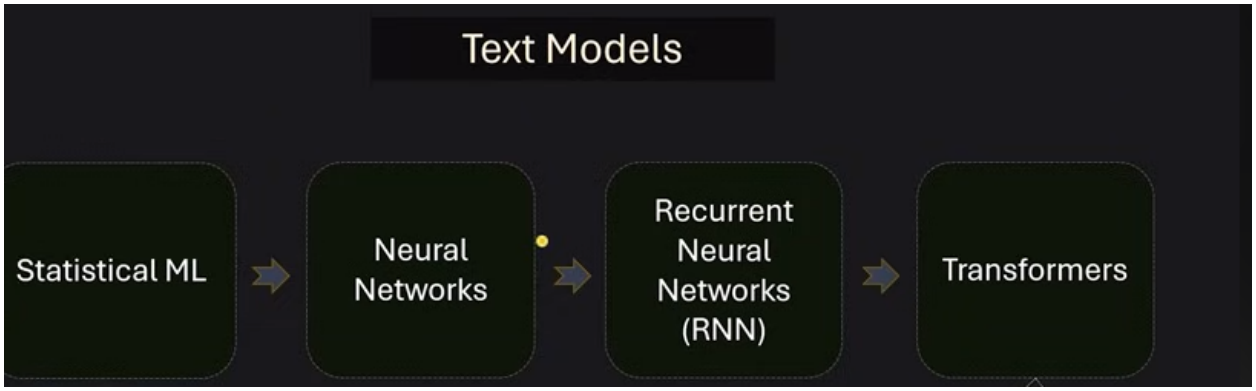
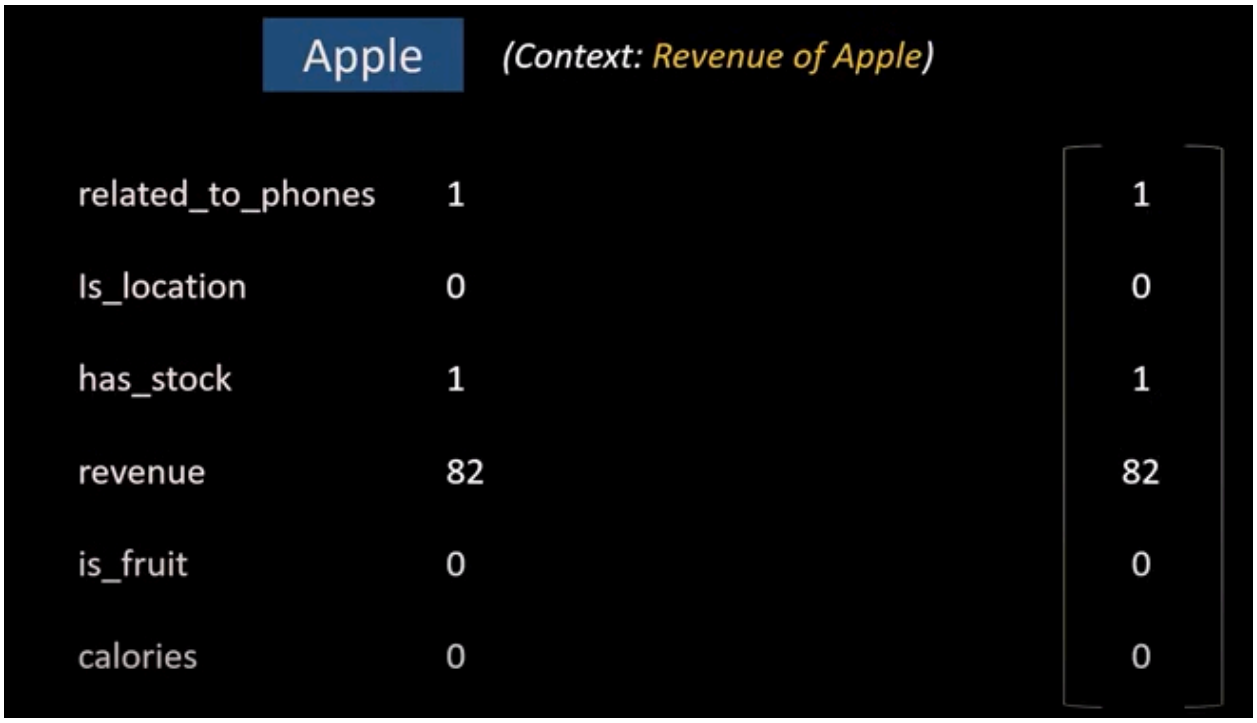
Language Model is an AI model that can predict the next word (or set of words) for a given sequence of words.



Embeddings - A numeric representation of text in the form of a vector such that you can capture the meaning of that text. Once you create embeddings for a given text you can do math with the words and sentences.
Example - Paris - France + India = Delhi

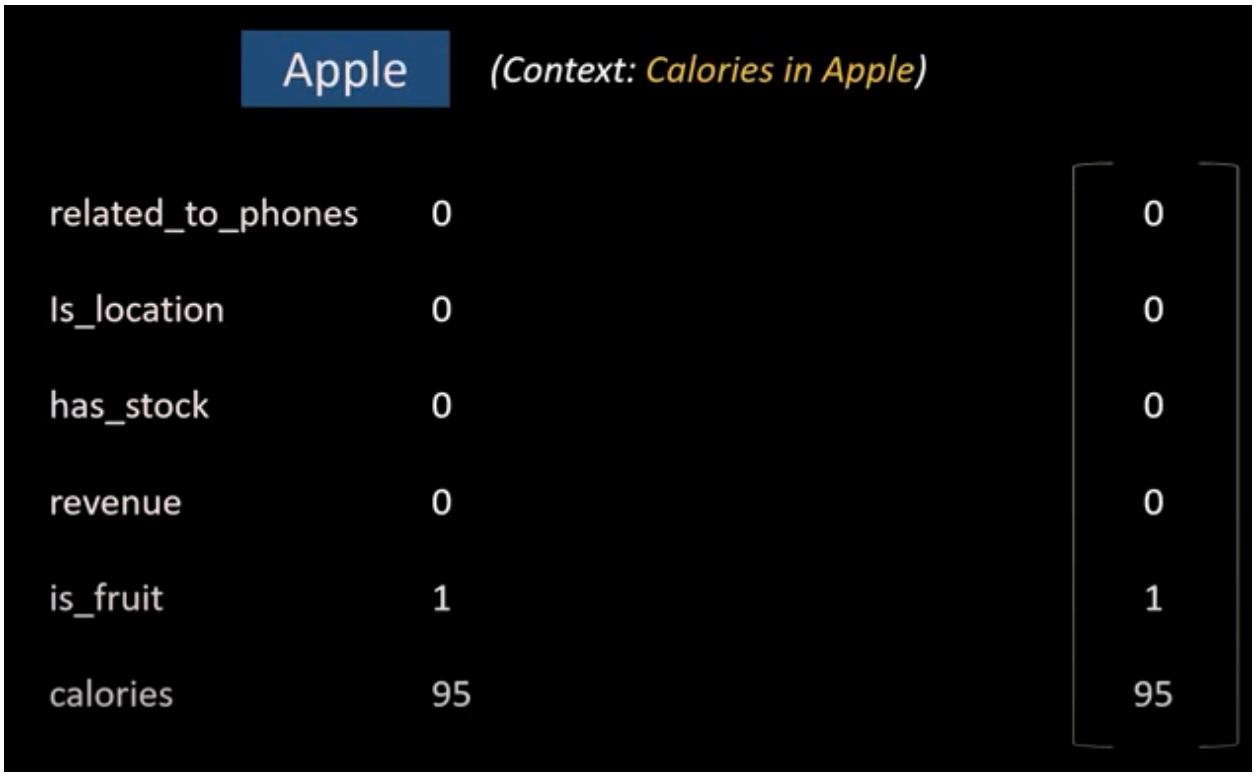
Vector database - allows you to store those embeddings and perform efficient search on these embeddings.
Example -

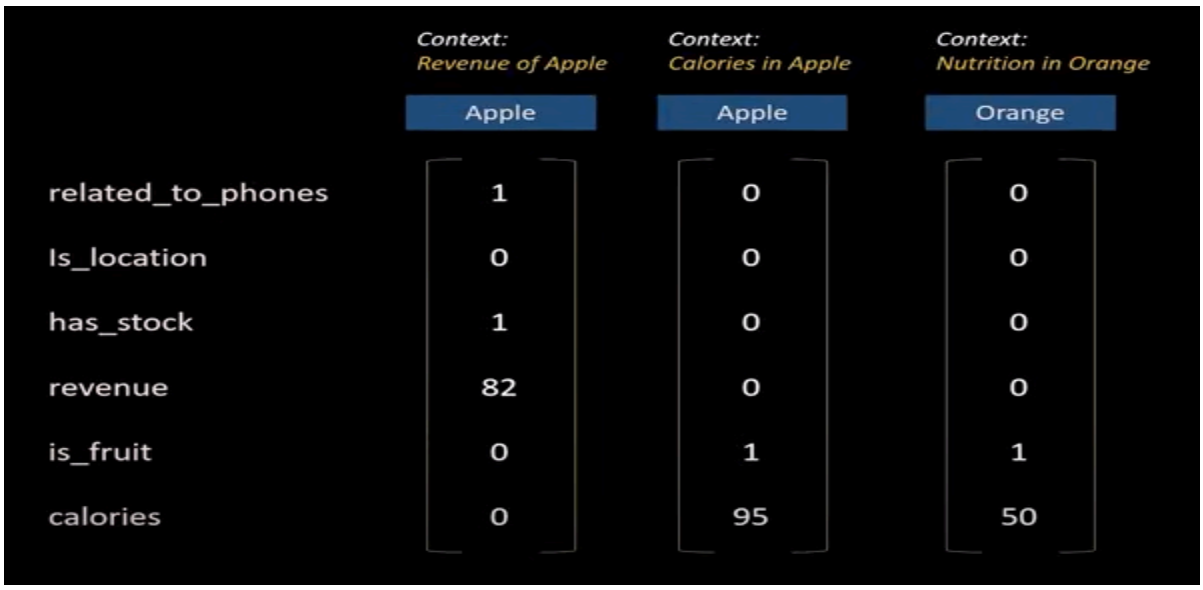


An entity like "Apple" can be encoded into a vector format based on specific attributes and their values.

This structured representation allows for efficient data management and retrieval, especially in applications
involving large-scale data and machine learning, where such vectors can be used to compare, classify, and analyze
entities based on their attributes.

Apple fruit -





Locality Sensitive Hashing (LSH) is a technique used to approximate nearest neighbor searches in high-dimensional spaces, which is particularly useful when dealing with large datasets. The main idea behind LSH is to hash input items in such a way that similar items are more likely to be hashed into the same buckets. This significantly reduces the search space and speeds up the process of finding similar items.

RAG, or Retrieval-Augmented Generation, is a framework designed to improve the performance of natural language generation tasks by combining the strengths of retrieval-based models and generative models. It was introduced by Facebook AI (now Meta AI) to leverage external knowledge during the generation process, which helps in creating more accurate and informative responses, especially in complex queries or tasks requiring specific information.
Example of RAG -

Langchain is a framework that allows you to build applications using LLMs.

The "chain" concept in Large Language Models (LLMs) like those used in LangChain refers to a series of linked operations or prompts that are executed sequentially to achieve a complex task. This concept is often referred to as "chaining" and it is used to break down a complicated process into smaller, manageable steps. Each step builds upon the previous one, creating a chain of operations that ultimately leads to the desired outcome.

Large Language Models (LLMs) like OpenAI's GPT models, "temperature" is a parameter that controls the randomness of the model's output. It influences the diversity of the generated text by adjusting the probability distribution of the next word or token in the sequence.

How Temperature Works -

Temperature = 1: This is the default setting. It results in a balance between randomness and predictability. The model samples from the probability distribution as it is.
Temperature < 1: This makes the model's output more focused and deterministic. Lowering the temperature reduces randomness, making the model more likely to choose high-probability words. A very low temperature (close to 0) makes the output more repetitive and deterministic.
Temperature > 1: This increases the randomness of the output. Higher temperature values make the model more likely to choose less probable words, leading to more diverse and creative outputs. However, very high temperatures can produce incoherent or nonsensical results.

High Temperature (e.g., 1.0 or 0.9):

```
from openai import OpenAI
llm = OpenAI(temperature=1.0)
response = llm("Once upon a time,")
print(response)
```

The model will generate text with a good balance of creativity and coherence.

Low Temperature (e.g., 0.2 or 0.5):

```
llm = OpenAI(temperature=0.2)
response = llm("Once upon a time,")
print(response)
```

The model will generate more predictable and repetitive text, staying closer to common phrases and structures.

Very High Temperature (e.g., 1.5):

```
llm = OpenAI(temperature=1.5)
response = llm("Once upon a time,")
print(response)
```

The output will be more random and creative, but it might also be less coherent.

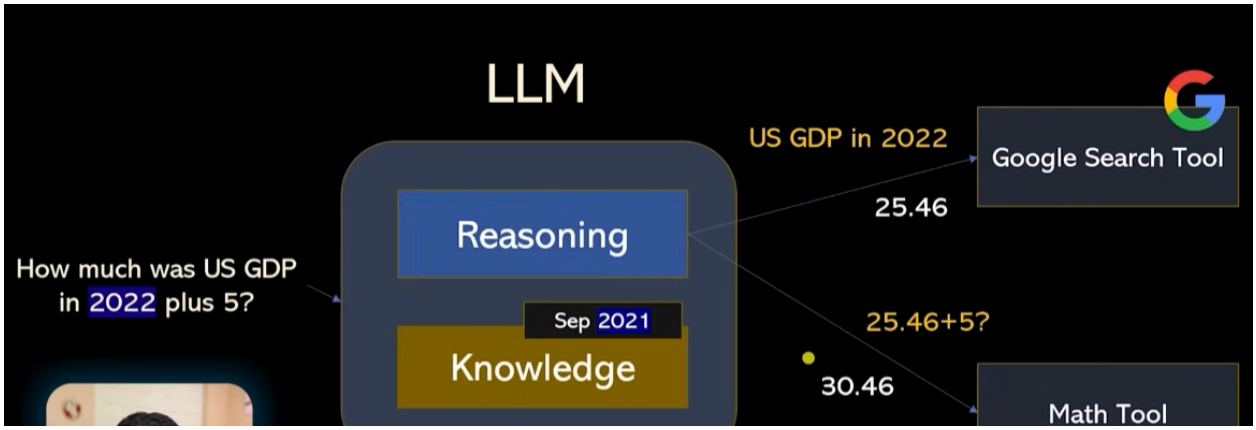
Sequential chaining is a technique used with large language models (LLMs) to perform tasks that require multiple steps. It involves connecting several LLMs or LLM processes together in a specific order, where the output of one step becomes the input for the next.

Examples -

Predicting the Next Word: Let's say you input the phrase "The cat sat on the" into the language model. The model processes each word sequentially, using the information from the preceding words to predict the next one. In this case, based on its training data, it might predict that the next word is "mat." So, it would generate the phrase "The cat sat on the mat."

Understanding Context: If you input the sentence "She opened the door and saw a huge," the model processes each word in the sequence, understanding that "door" is typically followed by something that can be described as "huge." So, it might predict that the next word is "monster." Thus, it generates the conclusion: "She opened the door and saw a huge monster."

Generating Coherent(logically connected, consistent, and makes sense as a whole.) Text: When generating longer passages of text, the model continues to process tokens sequentially, building up a coherent understanding of the context. For example, if you prompt the model with the beginning of a story, it will continue generating text based on the preceding tokens, maintaining consistency and coherence throughout the passage.



Reasoning - The reasoning component in Large Language Models (LLMs) refers to the model's ability to perform tasks that require understanding, inference, deduction, and the application of logic to arrive at conclusions.

Inference- involves drawing conclusions from implicit information or context. It is about understanding what is not directly stated but can be logically assumed based on the given data.

Example of Inference:
Input: "The ground is wet. It must have rained."

Inference: Even though the input doesn't explicitly state that it rained,
the model infers that rain is a likely cause of the wet ground based on common knowledge and context.

Deduction involves drawing specific conclusions from general principles or premises that are explicitly provided. It is a logical process where the conclusion necessarily follows from the given premises.

Example of Deduction: Input: "If a person is a student, then they have access to the library. John is a student."

Deduction: The model deduces that "John has access to the library."

AgentType.ZERO_SHOT_REACT_DESCRIPTION - Zero-Shot Learning: The agent does not require task-specific training examples. Instead, it can understand and perform the task based on the description provided to it.

Reasoning and Acting: The agent can reason about the task and make decisions or take actions based on the information and instructions it receives.

AgentType.ZERO_SHOT_REACT_DESCRIPTION is a powerful approach for building versatile AI agents capable of understanding and performing tasks based on descriptions alone, leveraging zero-shot learning and reasoning capabilities. This makes it suitable for applications where tasks can be varied and are described dynamically.