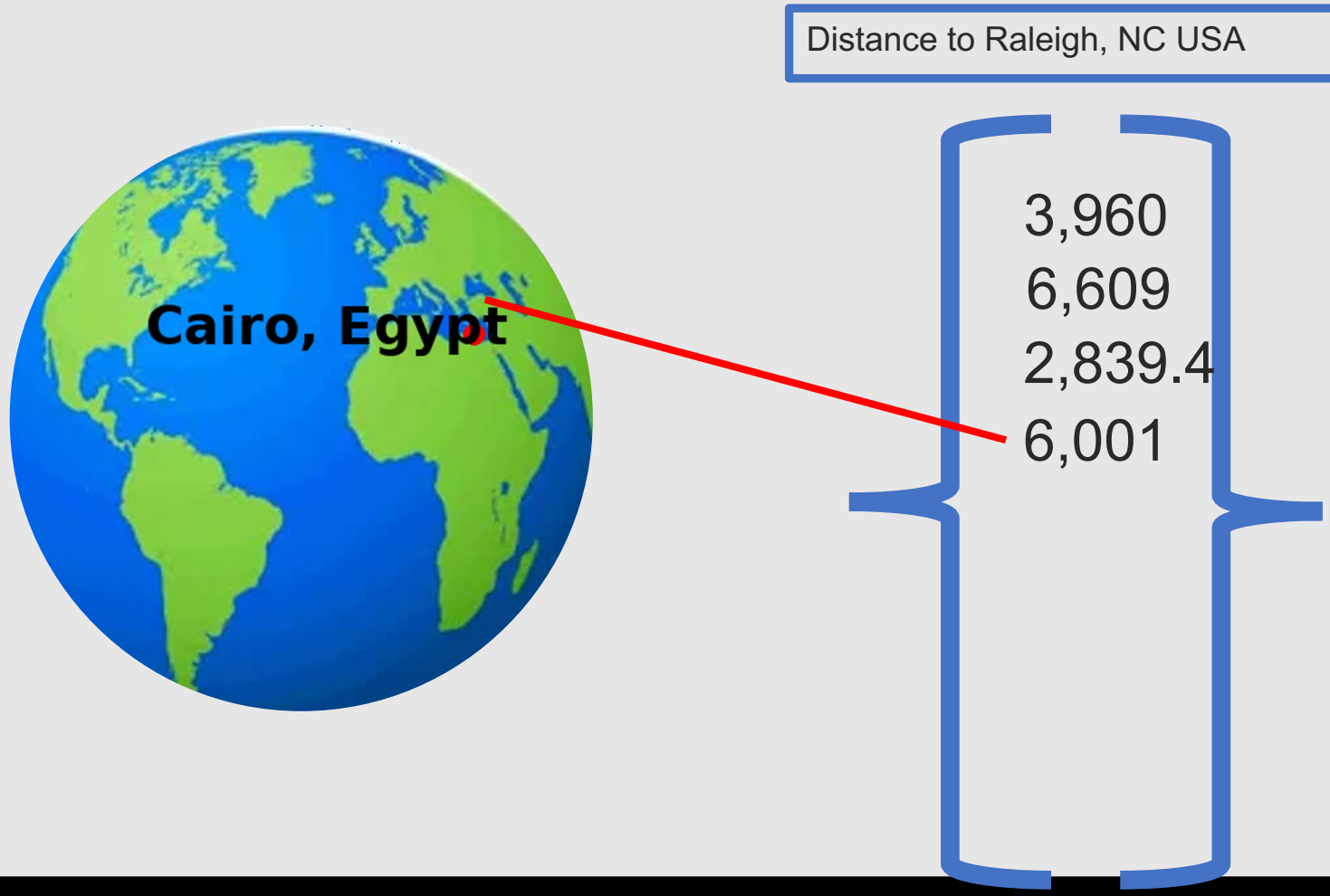
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



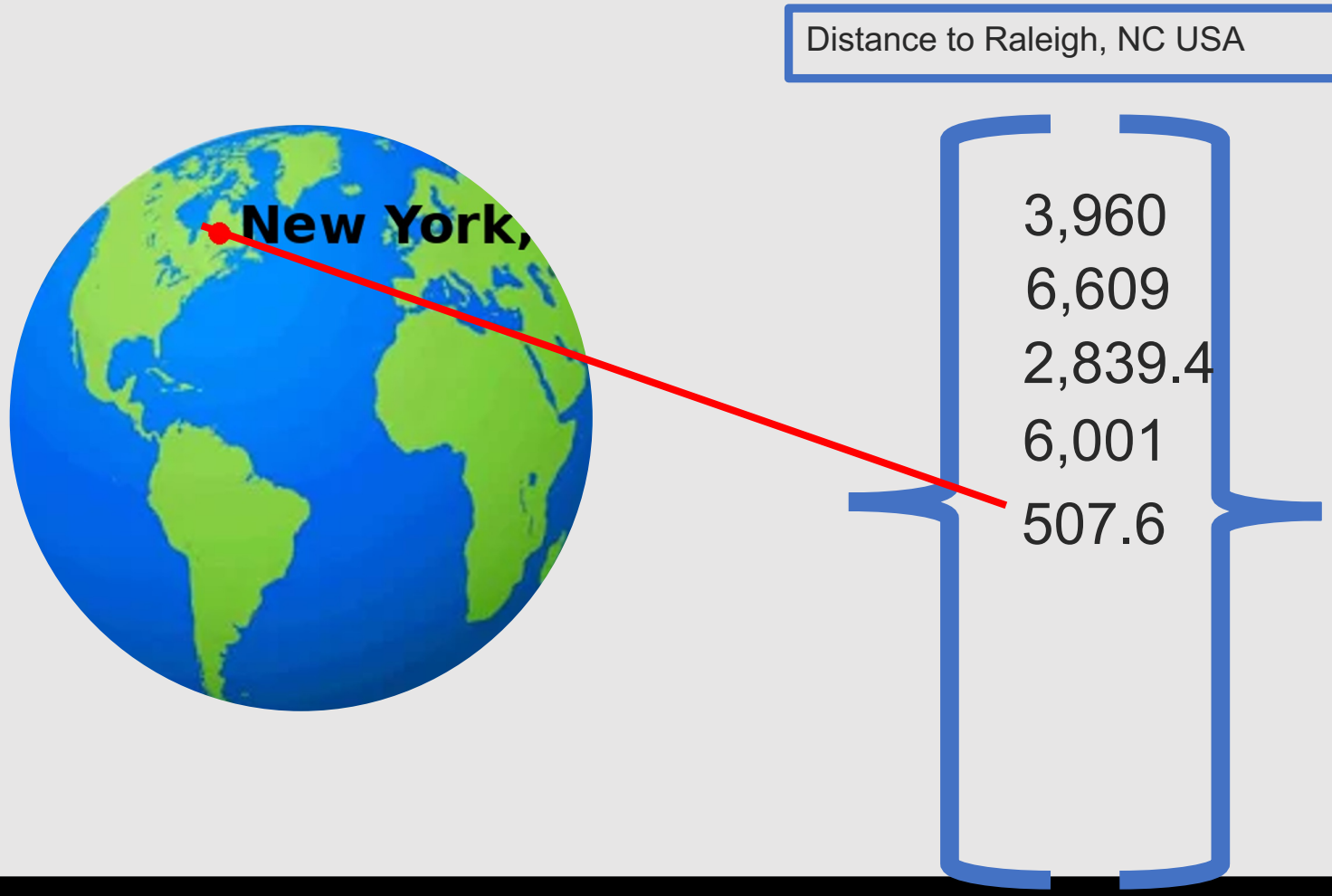
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



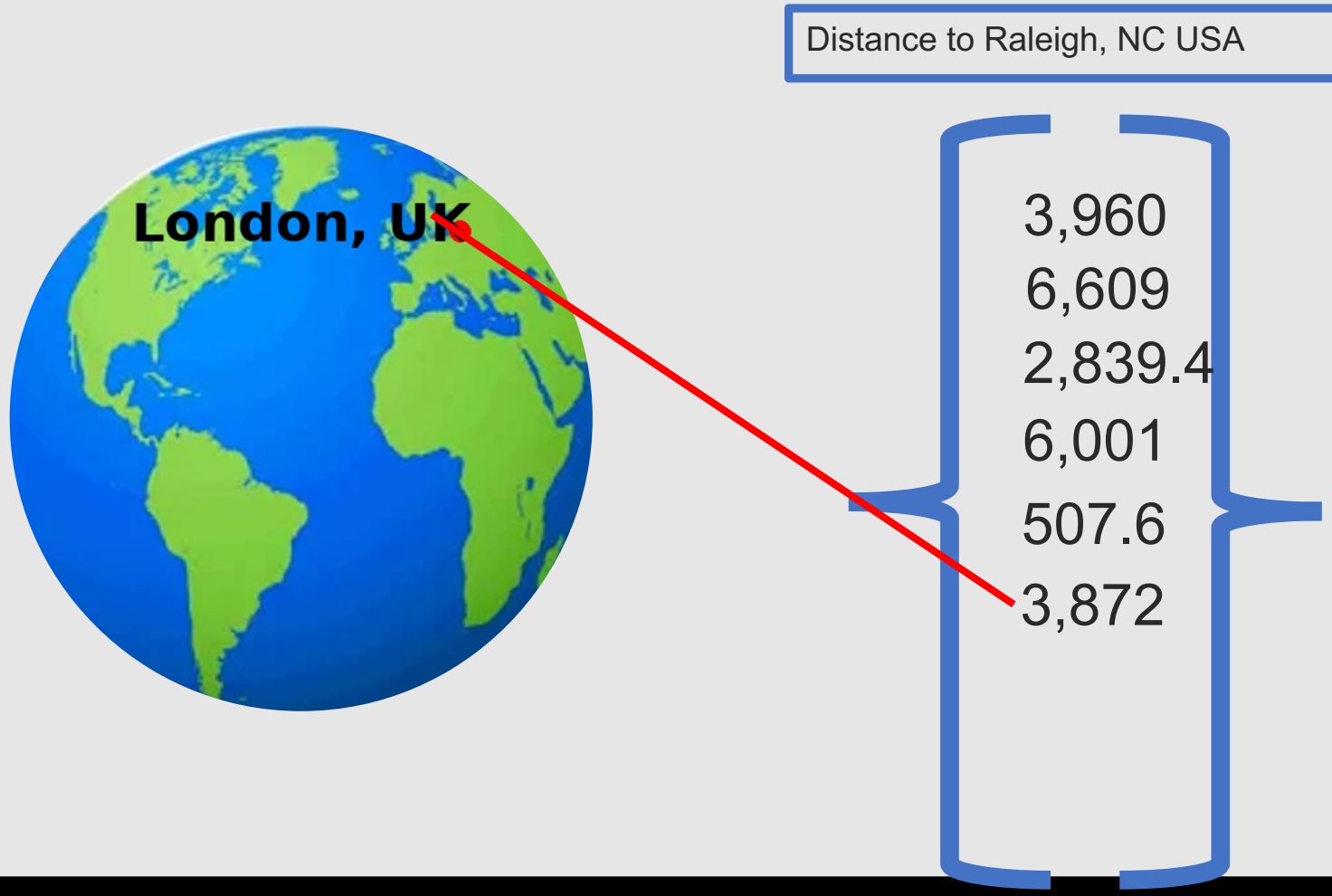
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



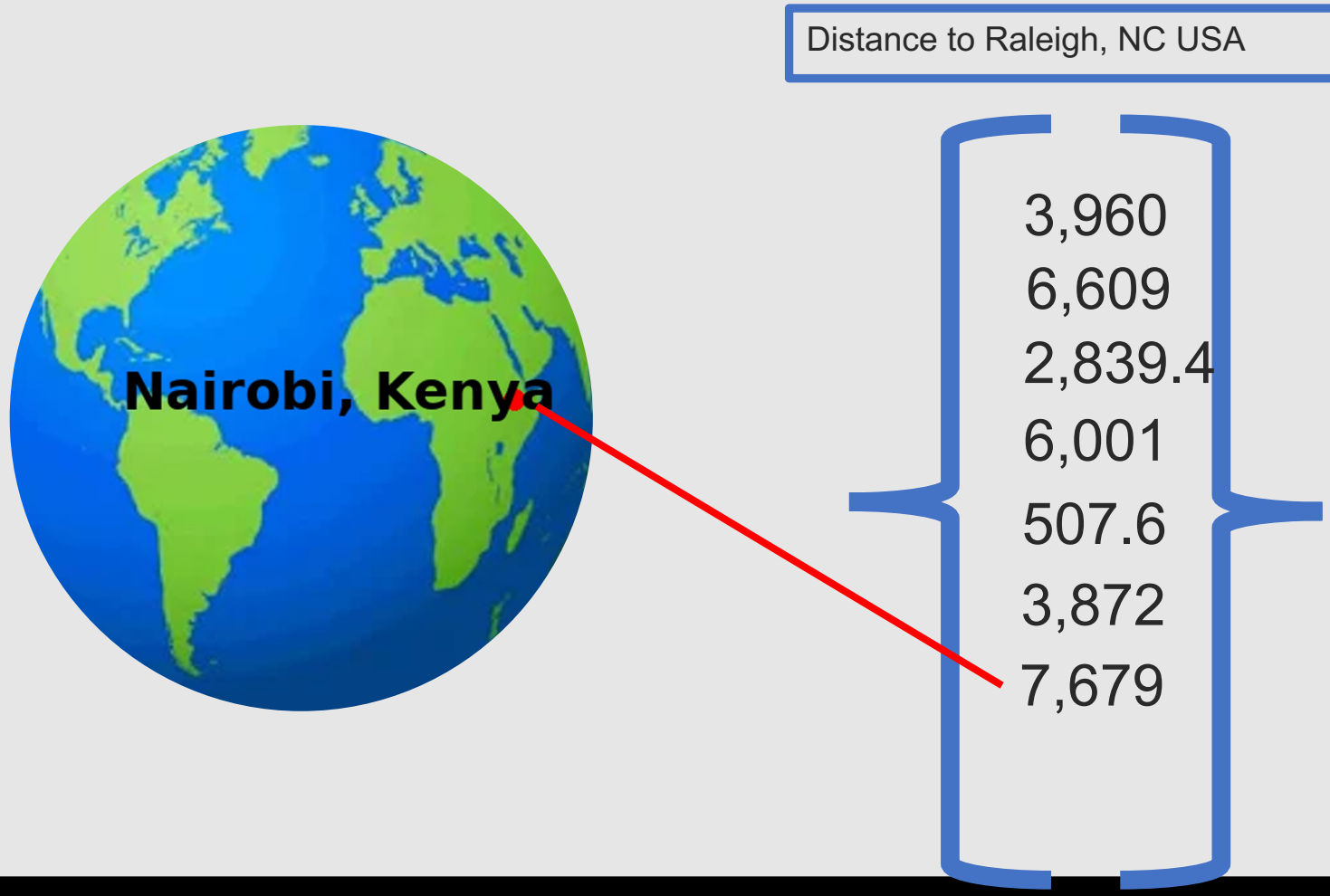
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



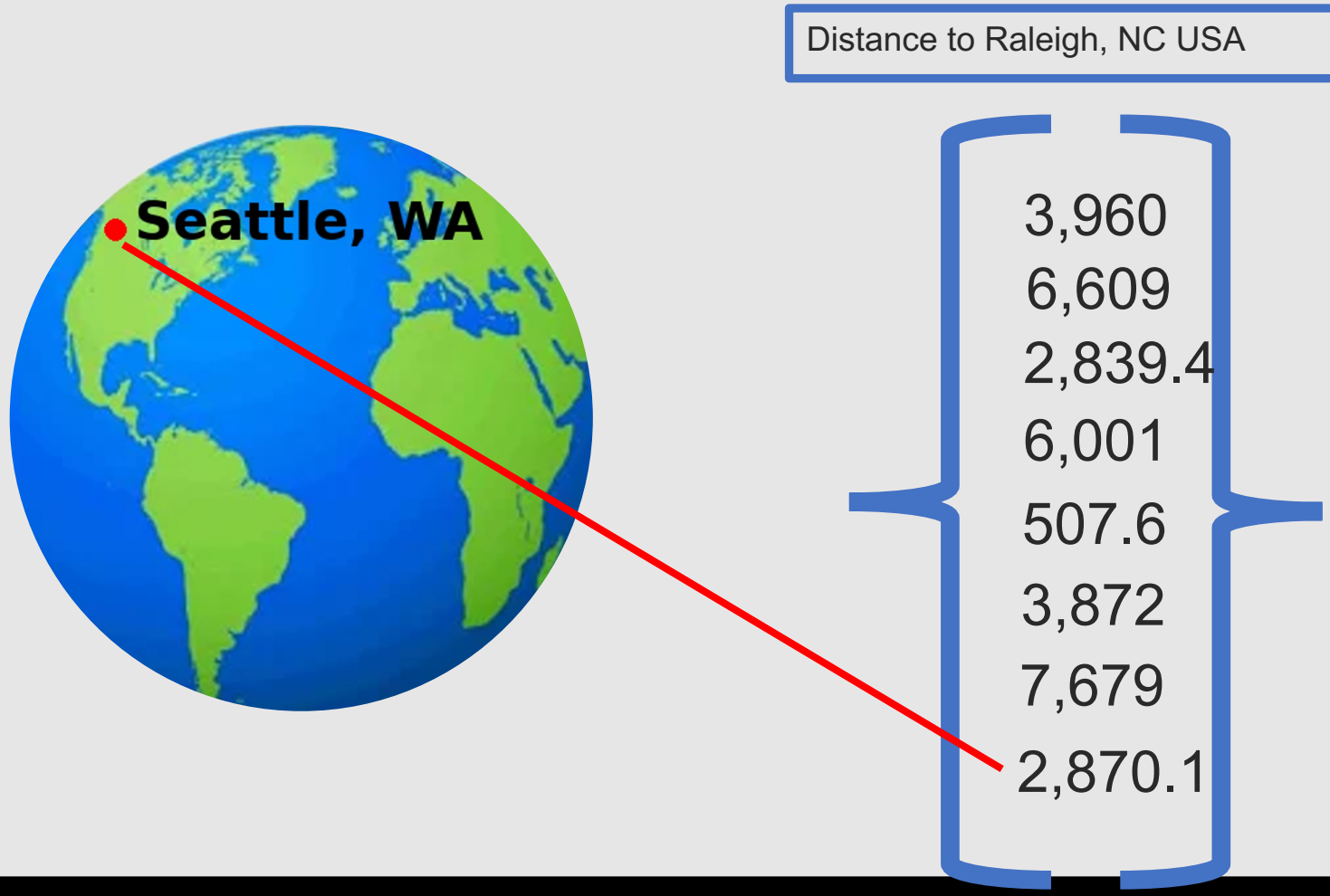
Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



Understanding vectors in AI

- Collection of data points that encapsulate an item's relationship to other items



Semantic meaning / relationships

- Suppose we have 3 words
- King and Queen are more similar to each other than they are to lunch
- In order for neural net to understand the relationships, each word needs to be represented as a vector
- Suppose each word is represented by a 2-dimensional vector

King $\begin{bmatrix} -130.16 \\ 89.5 \end{bmatrix}$

Queen $\begin{bmatrix} -115.43 \\ 95.2 \end{bmatrix}$

Lunch $\begin{bmatrix} -89.5 \\ 34.3 \end{bmatrix}$

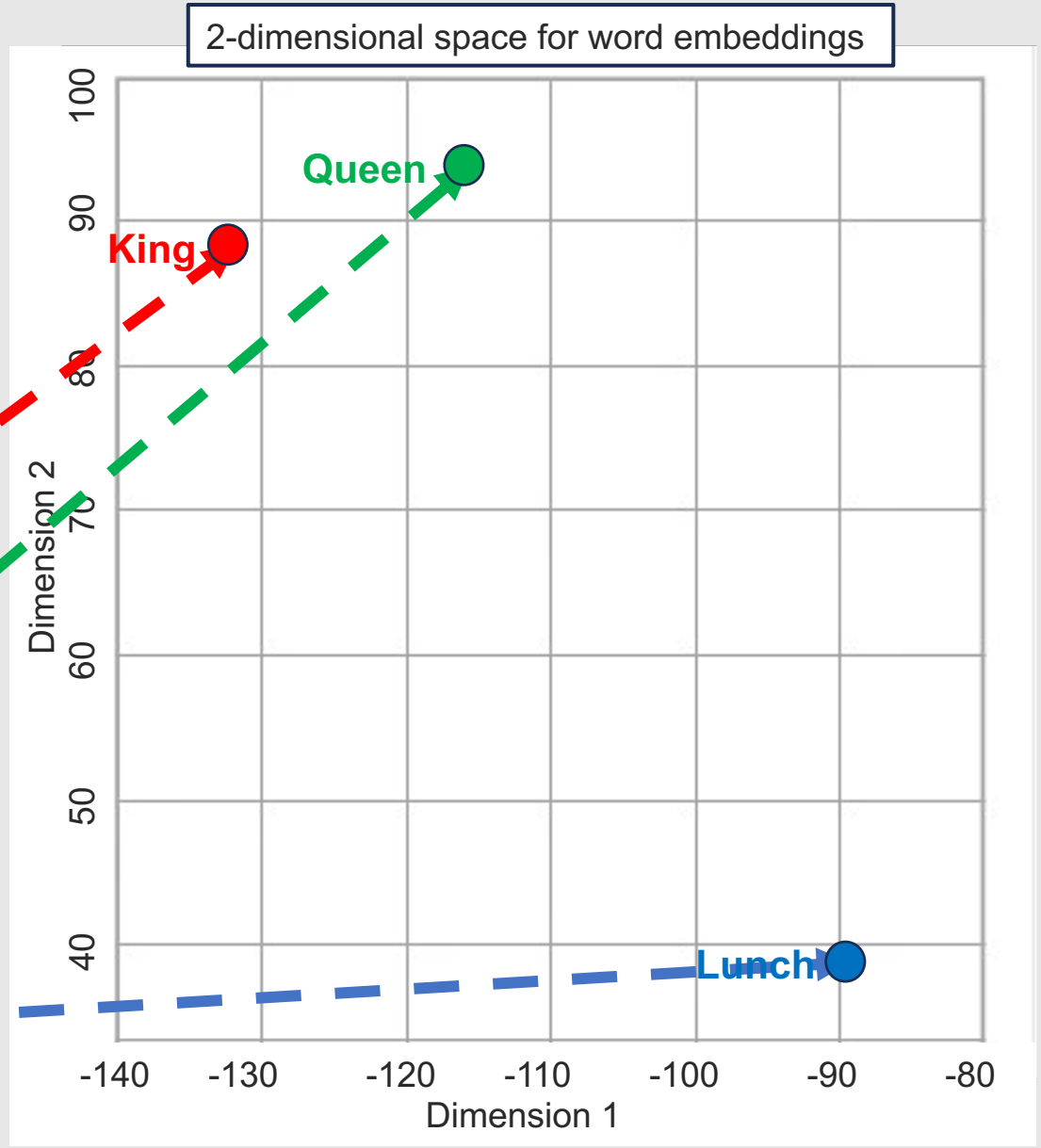
Embedding space

- Plotting in 2-dimensional embedding space shows relationships
- Way to let NN understand relationships between words
- We want the NN to learn that King and Queen are more similar to each other than they are to lunch

King [-130.16, 89.5]

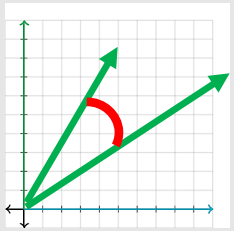
Queen [-115.43, 95.2]

Lunch [-89.5, 34.3]

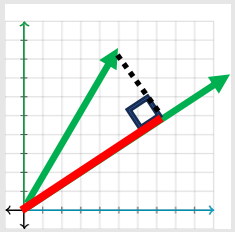


Searching for Vectors - similarity metrics

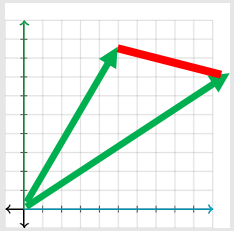
- 3 metrics commonly used to determine similarity of two vectors (2-dimensional representation)



Cosine similarity - measure the angle between two vectors; values from -1 to 1; 1 = both point in same direction; -1 point in opposite directions; 0 = orthogonal (perpendicular)



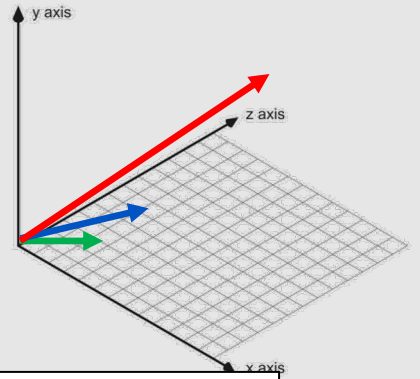
Dot product / inner product - measures how well 2 vectors align with each other; values from - ∞ to ∞; positive values indicate vectors are in same direction; negative values indicate opposite directions; 0 = orthogonal



Euclidean distance - measures the distance between two vectors; values from 0 to ∞; 0 = identical; larger numbers farther apart

imagine 3 vectors - a,b,c

$$a=\begin{bmatrix}.01\\.07\\.1\end{bmatrix} \quad b=\begin{bmatrix}.01\\.08\\.11\end{bmatrix} \quad c=\begin{bmatrix}.91\\.57\\.6\end{bmatrix}$$



Cosine similarity

$$sim(u,v)=\frac{u\cdot v}{\|u\|\|v\|}=\frac{\sum_{i=1}^{n}a_n b_n}{\sqrt{\sum_{i=1}^{n}u_n^2}\sqrt{\sum_{i=1}^{n}v_n^2}}$$

$$sim(a,b)=\frac{(a_1*b_1)+(a_2*b_2)+(a_3*b_3)}{\sqrt{a_1^2+a_2^2+a_3^2}\sqrt{b_1^2+b_2^2+b_3^2}}$$

$$=\frac{(0.01*0.01)+(0.07*0.08)+(0.1*0.11)}{\sqrt{0.01^2+0.07^2+0.1^2}\sqrt{0.01^2+0.08^2+0.11^2}}$$

0.0141

Dot product / inner product

$$u\cdot v=|u||v|cos\theta=\sum_{i=1}^{n}a_n b_n$$

Dot product formula

$$a\cdot b=(a_1b_1)+(a_2b_2)+(a_3b_3)$$

$$=(0.01*0.01)+(0.07*0.08)+(0.1*0.11)$$

0.0167

Euclidean distance

$$d(u,v)=\sqrt{\sum_{i=1}^{n}(u_i-v_i)^2}$$

Euclidean distance formula

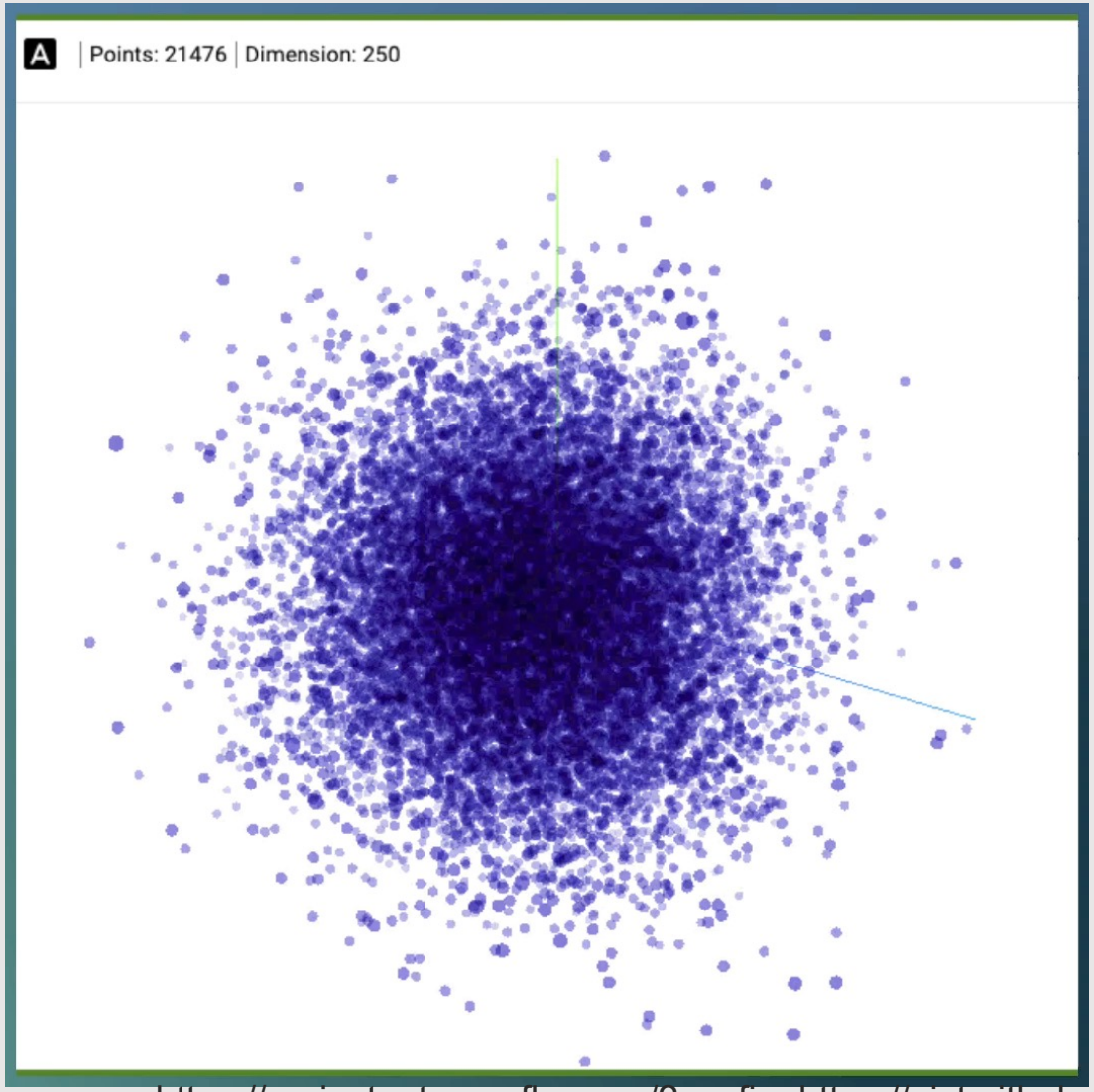
$$d(a,b)=\sqrt{(b_1-a_1)^2+(b_2-a_2)^2+(b_3-a_3)^2}$$

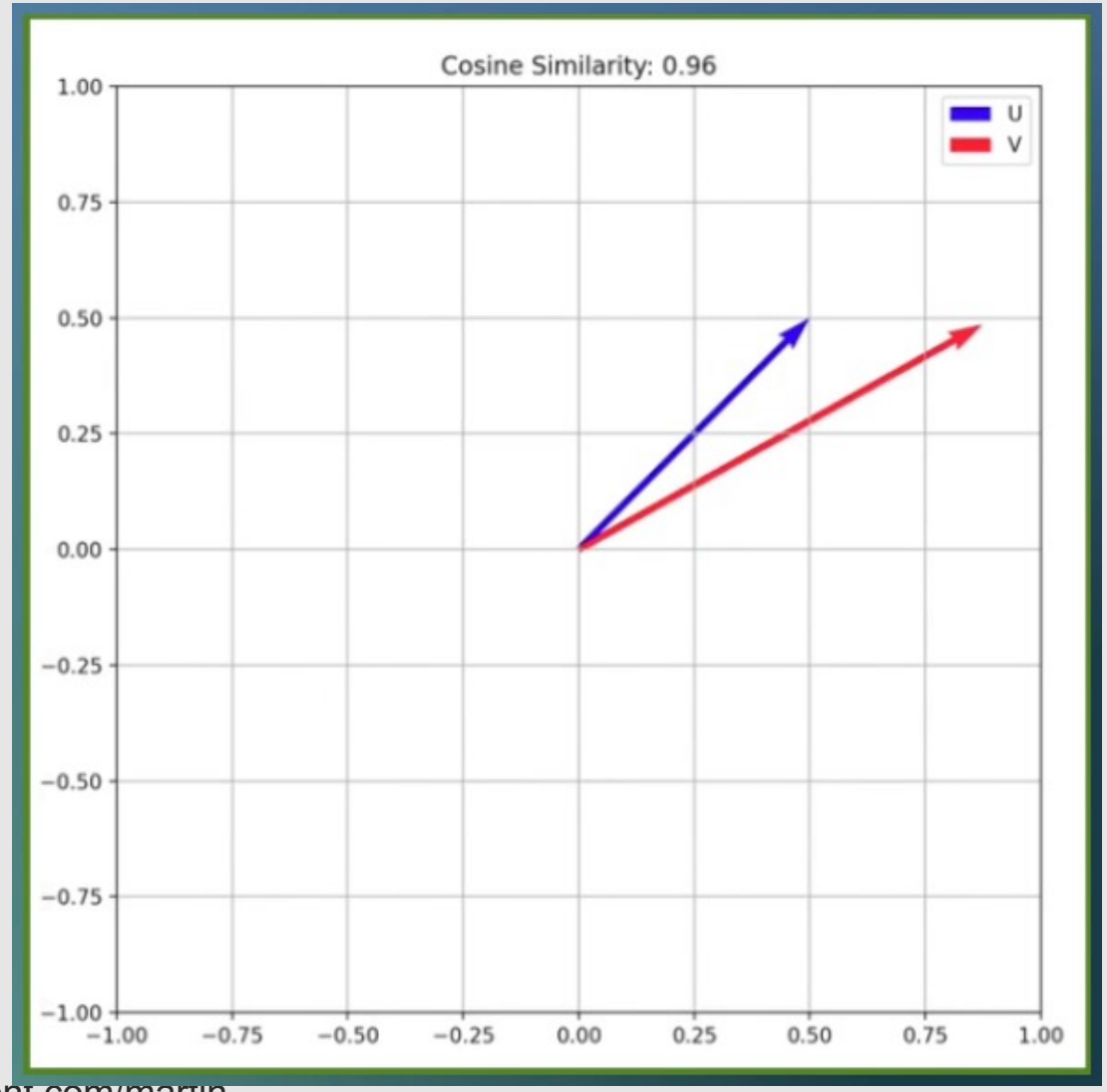
$$=\sqrt{(0.01-0.01)^2+(0.08-0.07)^2+(0.11-0.1)^2}$$

0.9998

Credit: https://towardsdatascience.com/similarity-metrics-in-nlp-acc0777e234c

Visualizing Embeddings and Vector Similarity

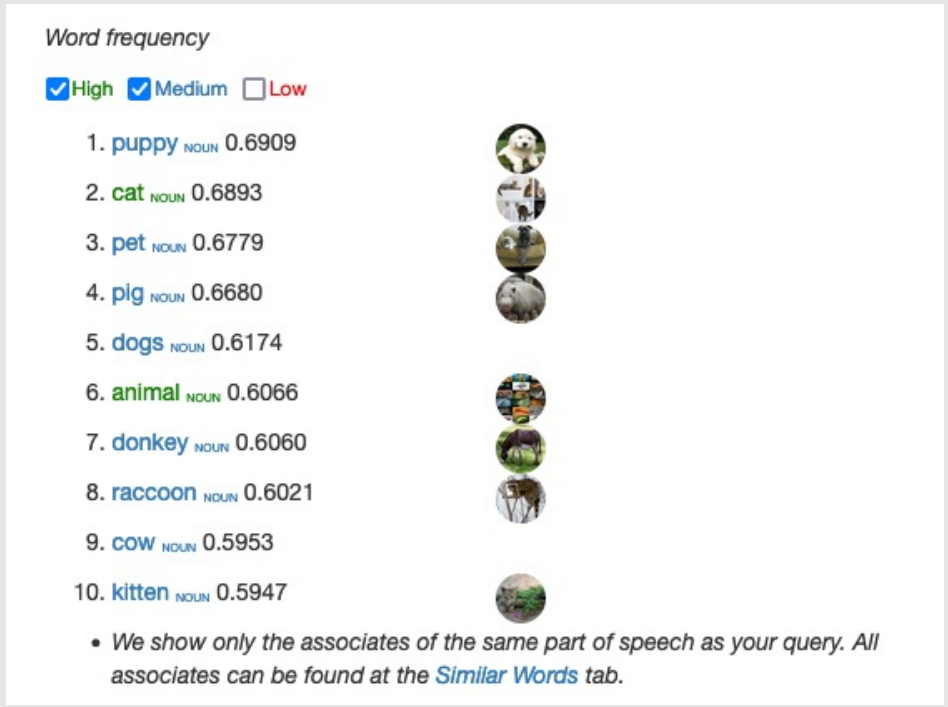


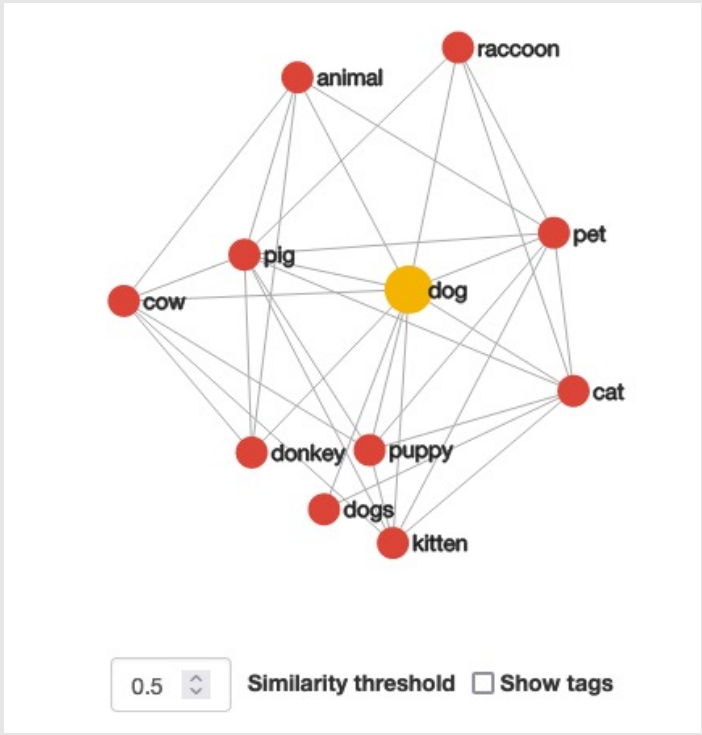


source: https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/martin-labrecque/4483ff5a104f0b56417585c3bc9a12f1/raw/57348e12a70c8d70c2c573d3dbc0122ac077556b/journaux_config.json

Vectors and relationships example

- Query - what words are related to "dog" in model "English Wikipedia"?





Show the raw vector of «dog» in model MOD_enwiki_upos_skipgram_300_2_2021:

[-0.03301828354597092, 0.05134638026356697, 0.0036009703762829304, -0.04066073149442673, 0.10361430048942566, 0.013021323829889297, 0.028161464259028435, -0.0027567853685468435, 0.03388035297393799, -0.044882044196128845, 0.005169689189642668, -0.05818631127476692, 0.0533536821603775, 0.016616210341453552, 0.02030780538916588, -0.008570297621190548, -0.10925538837909698, -0.0708925873041153, 0.04675082117319107, -0.03091960959136486, -0.05172094330191612, 0.04471702128648758, 0.008674593642354012, -0.01816382259130478, 0.05909318849444389, 0.10409023612737656, 0.05633684620261192, -0.024881813675165176, 0.01872968301177025, 0.007228093687444925, -0.023127363994717598, 0.01528552919626236, -0.0643191784620285, -0.010359424166381359, -0.06104437634348869, -0.13868044316768646, -0.023004498332738876, 0.0038427673280239105, -0.021551262587308884, -0.03467748314142227, 0.010687021538615227, -0.017304275184869766, 0.026886526495218277, -0.0030398862436413765, -0.03685504570603371, -0.06017328053712845, 0.047442398965358734, -0.10714898258447647, 0.14808930456638336, -0.06579480320215225, -0.004342162515968084, 0.06226382404565811, 0.08031187951564789, -0.055930640548467636, -0.07030591368675232, 0.015474628657102585, 0.05367768555879593, 0.0917837843298912, 0.031899698078632355, 0.055091146379709244, -0.025078952312469482, -0.048126623034477234, -0.09730836749076843, -0.07128141075372696, 0.019415033981204033, -0.025872433558106422, -0.01761292852461338, 0.015608762390911579, -0.029876720160245895, -0.008602319285273552, 0.049825914204120636, 0.06784739345312119, 0.005586292129009962, -0.07148509472608566, -0.03097137063741684, -0.020296750590205193, 0.05099814385175705, 0.14920306205749512,

Source: http://vectors.nlpl.eu/explore/embeddings/en/MOD_enwiki_upos_skipgram_300_2_2021/dog_NOUN/

Vector Databases

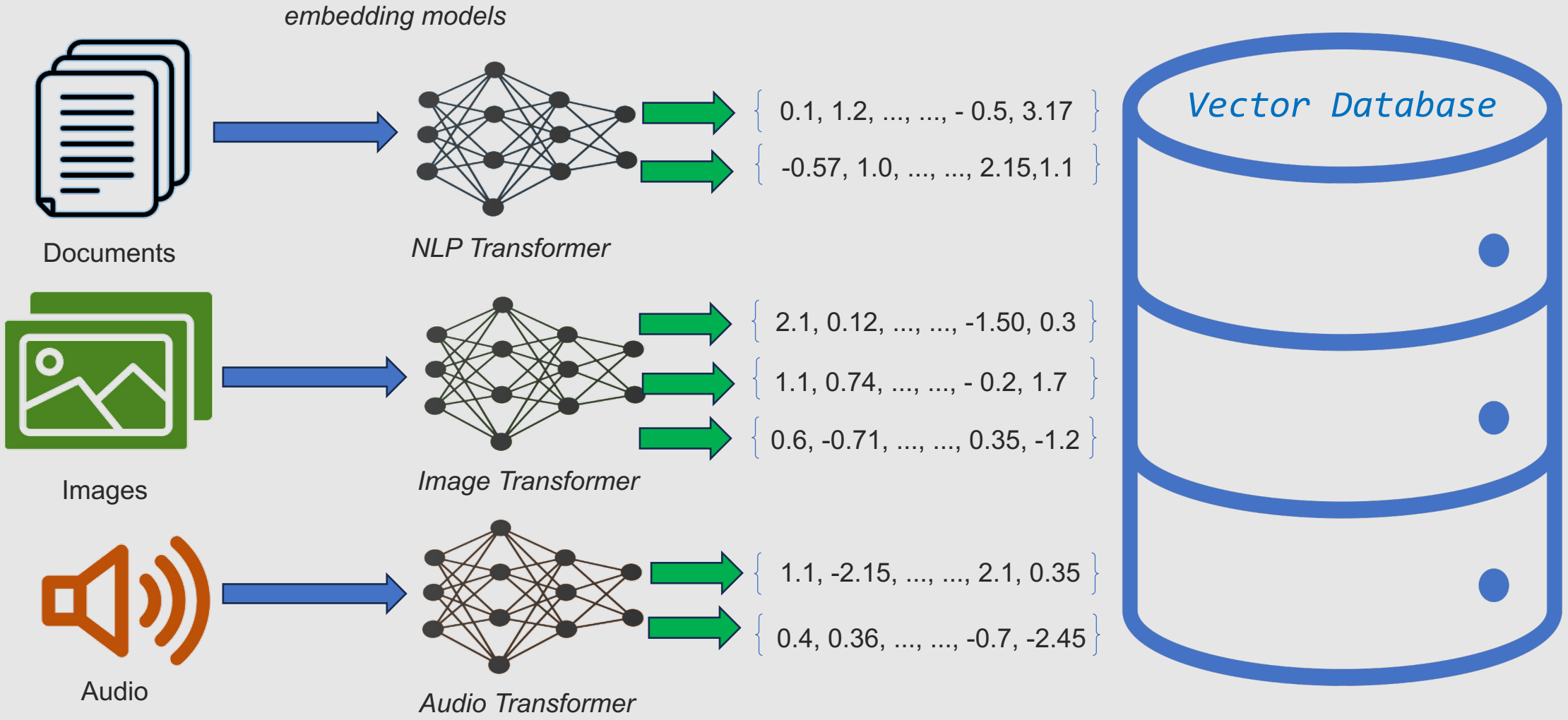
Vector Databases

- Specialized database that index and stores *vector embeddings*
- Useful for
  - fast retrieval
  - similarity search
- Offer comprehensive data management capabilities
  - metadata storage
  - filtering
  - dynamic querying based on associate metadata
- Scalable and can handle large volumes of vector data
- Support real-time updates
- Play key role in AI and ML applications

Vector Database

How data gets into Vector Databases

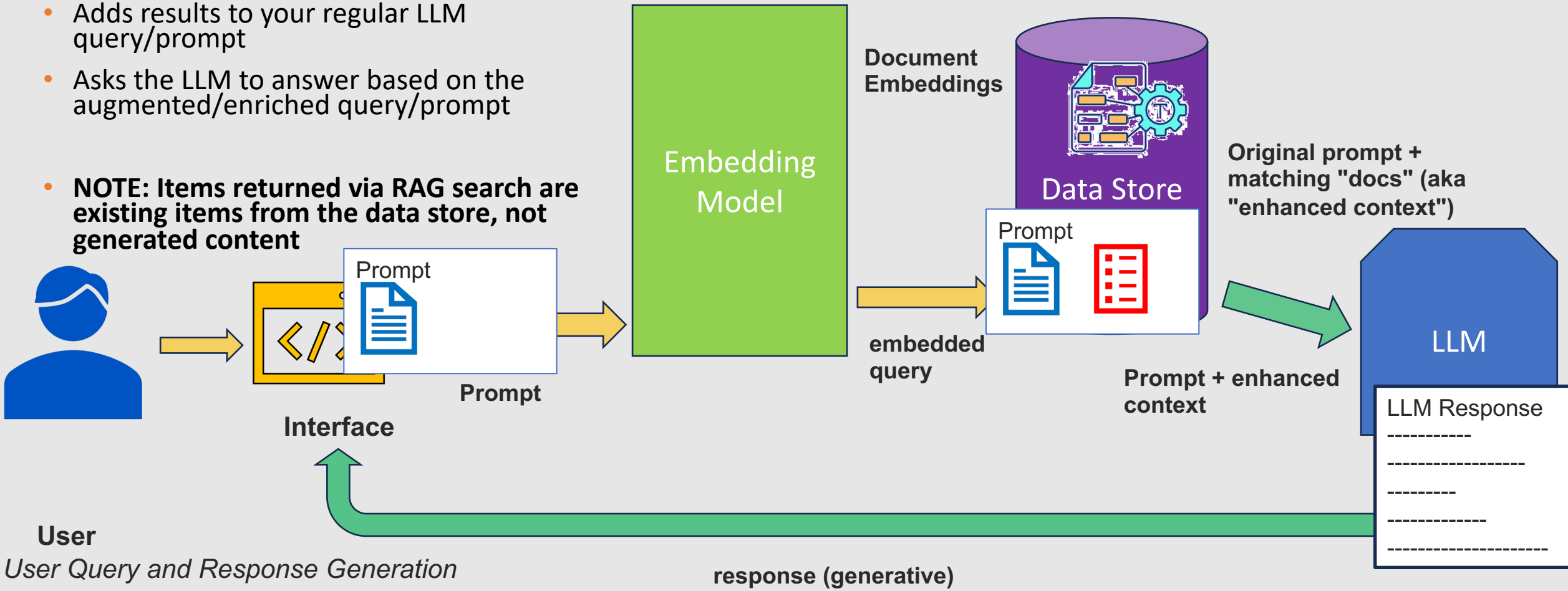
- Data is input, converted to embeddings (vectors) and stored
- Queries are input, converted to embeddings (vectors) and then **similarity metrics** are used to find results ("nearest neighbors")



How does RAG work?

- For queries/prompts, application gathers results (most relevant ones) from the vector database with your data
- Adds results to your regular LLM query/prompt
- Asks the LLM to answer based on the augmented/enriched query/prompt

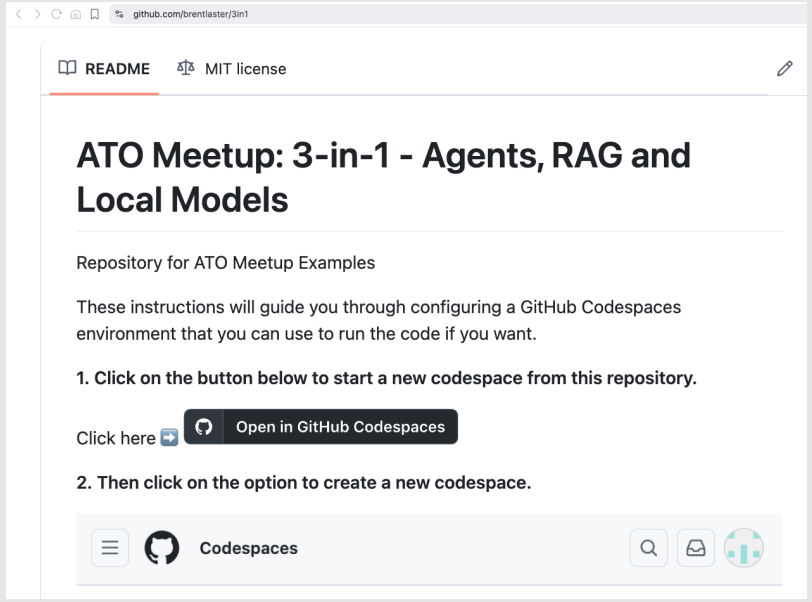
- **NOTE: Items returned via RAG search are existing items from the data store, not generated content**

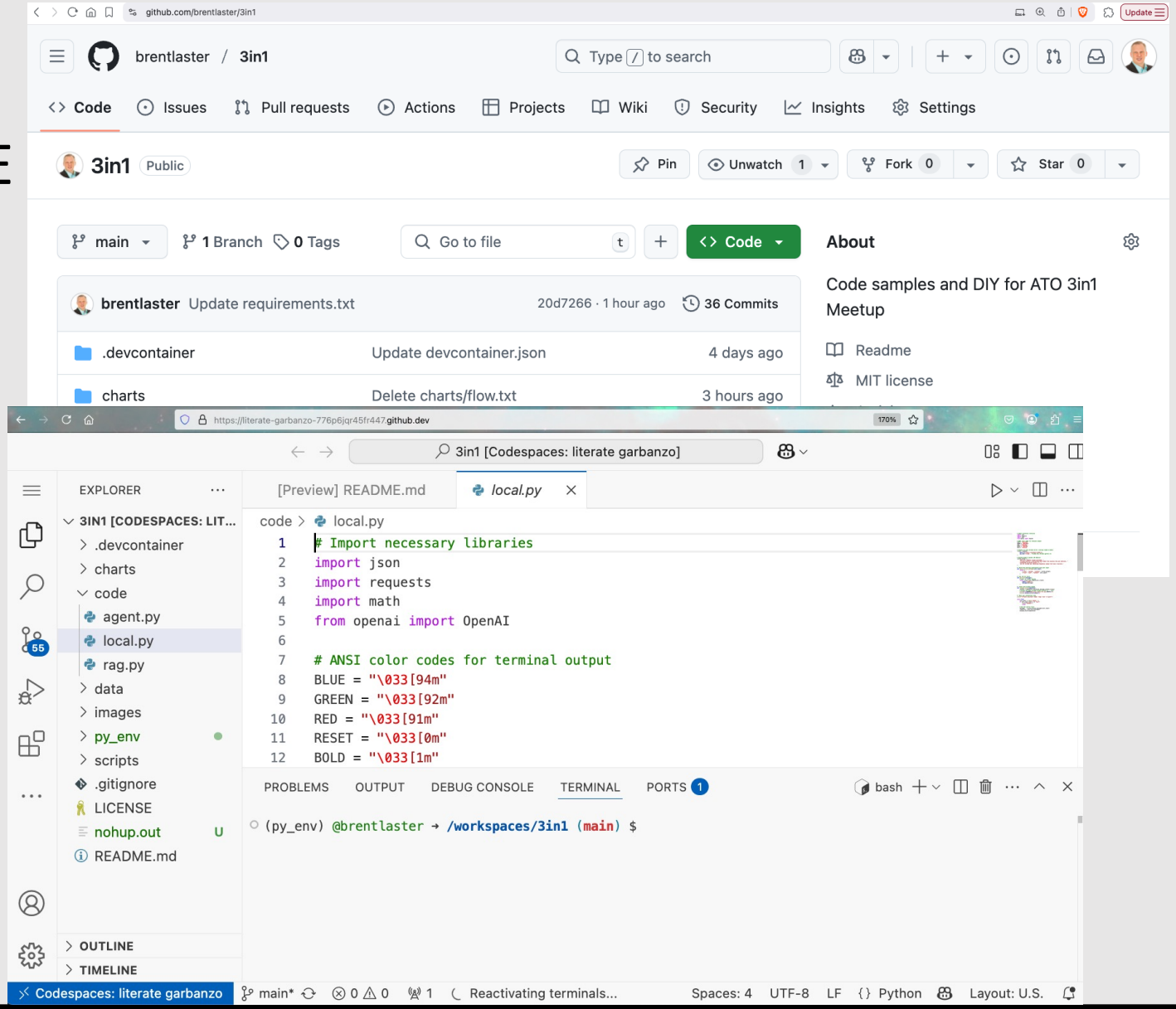


Demo #3 – Adding RAG to our code

DIY – github.com/brentlaster/3in1

- Fork if desired
- Click on button in README to start codespace
- Follow guide.md

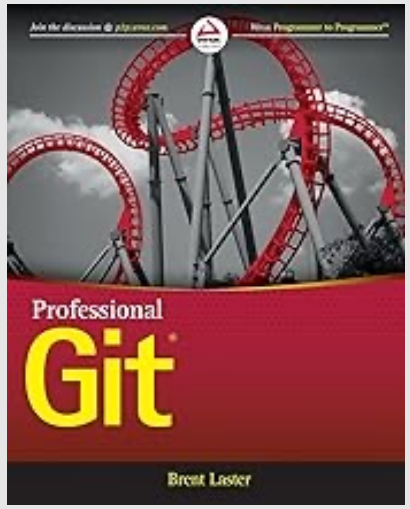


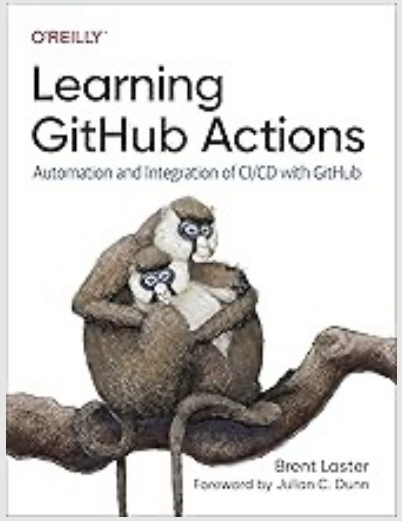


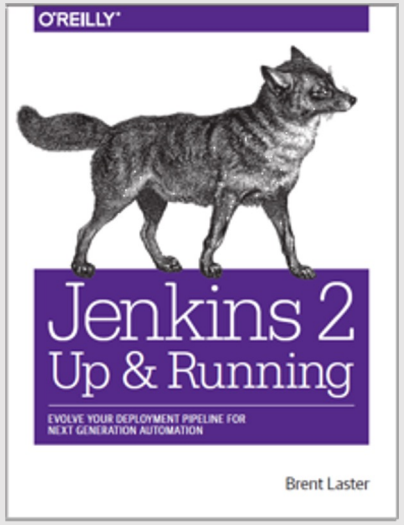
That's all - thanks!

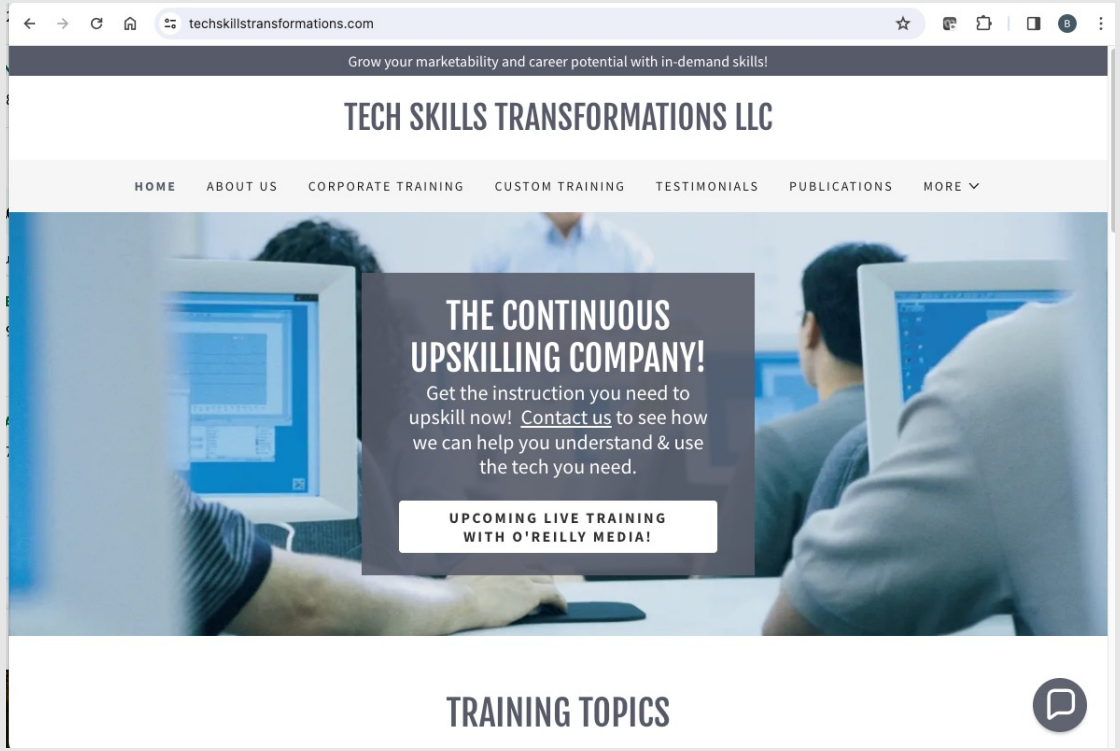
Contact: training@getskillsnow.com

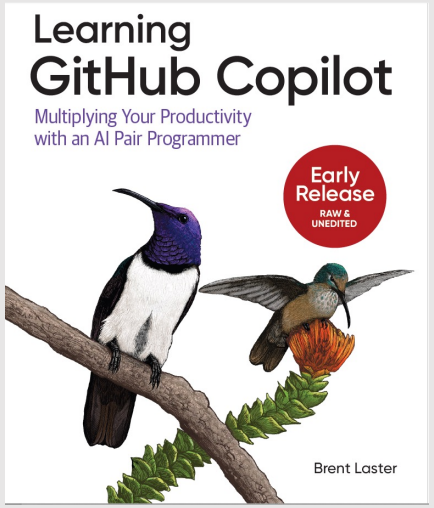
techskillstransformations.com
getskillsnow.com

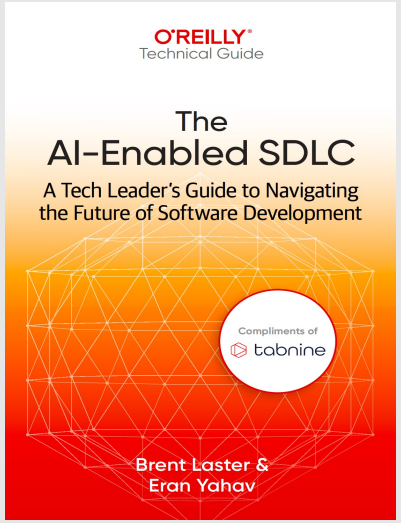












- ❑LinkedIn: brentlaster
- ❑X: @BrentCLaster
- ❑Bluesky: brentclaster.bsky.social
- ❑GitHub: brentlaster