# Part II: Building the AI Assistant

## Introduction

* [Attention Is All You Need](https://arxiv.org/pdf/1706.03762) Vaswani et al. (2017, Google Brain/ Research)
* 5 days to 1 million users (OpenAI)
* 1.8 billion monthly visits in March 2023 (OpenAI)

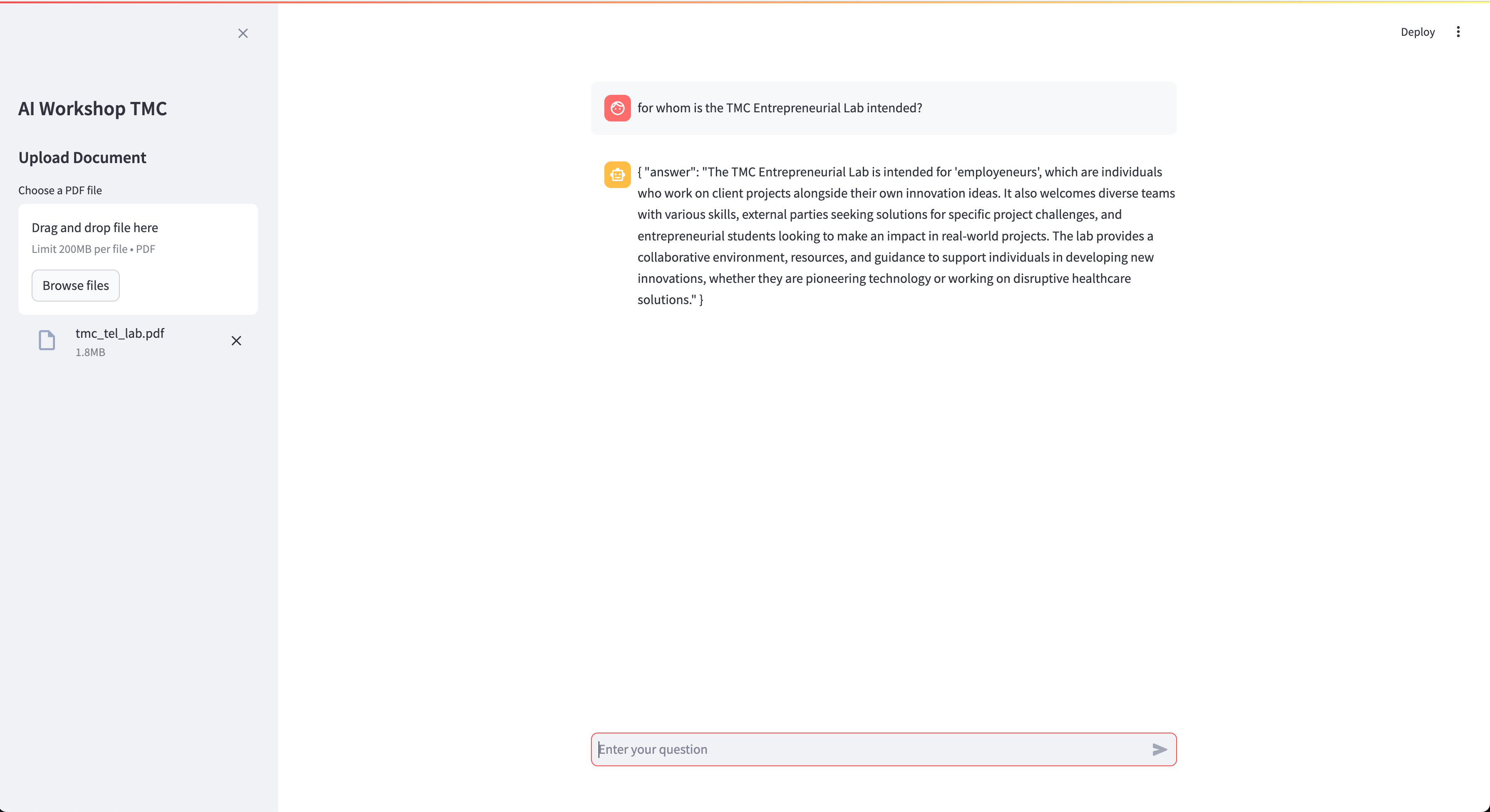
* **Agricultural Revolution**: Around 10,000 BCE, shift to settled farming.
* **Industrial Revolution**: Late 18th century, rise of industrialization.
* **Digital (Computer) Revolution**: Mid-20th century, advent of computers.
* **AI Revolution**: Early 21st century, integration of artificial intelligence.

### Agenda

* Introduction to LLMs
* Starting Docker Containers
* Build AI Assistant (Walkthrough & HandsOn)
  + Ingestion (Load, Split, Embed, Store)
  + Similarity Search
  + Combine Context
  + Response Generation
* Langserve and Streamlit App

* We will build a simple retrieval augmented generation (RAG) pipeline and complete HandsOn tasks.
* The notebook is based on [langchains rag intro](https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_1_to_4.ipynb).
* We build towards a broader understanding of the RAG langscape langchain's [rag from scratch](https://github.com/langchain-ai/rag-from-scratch/tree/main).

### AI Workshop TMC - Application Screenshot



#### Features

**Document Upload**

* Users can upload a PDF document through the sidebar.
* The uploaded document is sent to the backend service for processing.
* The app displays the result of the document upload in the sidebar.

**Chat Interface**

* The main interface includes a chat window where users can interact with the AI assistant.
* Users can enter their questions in the chat input field.
* The chat history is preserved across interactions, displaying both user questions and assistant responses.

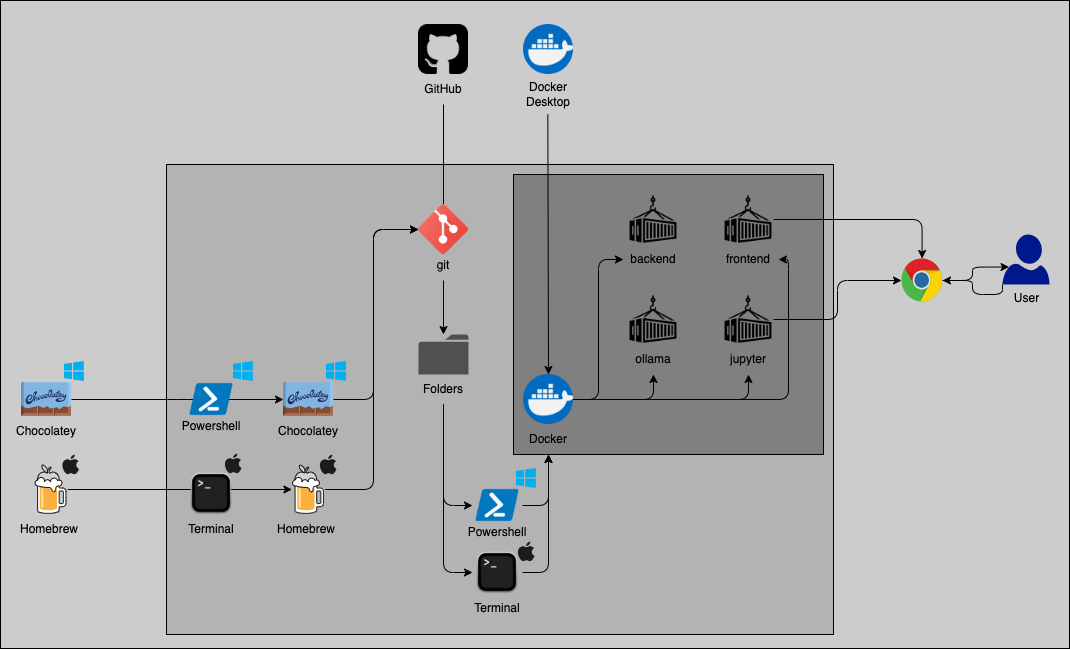
**Backend Queries**

* The app can query the backend service in two ways:
  + **Query with Document Context**: If a document has been uploaded, the app queries the backend with the context of the uploaded document.
  + **Direct Query**: If no document is uploaded, the app sends a direct query to the backend.
* The assistant's responses are displayed in the chat window.

## Learning Objectives - Part 2: Building the AI Assistant

1. Complete Installation of Essential Tools
2. Understand Functioning of Essential Tools
3. **Understand the Basics of Large Language Models (LLMs)**
4. **Understand on a Programmatic Level how AI Assistants are Built**

## Installation Overview



## Starting The Application

### Clone GitHub Repo

# Clone the repository from the given URL  
git clone https://github.com/marioisbeck/ai\_workshop\_tmc.git  
  
# Change directory to the cloned repository  
cd ai\_workshop\_tmc

### Run Docker-Comose to Startup Containers

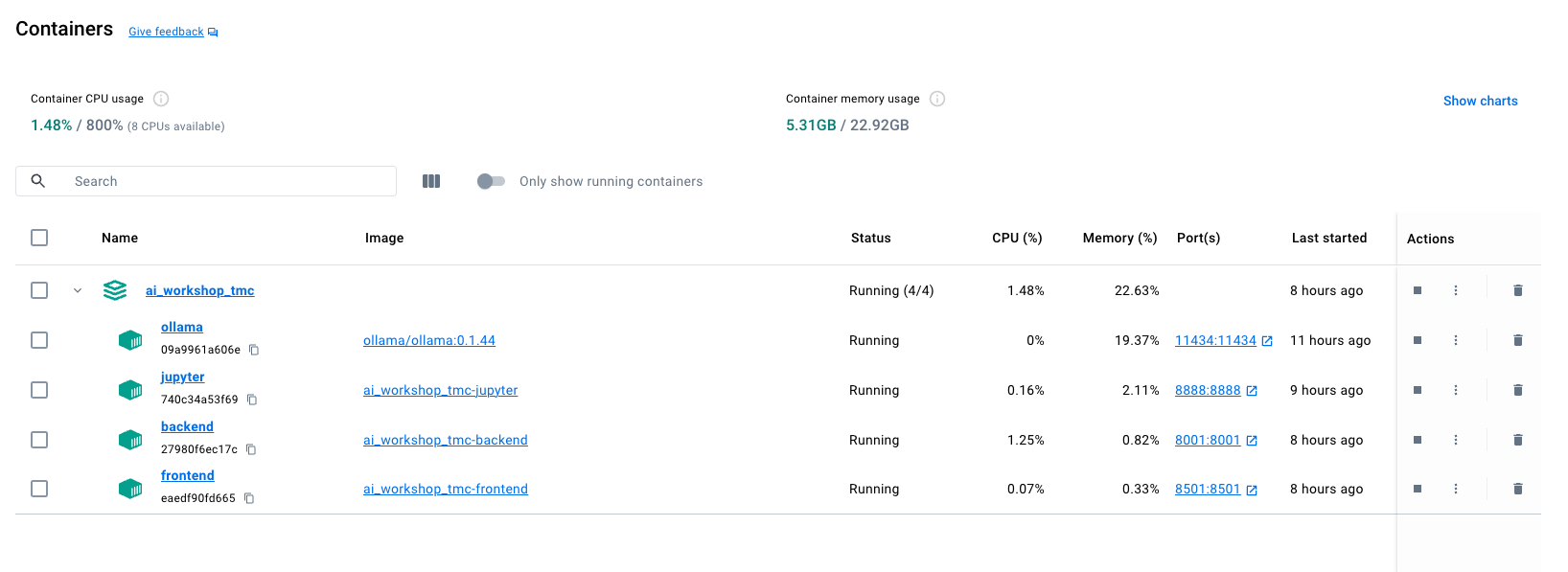
# Copy the example environment file to a new .env file  
cp .env.example .env  
  
# Build and start the Docker containers in detached mode  
docker-compose up -d --build

### Download LLMs

# Pull the mxbai-embed-large model in the ollama container  
docker exec ollama ollama pull mxbai-embed-large && \  
# Pull the all-minilm model in the ollama container  
docker exec ollama ollama pull all-minilm && \  
# Pull the wizardlm2:7b model in the ollama container  
docker exec ollama ollama pull wizardlm2:7b && \  
# Run the wizardlm2:7b model in the ollama container  
docker exec ollama ollama run wizardlm2:7b

### Test Application

Open Docker Desktop and go to Containers. You should see something like this (all container icons should be green):



* **Application** Go to <http://localhost:8501> and see if the frontend is there, upload a pdf (<5MB) and ask a question about it. If everything goes well, it should answer 😊.
* **Jupyter** Go to <http://localhost:8888> where you should see the option to run the ai\_workshop\_tmx.ipynb notebook. Within this notebook you can follow along.

### Remove Containers

# Stop and remove all containers defined in the docker-compose.yml file  
docker-compose down

## Client Journey

* programming interest, proactive approach
* presented working chatbot prototype to client
* client invested in dedicated GPU server

## TMChampionship

* TEL organises/-ed a project journey towards a shark tank like investor pitch in November
* Milan and I - started TMChampionship project Prometheon.ai to build a sustainable manufacturing knowledge expert
* TEL has many wonderful opportunities for you to try new things, learn, connect and especially grow

## Prometheon.ai

[www.prometheon.ai](https://www.prometheon.ai)

**Theon**

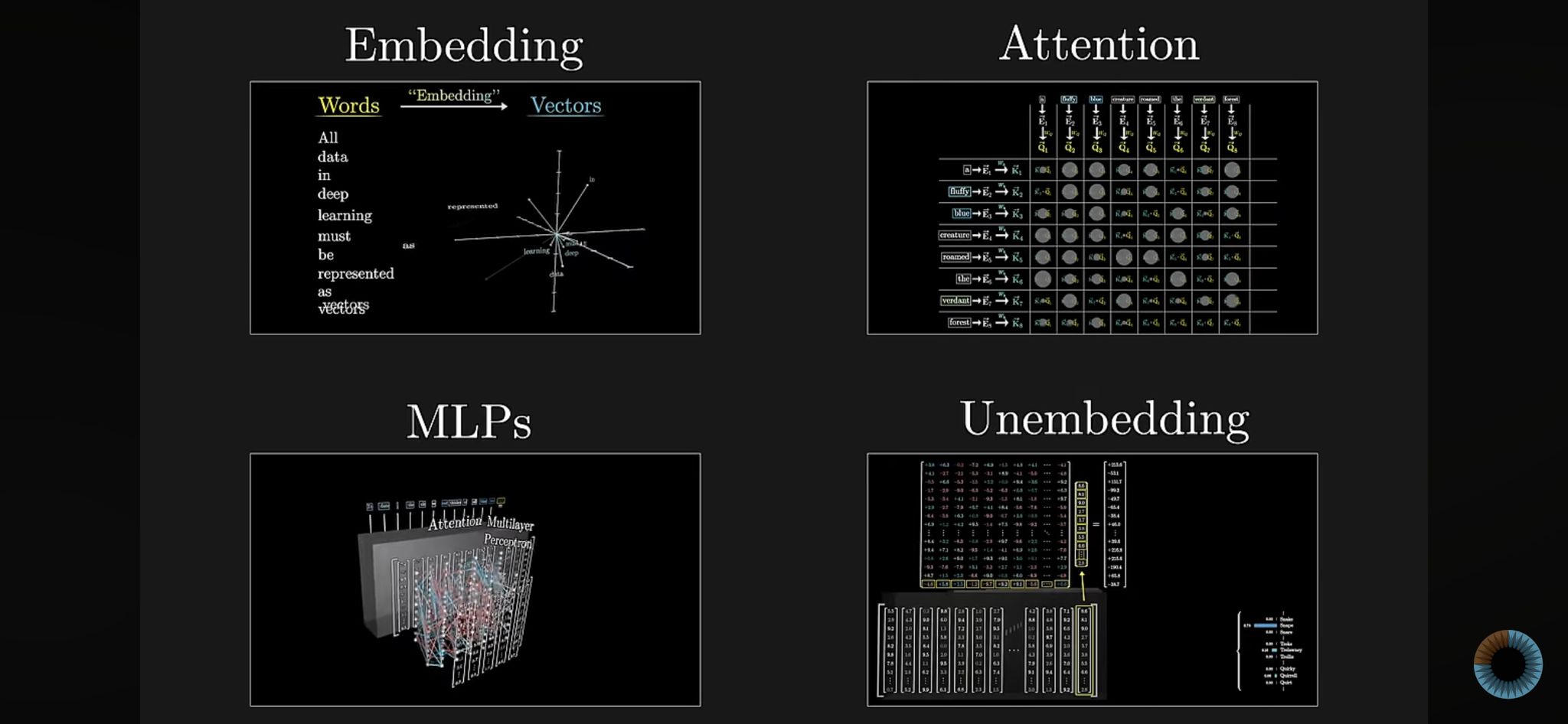
Theon helps manufacturing companies manage their data securely on-site and integrates with existing systems to improve decision-making and efficiency.

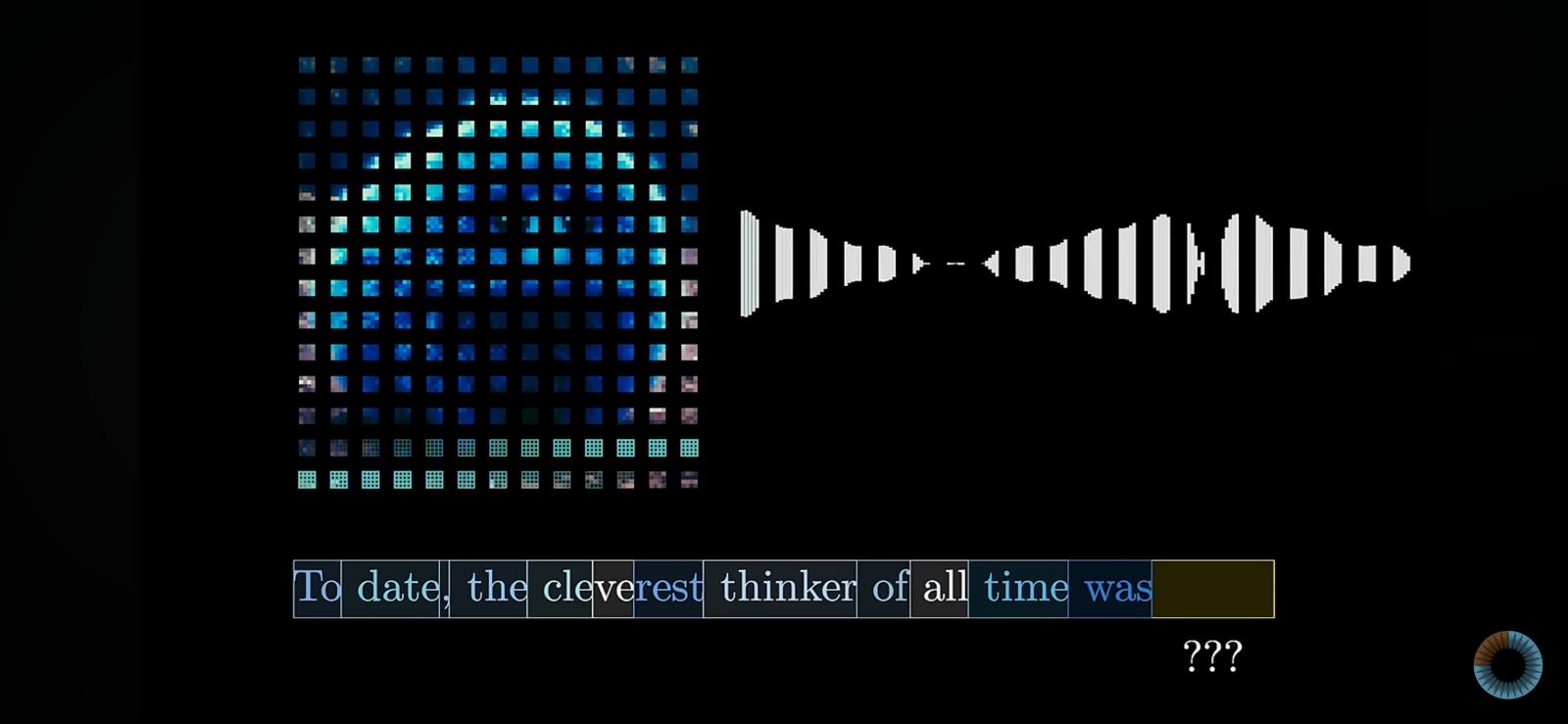
**Rachel**

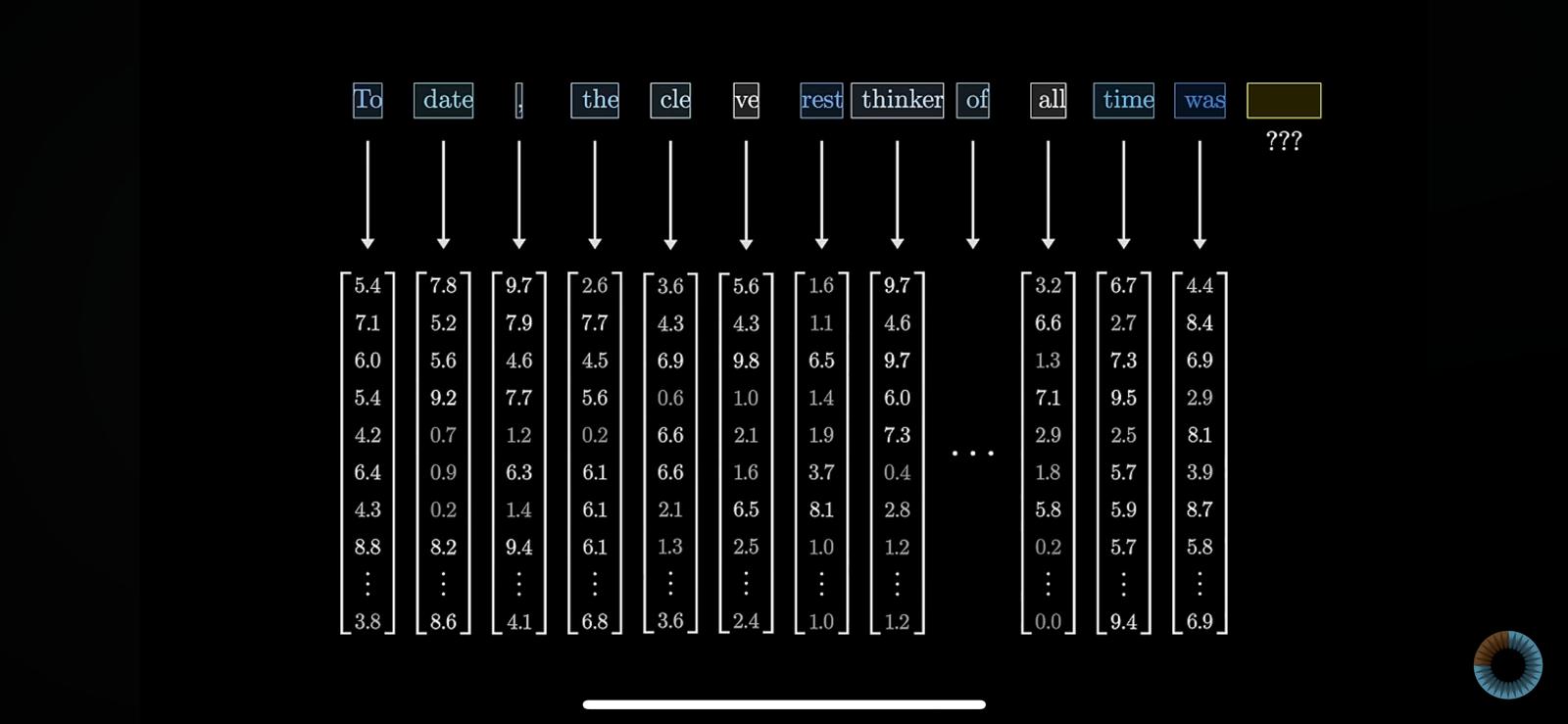
Rachel offers expert guidance on sustainable manufacturing through an AI chat interface, integrating supplier details, company data, and resources to promote eco-friendly practices while ensuring efficiency and profitability.

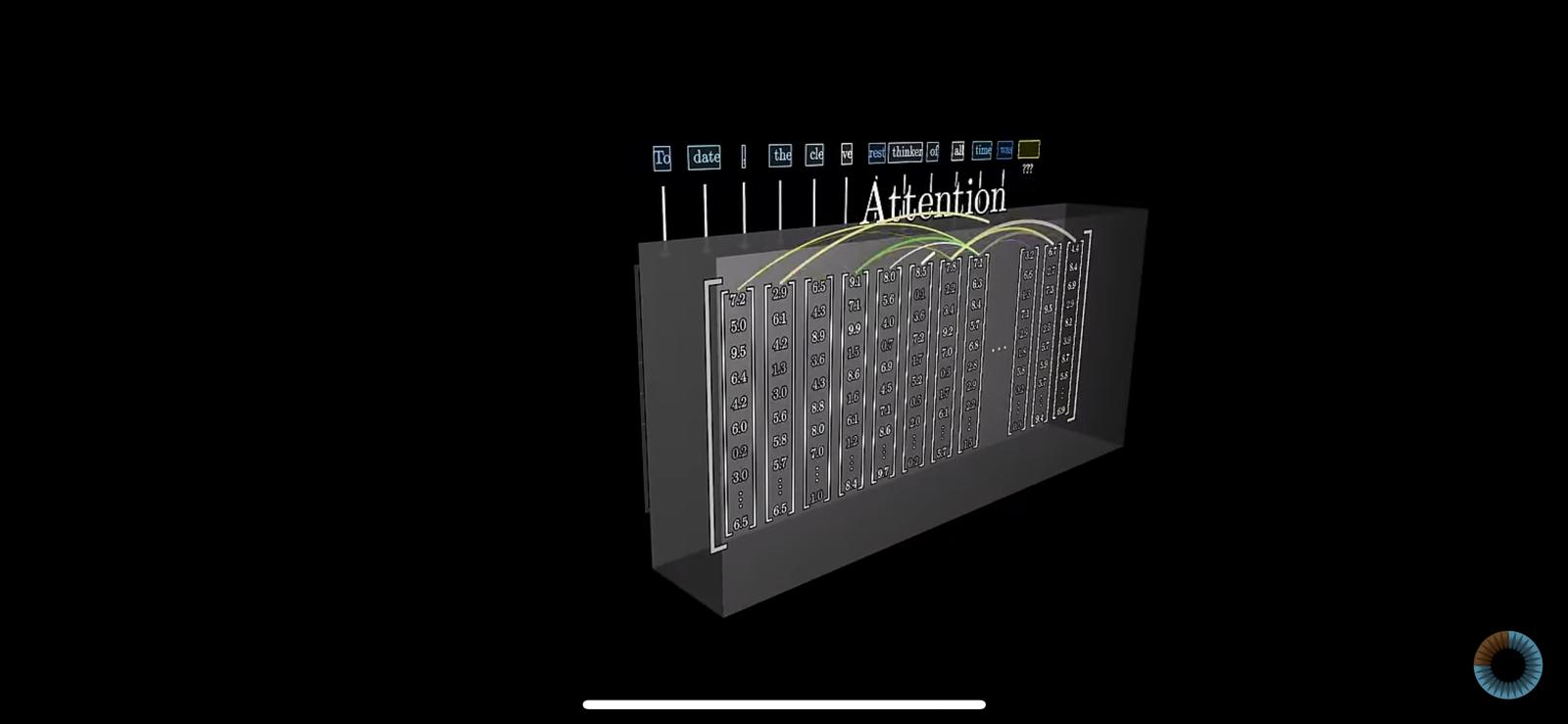
## Introduction to LLMs

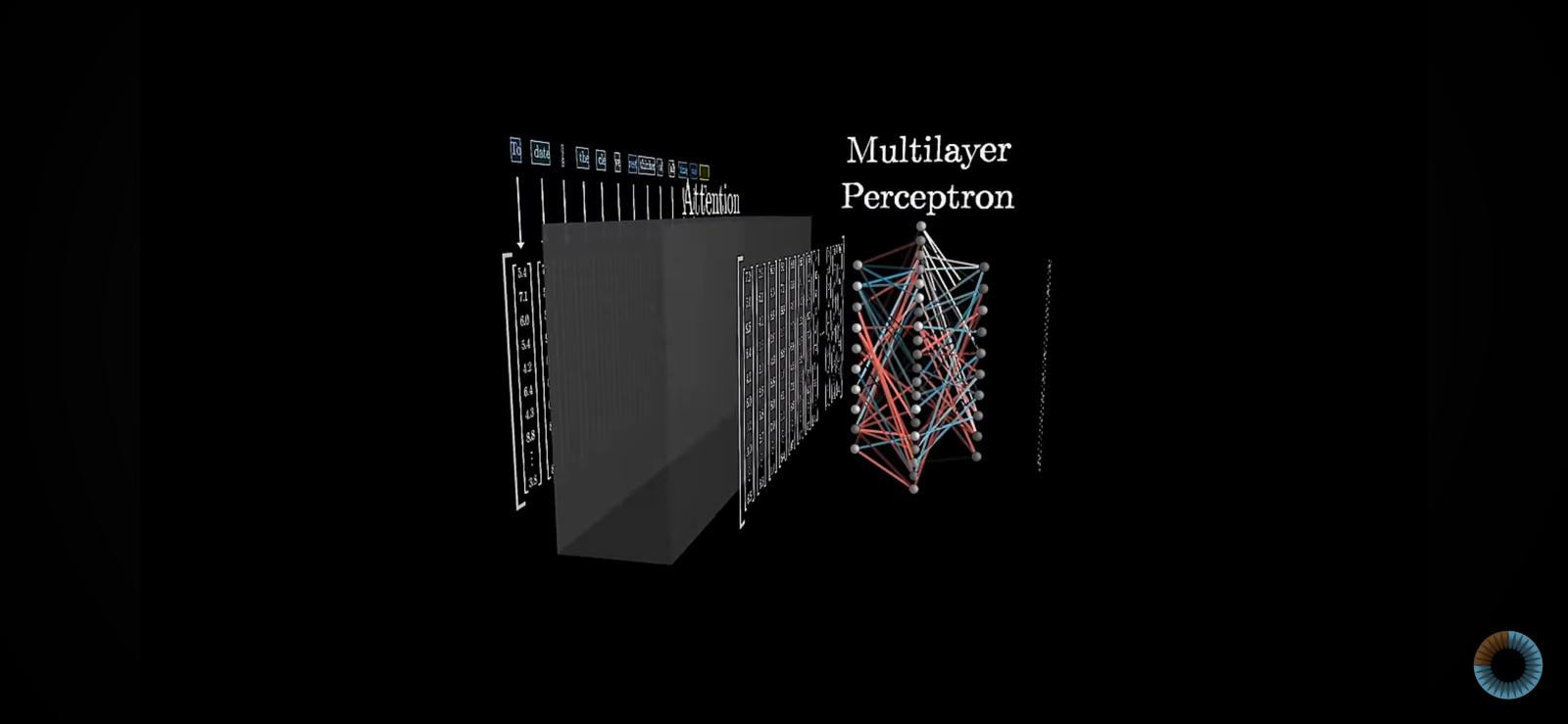
Slides are shamelessly taken from 3Blue1Brown's [But what is a GPT? Visual intro to transformers | Chapter 5, Deep Learning](https://www.youtube.com/watch?v=wjZofJX0v4M&ab_channel=3Blue1Brown)

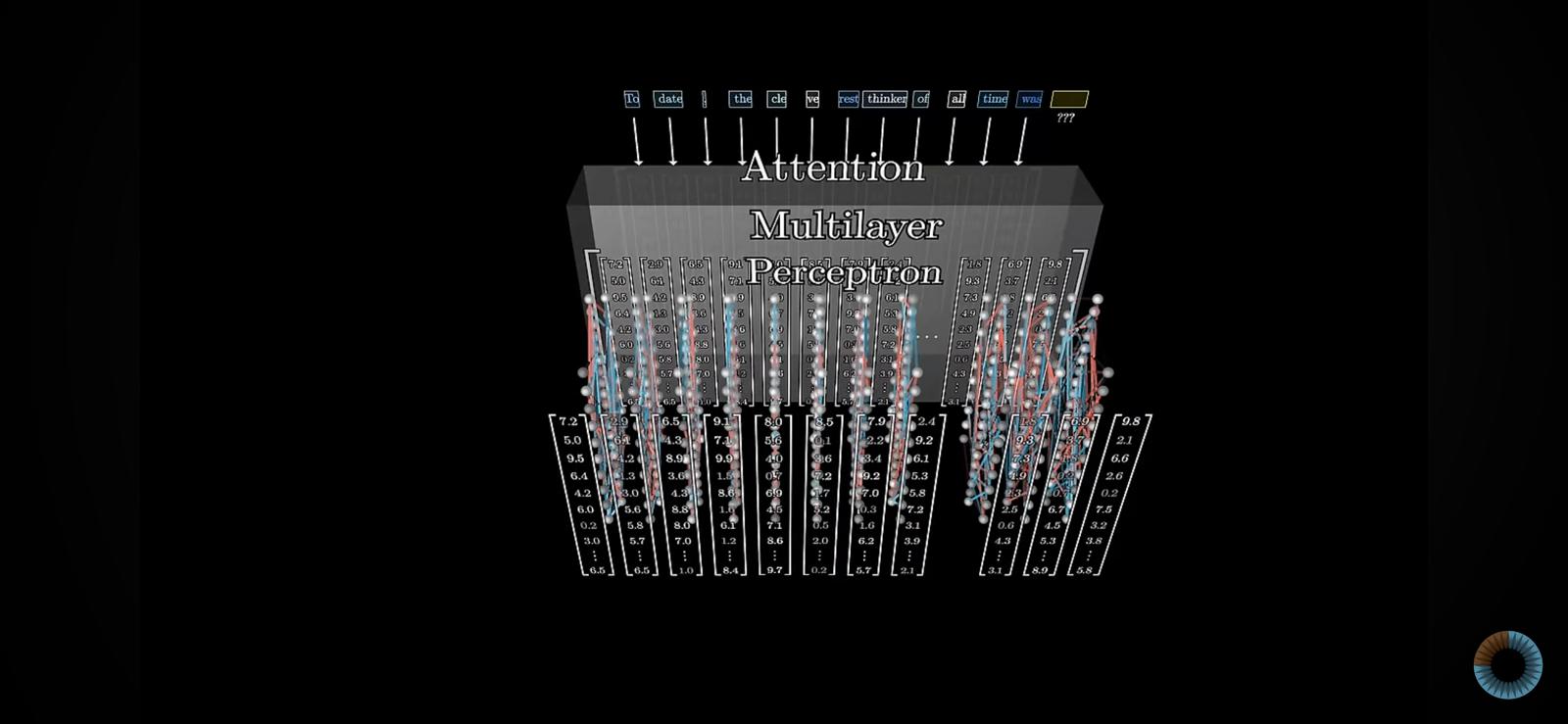


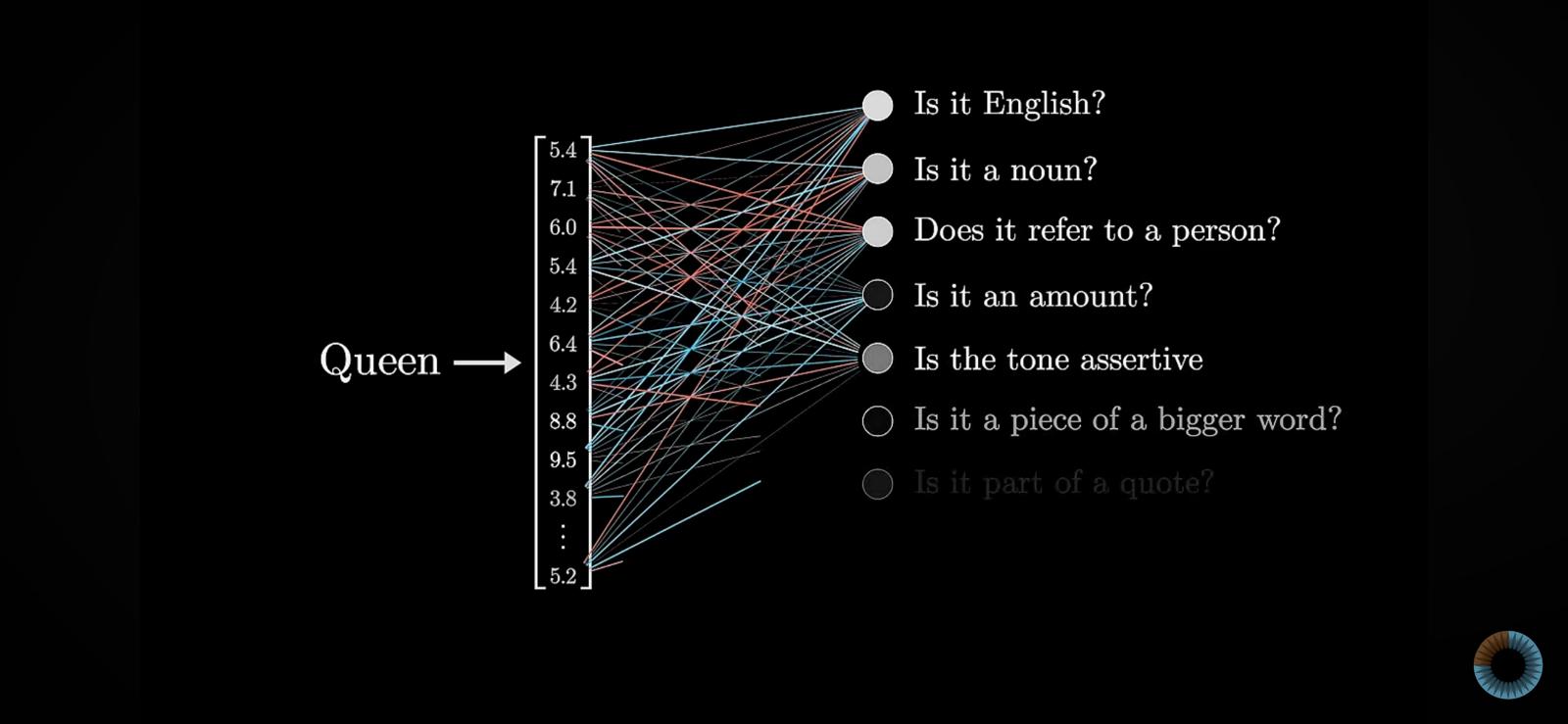


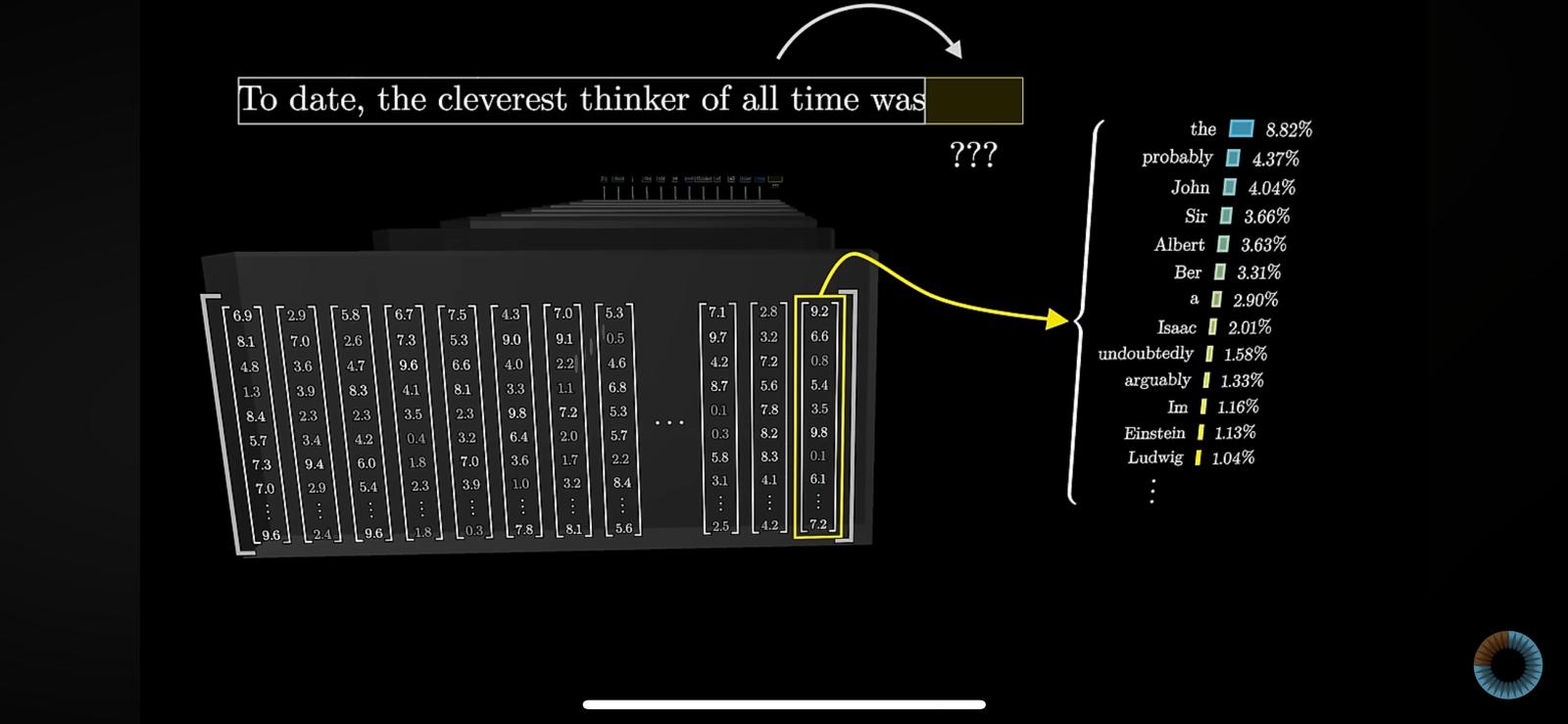


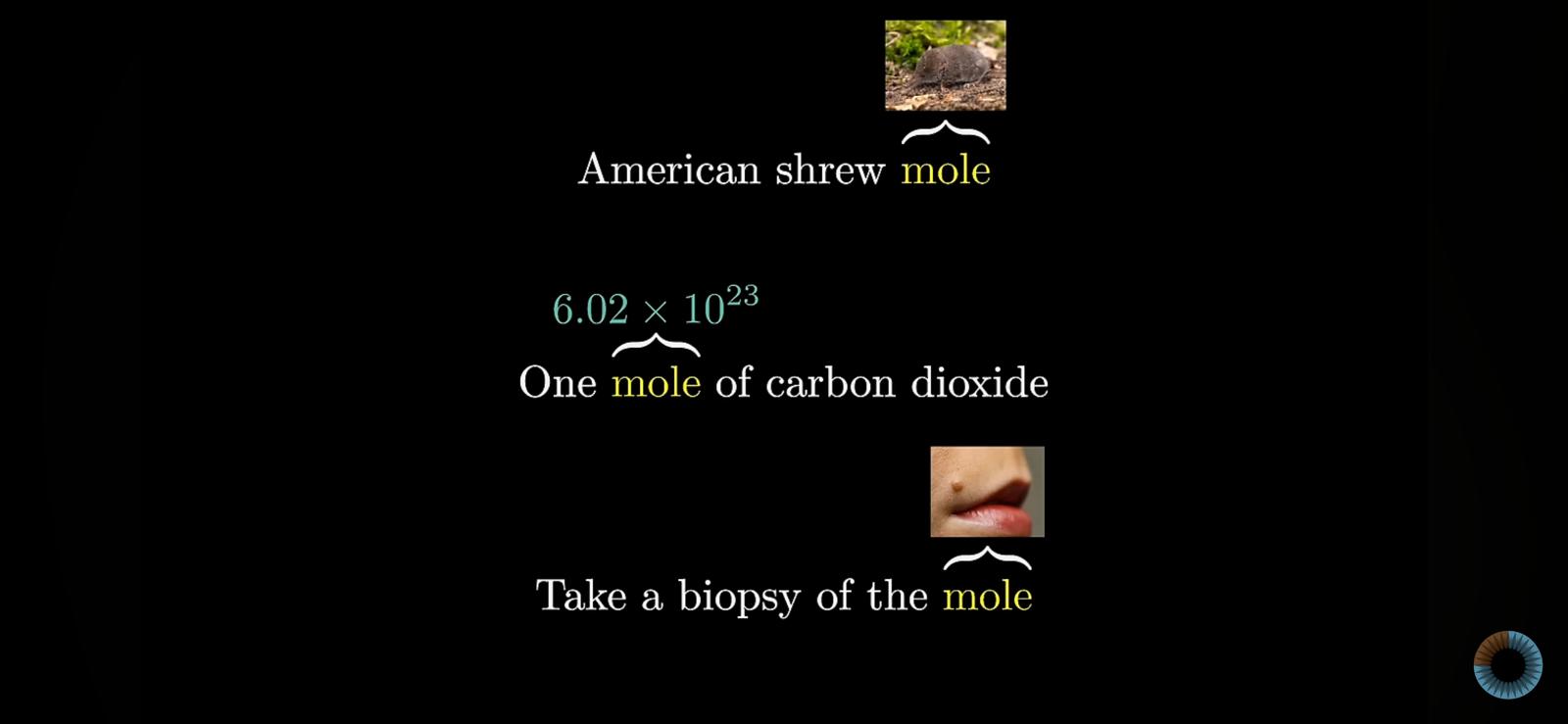


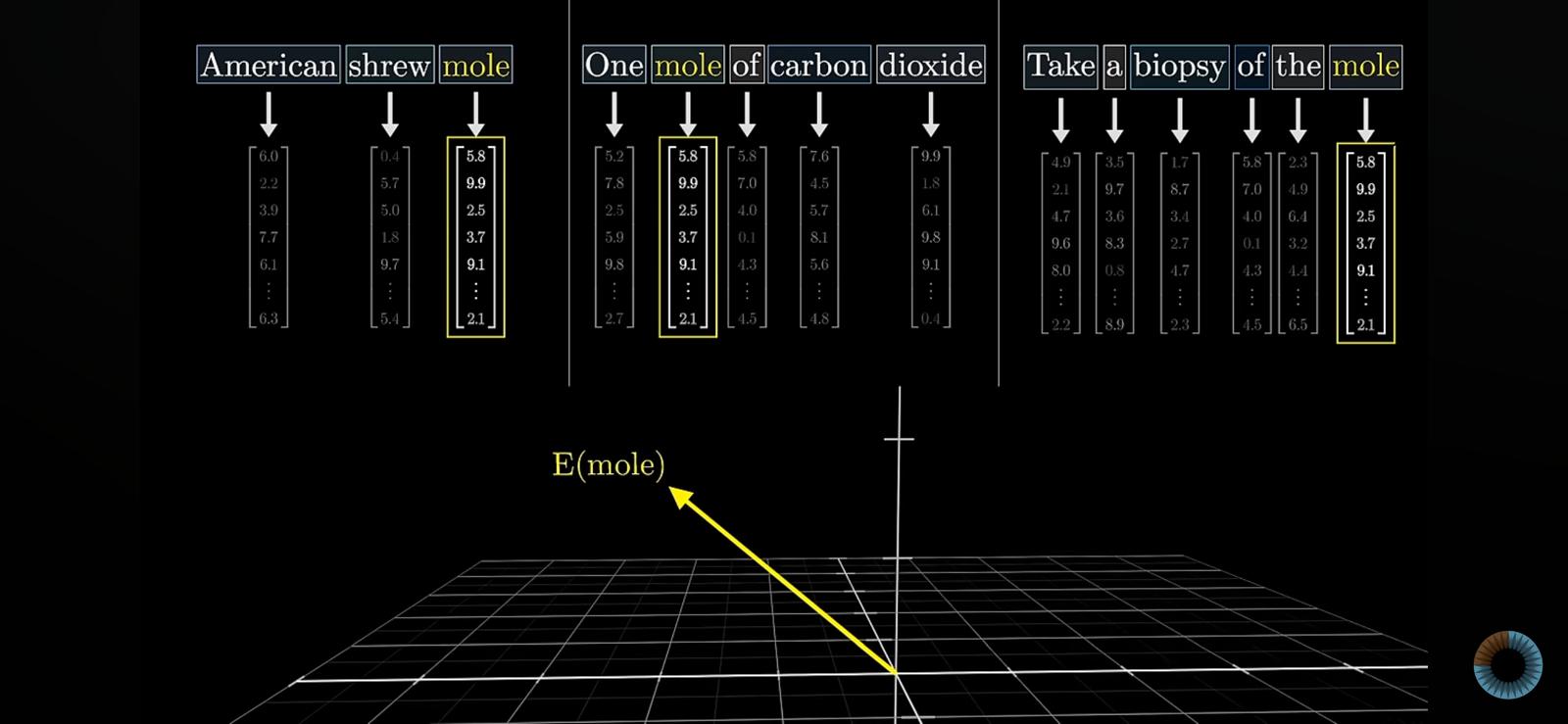


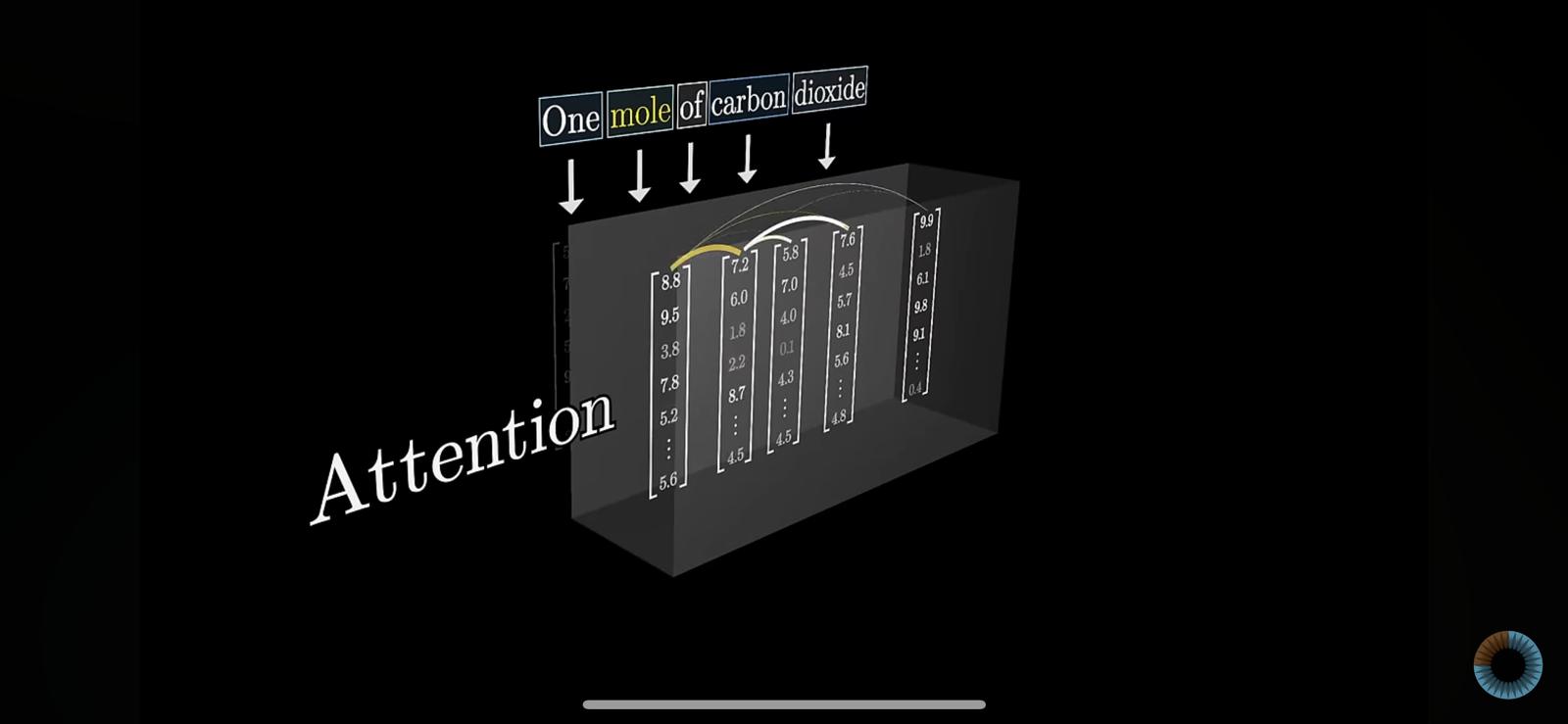


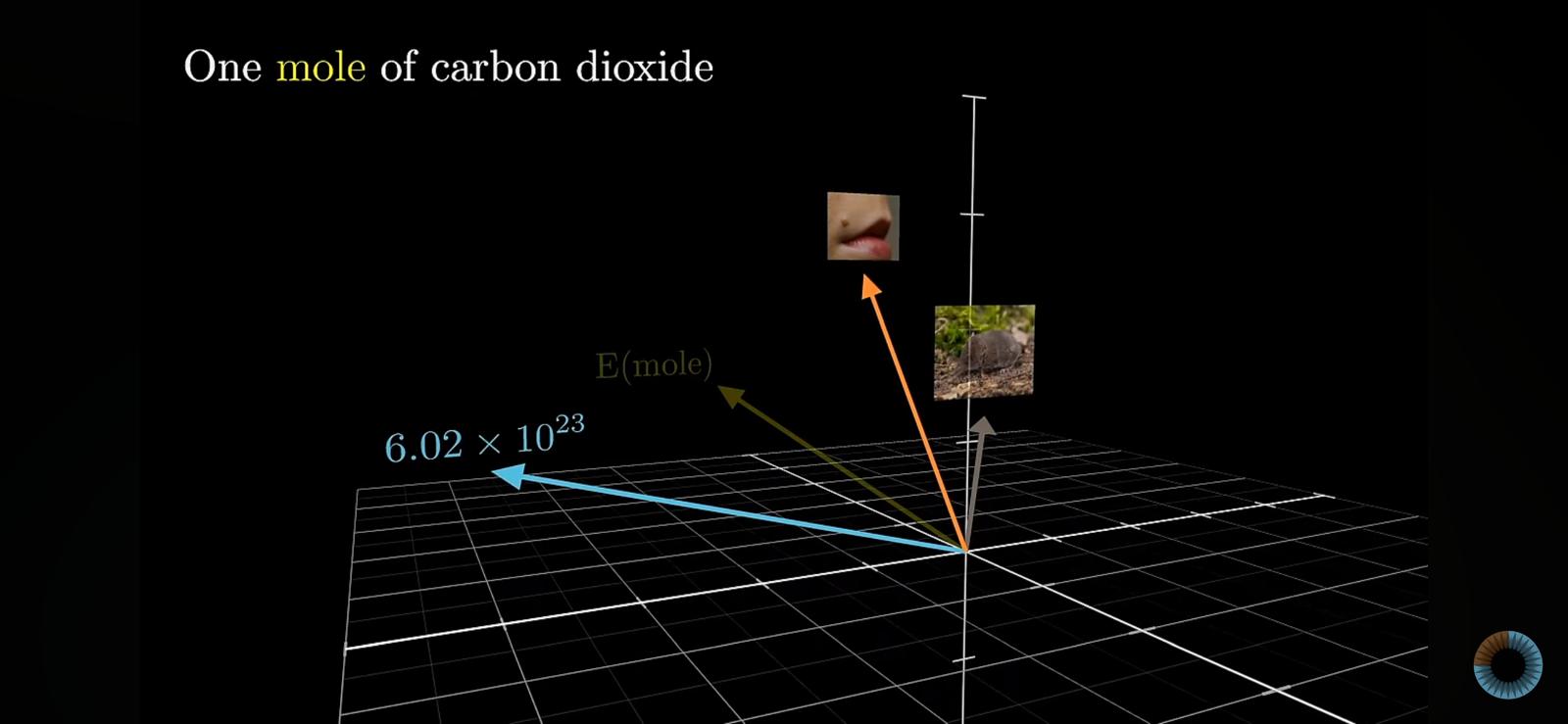


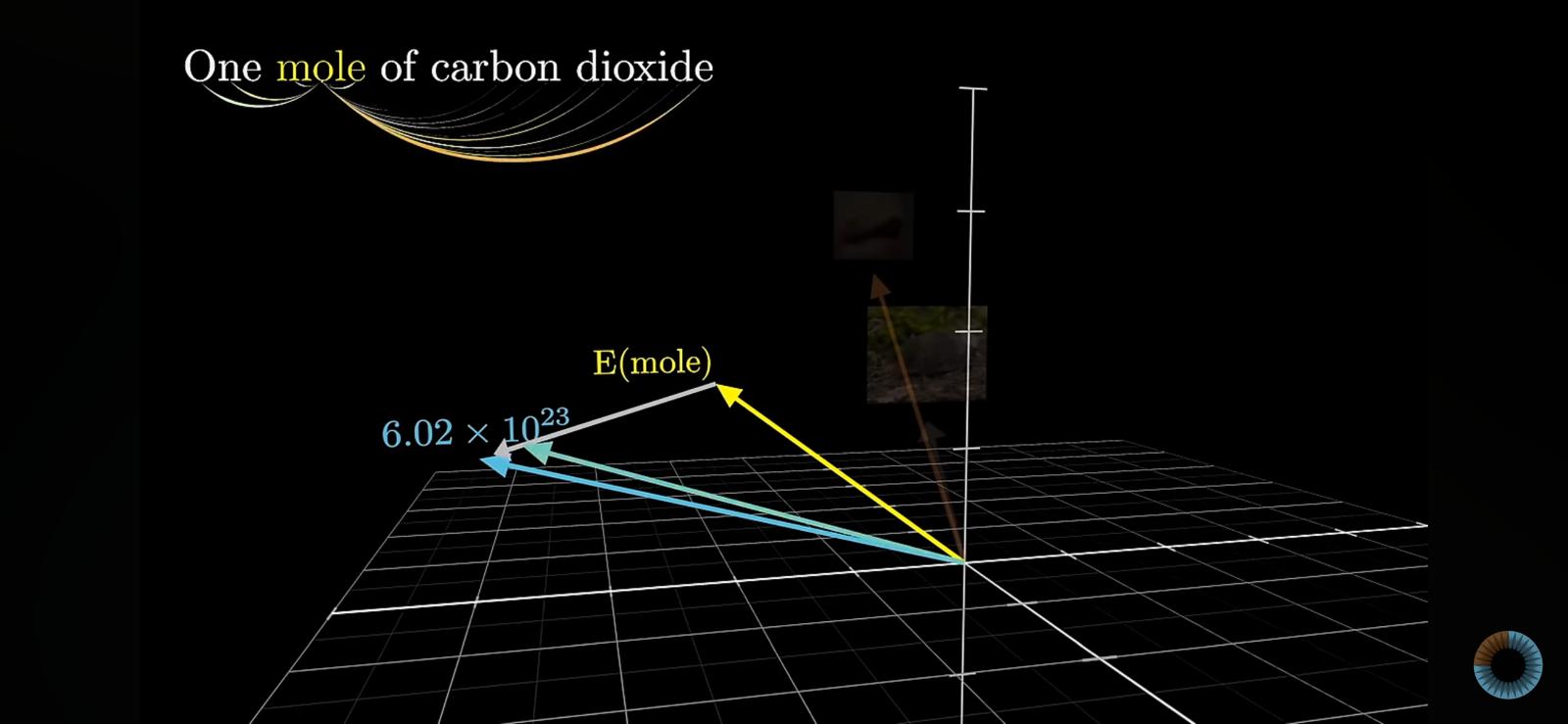




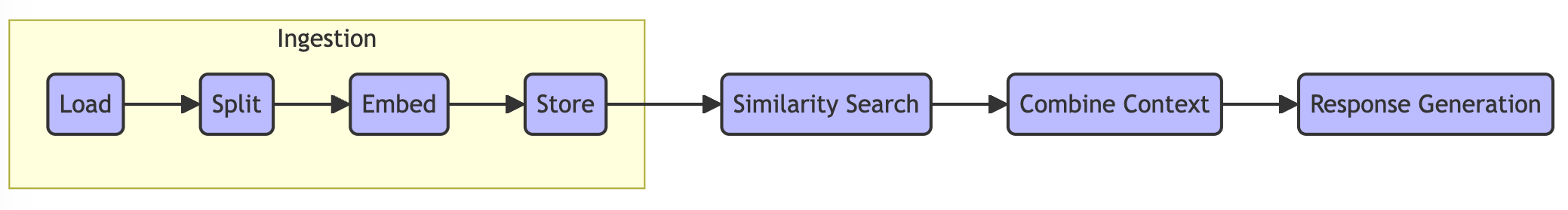








## Build AI Assistant



* **Ingestion**: Load and preprocess documents for further processing.
  + **Load**: Upload documents to the backend.
  + **Split**: Split documents into manageable chunks using characters, sections, semantic meaning, and delimiters.
  + **Embed**: Convert document chunks (and query) into vector embeddings for representation.
  + **Store**: Store the embeddings in a vector database (Vectorstore) for efficient retrieval.
* **Similarity Search**: Use the query embedding to search and retrieve the most relevant document chunks from the Vectorstore.
* **Combine Context**: Combine retrieved document chunks with the query to provide context for the generation model.
* **Response Generation**: Use a language model to generate a response based on the query and retrieved context.

### Setup

#### Keep it Clean

The following is only to suppress output which we do not care about in this workshop.

import warnings  
import logging  
import os  
import numpy as np  
  
# Setting USER\_AGENT variable for jupyter notebook  
os.environ['USER\_AGENT'] = 'jovyan'  
  
# Disable warnings  
warnings.filterwarnings('ignore')  
  
# Disable info messages  
logging.getLogger().setLevel(logging.WARNING)

### Load Libraries

import bs4 # Library for web scraping and parsing HTML/XML  
from langchain import hub # Access langchain hub for pre-built tools and models  
from langchain.text\_splitter import RecursiveCharacterTextSplitter # Tool to split text recursively by characters  
from langchain\_core.output\_parsers import StrOutputParser # Parses output into strings  
from langchain\_core.runnables import RunnablePassthrough # Pass-through runnable for data processing  
from langchain\_community.chat\_models import ChatOllama # Chat model from Langchain Community  
from langchain\_community.embeddings.ollama import OllamaEmbeddings # Ollama embeddings for text representation  
from langchain\_community.document\_loaders import WebBaseLoader # Load documents from the web  
from langchain\_community.vectorstores import Chroma # Chroma vector store for efficient retrieval  
from langchain\_community.document\_loaders import PyPDFLoader # reading in pdfs  
from langchain.prompts import ChatPromptTemplate # class for promts

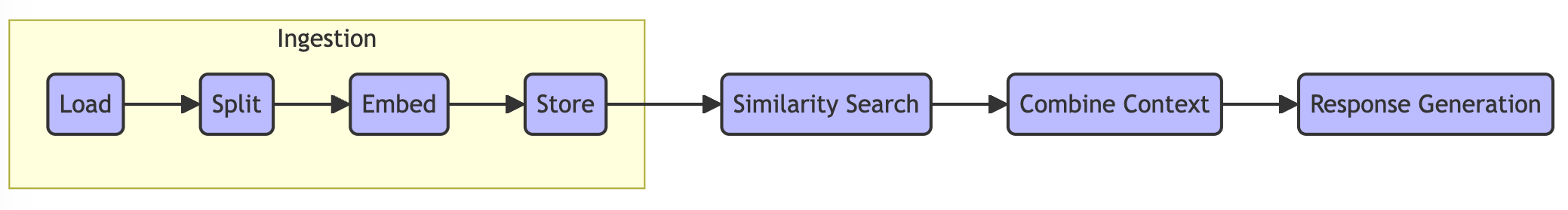
### Settings

OLLAMA\_LARGE\_LANGUAGE\_MODEL = "wizardlm2:7b" # Specifies the large language model version  
OLLAMA\_SERVER = "http://ollama:11434" # URL for the Ollama server

question = "What is the TMC Entrepreneurial Lab?"

### Ingestion

#### Load



[Document Loaders](https://python.langchain.com/docs/integrations/document_loaders/)

#### PDF Loader

# pdf document loader  
loader = PyPDFLoader(  
 "./backend/tmc\_tel\_lab.pdf"  
)  
docs = loader.load()

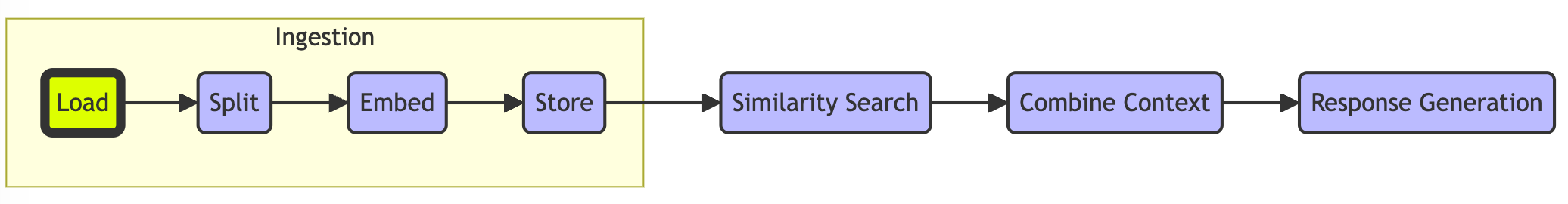
len(docs)

4

docs[0].page\_content[:300]

"Room to play\nAt the Entrepreneurial Lab, we believe in the power of multidisciplinary collaboration and\nthe freedom to explore. Here, diverse teams bringing together various skills to tackle the\nmultifaceted challenges of today. It's where individual competencies unite to solve\ncomplex problems, dri"

#### Split



[Splitter](https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/recursive_text_splitter/)

# Split  
text\_splitter = RecursiveCharacterTextSplitter.from\_tiktoken\_encoder(  
 chunk\_size=200,   
 chunk\_overlap=20  
)  
  
# Make splits  
splits = text\_splitter.split\_documents(docs)

This text splitter is the recommended one for generic text. It is parameterized by a list of characters. It tries to split on them in order until the chunks are small enough. The default list is ["\n\n", "\n", " ", ""]. This has the effect of trying to keep all paragraphs (and then sentences, and then words) together as long as possible, as those would generically seem to be the strongest semantically related pieces of text.

len(splits)

6

splits[0].page\_content[:500]

"Room to play\nAt the Entrepreneurial Lab, we believe in the power of multidisciplinary collaboration and\nthe freedom to explore. Here, diverse teams bringing together various skills to tackle the\nmultifaceted challenges of today. It's where individual competencies unite to solve\ncomplex problems, driving innovative solutions through collective expertise.\nThrough hands-on, experienced-based learning, we create an environment where\nexperimentation is encouraged, failures are embraced as opportuniti"

docs[0].page\_content

"Room to play\nAt the Entrepreneurial Lab, we believe in the power of multidisciplinary collaboration and\nthe freedom to explore. Here, diverse teams bringing together various skills to tackle the\nmultifaceted challenges of today. It's where individual competencies unite to solve\ncomplex problems, driving innovative solutions through collective expertise.\nThrough hands-on, experienced-based learning, we create an environment where\nexperimentation is encouraged, failures are embraced as opportunities for growth, and\nbreakthroughs are celebrated.\nWhether it's pioneering technology or disruptive healthcare solutions, the Entrepreneurial\nLab provides the space, resources, and collaborative spirit to turn your vision into a\nsuccessful venture.\nHow it works\nSounds promising, doesn't it? But how exactly does the Entrepreneurial Lab operate?\n> Our employeneurs work on their own innovation if they desire, but they do this\nalongside their client project.\n> TMC supports with a physical lab and possibly even with financial resources.\n> The ownership of the innovation remains of our employeneurs; TMC has no interest in\nit.\nBUILDING THE FUTURE IN\nThe Entrepreneurial Lab\nThis is the place where you can let your technical dreams come true. If you\nhave a groundbreaking idea for a new innovation, you can initiate your\nproject and take it from concept to reality with the support of a vibrant\ncommunity of like-minded innovators.\x00 \x00"

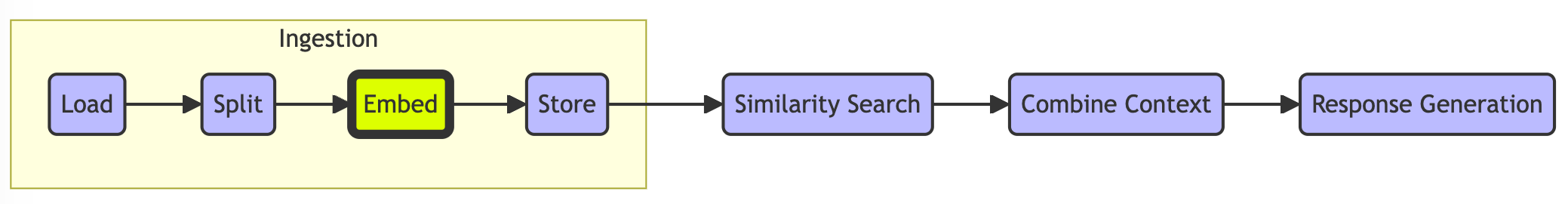
splits[0].page\_content

"Room to play\nAt the Entrepreneurial Lab, we believe in the power of multidisciplinary collaboration and\nthe freedom to explore. Here, diverse teams bringing together various skills to tackle the\nmultifaceted challenges of today. It's where individual competencies unite to solve\ncomplex problems, driving innovative solutions through collective expertise.\nThrough hands-on, experienced-based learning, we create an environment where\nexperimentation is encouraged, failures are embraced as opportunities for growth, and\nbreakthroughs are celebrated.\nWhether it's pioneering technology or disruptive healthcare solutions, the Entrepreneurial\nLab provides the space, resources, and collaborative spirit to turn your vision into a\nsuccessful venture.\nHow it works\nSounds promising, doesn't it? But how exactly does the Entrepreneurial Lab operate?\n> Our employeneurs work on their own innovation if they desire, but they do this\nalongside their client project."

splits[1].page\_content

'alongside their client project.\n> TMC supports with a physical lab and possibly even with financial resources.\n> The ownership of the innovation remains of our employeneurs; TMC has no interest in\nit.\nBUILDING THE FUTURE IN\nThe Entrepreneurial Lab\nThis is the place where you can let your technical dreams come true. If you\nhave a groundbreaking idea for a new innovation, you can initiate your\nproject and take it from concept to reality with the support of a vibrant\ncommunity of like-minded innovators.\x00 \x00'

#### Embed



[Text embedding models](https://python.langchain.com/docs/integrations/text_embedding/openai)

#### Text Embedding

OLLAMA\_EMBEDDING\_MODEL = "mxbai-embed-large"

embedding = OllamaEmbeddings(model=OLLAMA\_EMBEDDING\_MODEL, base\_url=OLLAMA\_SERVER)  
query\_result = embedding.embed\_query(question)  
split\_result = embedding.embed\_query(splits[0].page\_content)

len(query\_result)

1024

len(split\_result)

1024

split\_result[:4]

[0.6084602475166321,  
 0.10699772834777832,  
 -0.6222756505012512,  
 -0.09857344627380371]

#### Cosine Similarity

[Cosine similarity](https://platform.openai.com/docs/guides/embeddings/frequently-asked-questions) is reccomended (1 indicates identical).

def cosine\_similarity(vec1, vec2, print\_output = False):  
   
 dot\_product = np.dot(vec1, vec2)  
 norm\_vec1 = np.linalg.norm(vec1)  
 norm\_vec2 = np.linalg.norm(vec2)  
   
 similarity = dot\_product / (norm\_vec1 \* norm\_vec2)  
   
 if print\_output:  
 print("Cosine Similarity:", similarity)  
   
 return similarity

similarity = cosine\_similarity(query\_result, split\_result, True)

Cosine Similarity: 0.7252169697102998

#### HandsOn --- Stroopwafel

Now the big question: What item in Germany corresponds to what the stroopwafel is in the Netherlands.

OLLAMA\_EMBEDDING\_MODEL = "mxbai-embed-large"  
embedding = OllamaEmbeddings(model=OLLAMA\_EMBEDDING\_MODEL, base\_url=OLLAMA\_SERVER)

# Assuming embedding is some pre-trained embedding model with an embed\_query method  
words = ["Bratwurst", "Mercedes", "Schwarzwälder Kirschtorte", "Berliner", "Lebkuchen"]

# getting vectors of tokens/ words  
stroopwafel\_embedding = np.array(embedding.embed\_query("Stroopwafel"))  
netherlands\_embedding = np.array(embedding.embed\_query("Netherlands"))  
germany\_embedding = np.array(embedding.embed\_query("Germany"))

# calculating the comparison vector  
comparison\_embedding = stroopwafel\_embedding - (netherlands\_embedding - germany\_embedding)

# initiating variables  
highest\_similarity = -1  
closest\_word = None

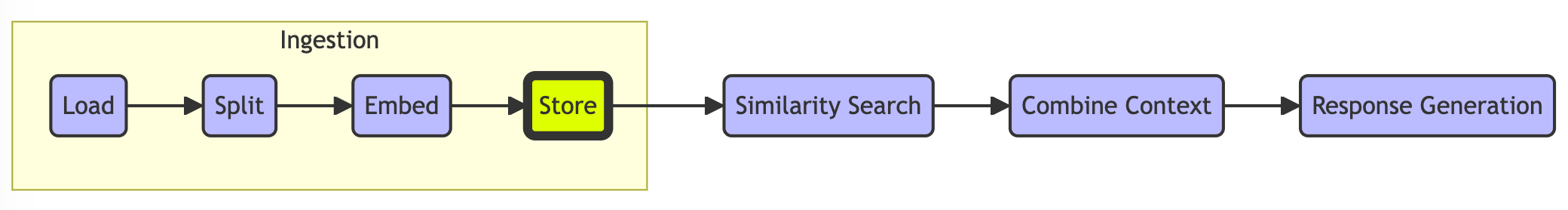
# running the loop  
for word in words:  
   
 # embedding the query word  
 word\_embedding = np.array(embedding.embed\_query(word))  
  
 # generating output  
 print(word)  
 similarity = cosine\_similarity(comparison\_embedding, word\_embedding, True)  
 print("")  
  
 # capturing highest similarity  
 if similarity > highest\_similarity:  
 highest\_similarity = similarity  
 closest\_word = word

Bratwurst  
Cosine Similarity: 0.7104238314729429  
  
Mercedes  
Cosine Similarity: 0.5495463777308662  
  
Schwarzwälder Kirschtorte  
Cosine Similarity: 0.6412197961550299  
  
Berliner  
Cosine Similarity: 0.7296874018682962  
  
Lebkuchen  
Cosine Similarity: 0.6988265639666382

# final evaluation  
print(f"The word closest to 'stroopwafel' is '{closest\_word}' with a cosine similarity of {highest\_similarity}.")

The word closest to 'stroopwafel' is 'Berliner' with a cosine similarity of 0.7296874018682962.

#### Store



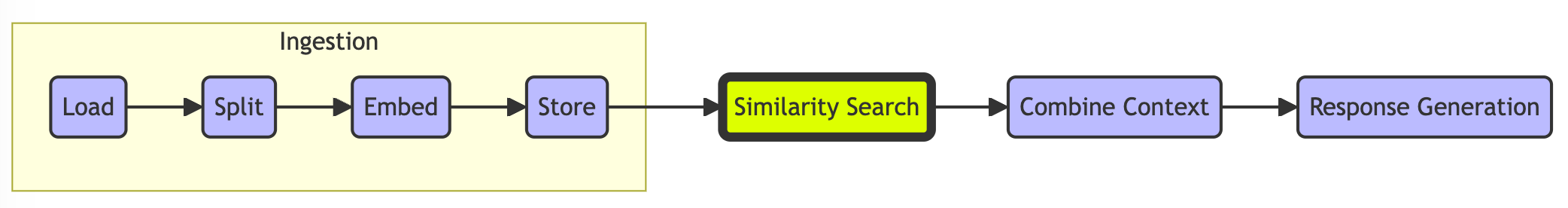
[Vectorstores](https://python.langchain.com/docs/integrations/vectorstores/)

vectorstore = Chroma.from\_documents(  
 collection\_name=OLLAMA\_EMBEDDING\_MODEL,  
 documents=splits,  
 embedding=OllamaEmbeddings(model=OLLAMA\_EMBEDDING\_MODEL, base\_url=OLLAMA\_SERVER)  
)

vectorstore.get(limit = 1)

{'ids': ['6ba9d156-ab8e-4b7b-b81e-a4a02f0b897a'],  
 'embeddings': None,  
 'metadatas': [{'page': 3, 'source': './backend/tmc\_tel\_lab.pdf'}],  
 'documents': ['Careers\nGraduate programs\nVIE Program\nCorporate vacancies\nAll vacancies\nService areas\nTechnology & Engineering\nDigital & IT\nEnergy & Renewables\nLife Sciences & Pharma\nDiscover TMC\nAbout us\nUpdates\nFAQ\nContact\n© 2024 TMC |Terms and conditions |Privacy statement |Cookie Statement |Settings'],  
 'uris': None,  
 'data': None,  
 'included': ['metadatas', 'documents']}

### Similarity Search

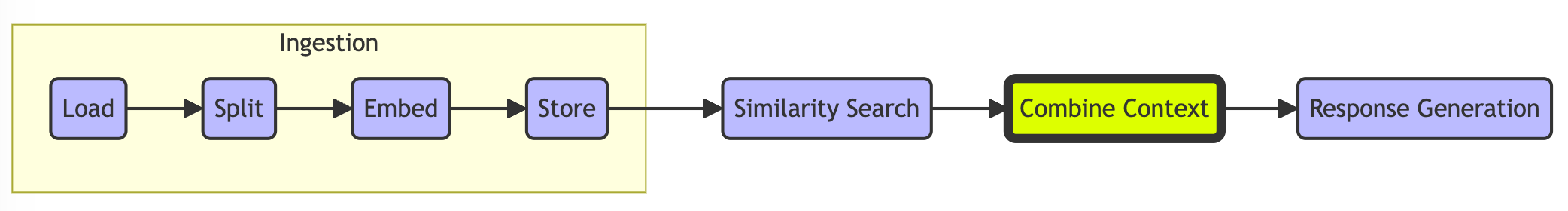


returned\_docs = vectorstore.similarity\_search\_with\_relevance\_scores(question, k = 2)

returned\_docs

[(Document(page\_content='alongside their client project.\n> TMC supports with a physical lab and possibly even with financial resources.\n> The ownership of the innovation remains of our employeneurs; TMC has no interest in\nit.\nBUILDING THE FUTURE IN\nThe Entrepreneurial Lab\nThis is the place where you can let your technical dreams come true. If you\nhave a groundbreaking idea for a new innovation, you can initiate your\nproject and take it from concept to reality with the support of a vibrant\ncommunity of like-minded innovators.\x00 \x00', metadata={'page': 0, 'source': './backend/tmc\_tel\_lab.pdf'}),  
 -111.32334465887976),  
 (Document(page\_content="Room to play\nAt the Entrepreneurial Lab, we believe in the power of multidisciplinary collaboration and\nthe freedom to explore. Here, diverse teams bringing together various skills to tackle the\nmultifaceted challenges of today. It's where individual competencies unite to solve\ncomplex problems, driving innovative solutions through collective expertise.\nThrough hands-on, experienced-based learning, we create an environment where\nexperimentation is encouraged, failures are embraced as opportunities for growth, and\nbreakthroughs are celebrated.\nWhether it's pioneering technology or disruptive healthcare solutions, the Entrepreneurial\nLab provides the space, resources, and collaborative spirit to turn your vision into a\nsuccessful venture.\nHow it works\nSounds promising, doesn't it? But how exactly does the Entrepreneurial Lab operate?\n> Our employeneurs work on their own innovation if they desire, but they do this\nalongside their client project.", metadata={'page': 0, 'source': './backend/tmc\_tel\_lab.pdf'}),  
 -139.5120848479825)]

### Combine Context



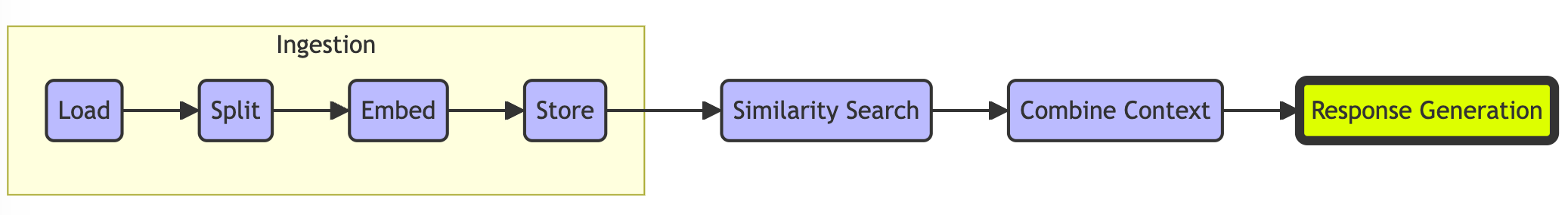
# Prompt  
template = """Answer the question based only on the following context:  
{context}  
  
Question: {question}  
"""  
  
prompt = ChatPromptTemplate.from\_template(template)  
prompt

ChatPromptTemplate(input\_variables=['context', 'question'], messages=[HumanMessagePromptTemplate(prompt=PromptTemplate(input\_variables=['context', 'question'], template='Answer the question based only on the following context:\n{context}\n\nQuestion: {question}\n'))])

# LLM  
llm = ChatOllama(model=OLLAMA\_LARGE\_LANGUAGE\_MODEL, base\_url=OLLAMA\_SERVER)

# Chain  
chain = prompt | llm

### Response Generation



[RAG chains](https://python.langchain.com/docs/expression_language/get_started#rag-search-example)

# Invoking the RAG chain  
response = chain.invoke({"context":returned\_doc[0], "question":question})  
response.content

' Based on the provided context, the TMC Entrepreneurial Lab is a facility or program that supports innovative projects initiated by its client project teams, which include employeneurs (self-employed professionals who work within an organization like TMC). The lab provides a physical space (a "vibrant community of like-minded innovators") where these teams can develop and build their ideas from conception to reality. It offers both the necessary infrastructure (a lab) and possibly financial support. Notably, TMC does not claim ownership over the innovations that are developed within its Entrepreneurial Lab; the intellectual property remains with the employeneurs who initiate these projects. The lab\'s purpose is to help build the future by fostering technological advancements and entrepreneurial spirit.'

In the chain.invoke() example above we used directly the result output of a similarity search of the vector database. Langchain has a better approach for this via retrievers.

#### Retriever

# here we create a retriever from the vectorstore which can perform similarity search and returns one document  
retriever = vectorstore.as\_retriever(search\_type="similarity", search\_kwargs={"k": 1}, )

# with this retreiver the context (relevant split) is directry passed to the question addressing the LLM.  
response = chain.invoke({"context":retriever,"question":question})  
response.content

'Based on the context provided, which includes \'Chroma\' and \'OllamaEmbeddings\' within tags and a reference to a `Chroma` object from the `langchain\_community.vectorstores` module, it seems like you are referring to a system or application that uses Chroma, which is a vector store for similarity retrieval, possibly in the context of a larger system or service named "TMC Entrepreneurial Lab."\n\nHowever, without additional context specific to "TMC Entrepreneurial Lab," it\'s not possible to provide a definitive answer about what it is. The term "TMC Entrepreneurial Lab" does not directly relate to the provided code or tags. It could be a separate entity, a project, or a program that utilizes similar technologies for search or data retrieval tasks, potentially leveraging vector embeddings (like those from \'OllamaEmbeddings\') stored in a Chroma vector store.\n\nIf "TMC Entrepreneurial Lab" is indeed a specific initiative, organization, or product that uses these technologies, you would need to consult documentation or resources specific to that entity to understand its purpose and functions. It could be an incubator for startups, a research lab focused on entrepreneurship, or any other initiative that employs such technology for its operations, such as to analyze startup pitches, market trends, or other data relevant to entrepreneurship using vector embeddings and similarity search capabilities.'

## HandsOn --- Direct LLM Question

How to run a directly question to the LLM without involving the retrieval augmented generation (RAG) part?

# LLM  
llm = ChatOllama(model=OLLAMA\_LARGE\_LANGUAGE\_MODEL, base\_url=OLLAMA\_SERVER)

answer = llm.invoke("Tell me a joke about the weather.")

answer.content

'Sure, here\'s a light-hearted weather joke for you:\n\nWhy don\'t clouds ever get cold?\n\nBecause most of their sweat evaporates before it reaches their feet! (A play on words with "evaporation" and "sweat")\n\nRemember, jokes about the weather are often as changeable as the forecast itself!'

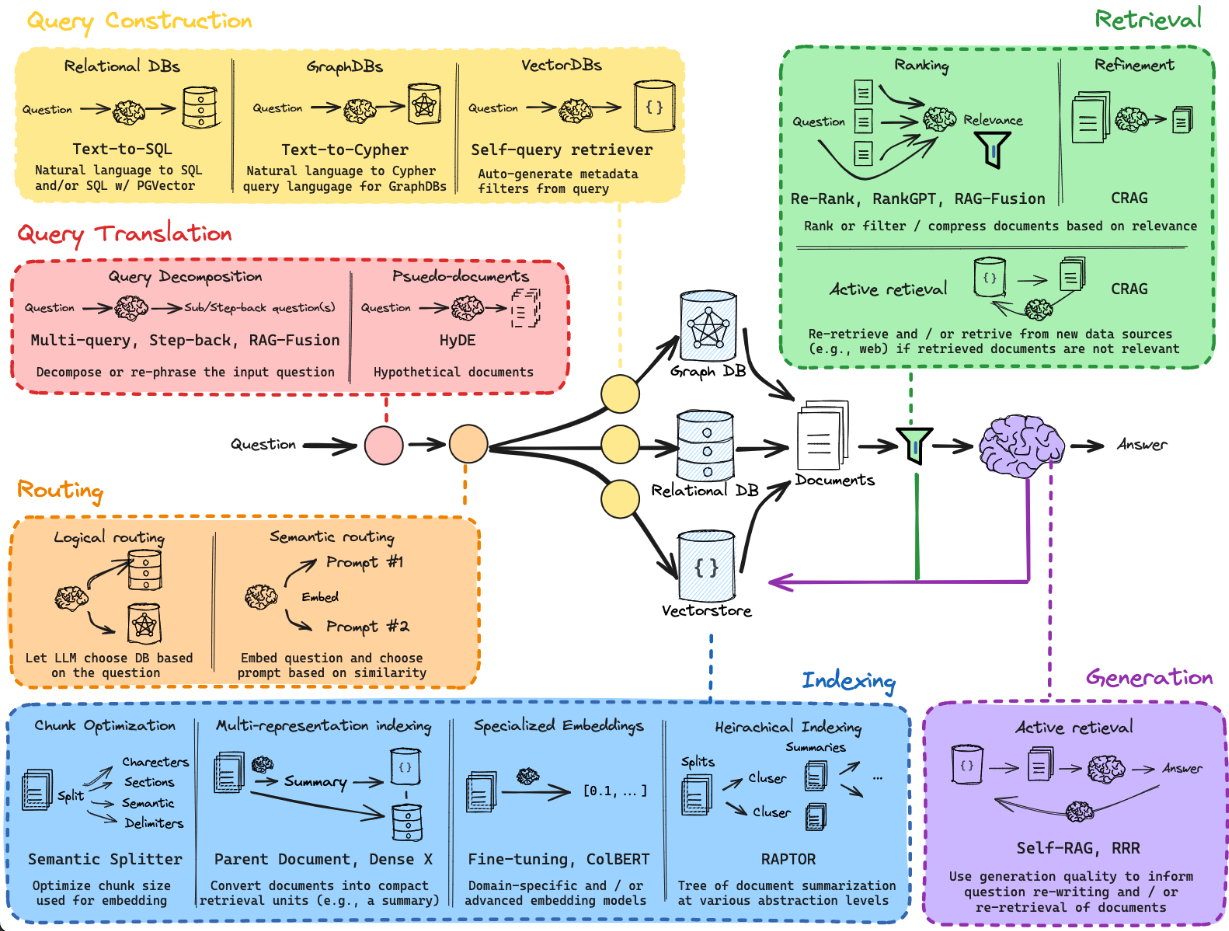
## Summary

Implementation of Simple Retrieval-Augmented Generation (RAG) from Scratch

* **Ingestion Phase**:
  + **Load**: Loading documents into the system.
  + **Split**: Splitting documents into manageable chunks.
  + **Embed**: Embedding document chunks into vector representations.
  + **Store**: Storing the embedded documents in a vector store.
* **Similarity Search**: Searching for relevant documents using embedded query.
* **Combine Context**: Combining the retrieved document context with the query.
* **Response Generation**: Generating the final response using a Large Language Model (LLM).

## Next Steps

There is plenty more to discover at [langchain's](https://github.com/langchain-ai) and many other websites! Especially check out: [YouTube](https://www.youtube.com/watch?v=sVcwVQRHIc8&ab_channel=freeCodeCamp.org) and [Github](https://github.com/langchain-ai/rag-from-scratch/tree/main). Here an overview:



## Thank You

* **Technical Support**: Milan and Raul
* **Organisational Support**: Marlies, Wendy, and Varsha
* **Motivational Support**: TMChampionship/ TEL/ Pepijn

## Feedback



[Feedback](https://forms.office.com/e/CwRvint3LY?origin=lprLink)

## Appendix

* **credits**: this notebook heavily borrows from langchain's [rag\_from\_scratch\_1\_to\_4.ipynb](%22https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_1_to_4.ipynb%22)

* **GPT**: Generative Pre-trained Transformer, a type of language model developed by OpenAI that generates human-like text using transformer architecture.
* **LLM**: Large Language Model, a machine learning model trained on vast amounts of text data to understand and generate human language.
* **Transformer**: Deep learning model using attention mechanism for context understanding and parallel processing, introduced in the "Attention is All You Need" paper.
* **Embedding Models**: Convert text to vector representations (e.g., BERT).
* **Generation Models**: Generate text from prompts (e.g., GPT-3).
* **Softmax Function**: Converts values to probabilities, used in classification models.
* **Fine-Tune vs. Retrieval "Augmented Generation**
  + **Fine-Tuning an LLM**: Adapts model to specific tasks using labeled data.
  + **RAG (Retrieval-Augmented Generation)**: Combines retrieval with generation for context-specific responses.
* **Micro Timeline**
  + **2017**: "Attention is All You Need" paper.
  + **2018**: BERT, GPT-2
  + **2020**: GPT-3.
* **Quantization**: Reduces precision of model parameters.
  + **Benefits**: Smaller size, faster inference, lower power consumption.
  + **Types**: Static, Dynamic, Quantization-Aware Training.
  + **Challenges**: Accuracy loss, hardware support needed.
* **Not all LLMs are GPTs**: Other models include BERT, T5, XLNet, RoBERTa.
* **Not all LLMs use transformers**: Other architectures include RNNs, CNNs, MoE, Memory-Augmented Networks.