

# Who Am I?

**Paulo Dichone**

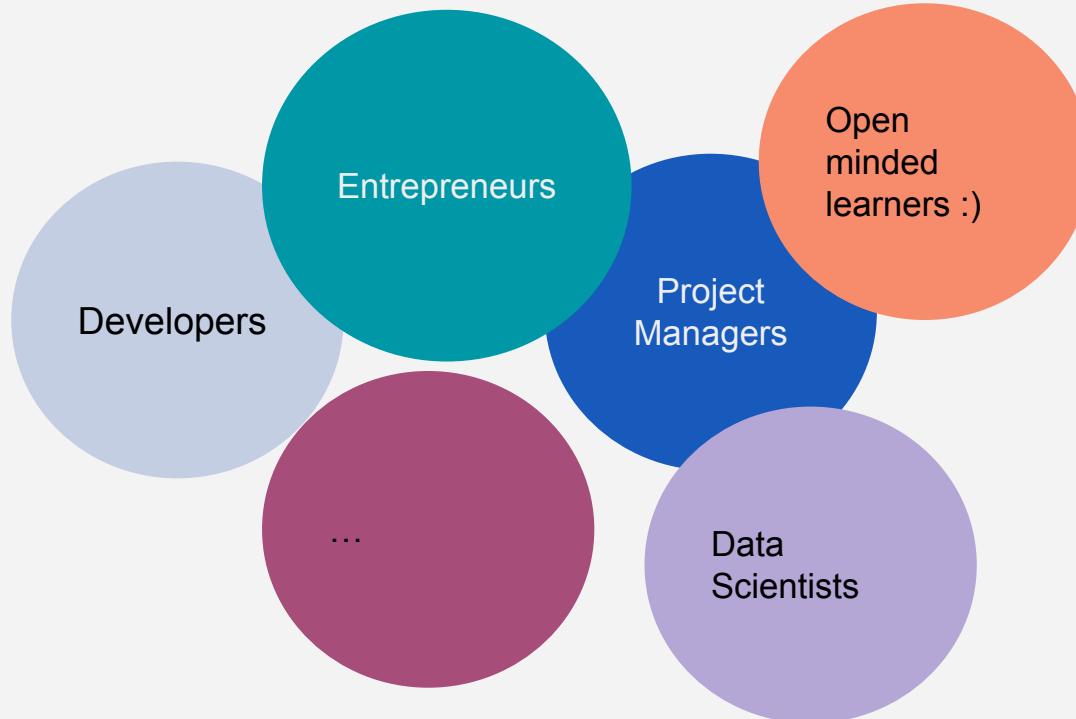
Software, Cloud, AI Engineer  
and Instructor



# What Is This Course About?

- Generative AI Mastery & LLM Development
  - GenAI
  - LLMs (Large Language Models)
  - Deep Learning\*
  - Machine Learning\*
  - Prompting
  - RAG
  - Agents
  - Fine-tuning
  - Building AI/LLM-based Applications
  - **Comprehensive Course**

# Who Is This Course For



# Course Prerequisites

1. Basics of programming
  - a. *We will be using Python*
2. *I'll include some Python basics (you can always skip)*
3. Willingness to learn :)

# Course Structure

Theory (Fundamental Concepts)

Mixture of both

**Hands-on**

***Python Fundamentals*** -- you can skip this

# Why Python?

GenAI....

Machine Learning

Deep Learning

LLMs (Large Language Models)

*Python*

# How to Get the Most from this Course

- Don't just watch lectures, take notes and code along.
- If you don't understand something, rewatch the lecture, then do some research, then ask questions in the course forum.
- Take your time -- this course is yours, indefinitely!
- Enjoy the journey!

# Development Environment Setup

- *Python setup (Win, Mac)*
- *VS Code*
  - *extensions...*

# Python Deep Dive

GenAI....

Machine Learning

Deep Learning

LLMs (Large Language Models)

*Python*

# Development Environment setup

- Python
- VS Code (or any other code editor)
- OpenAI Account and an OpenAI API Key

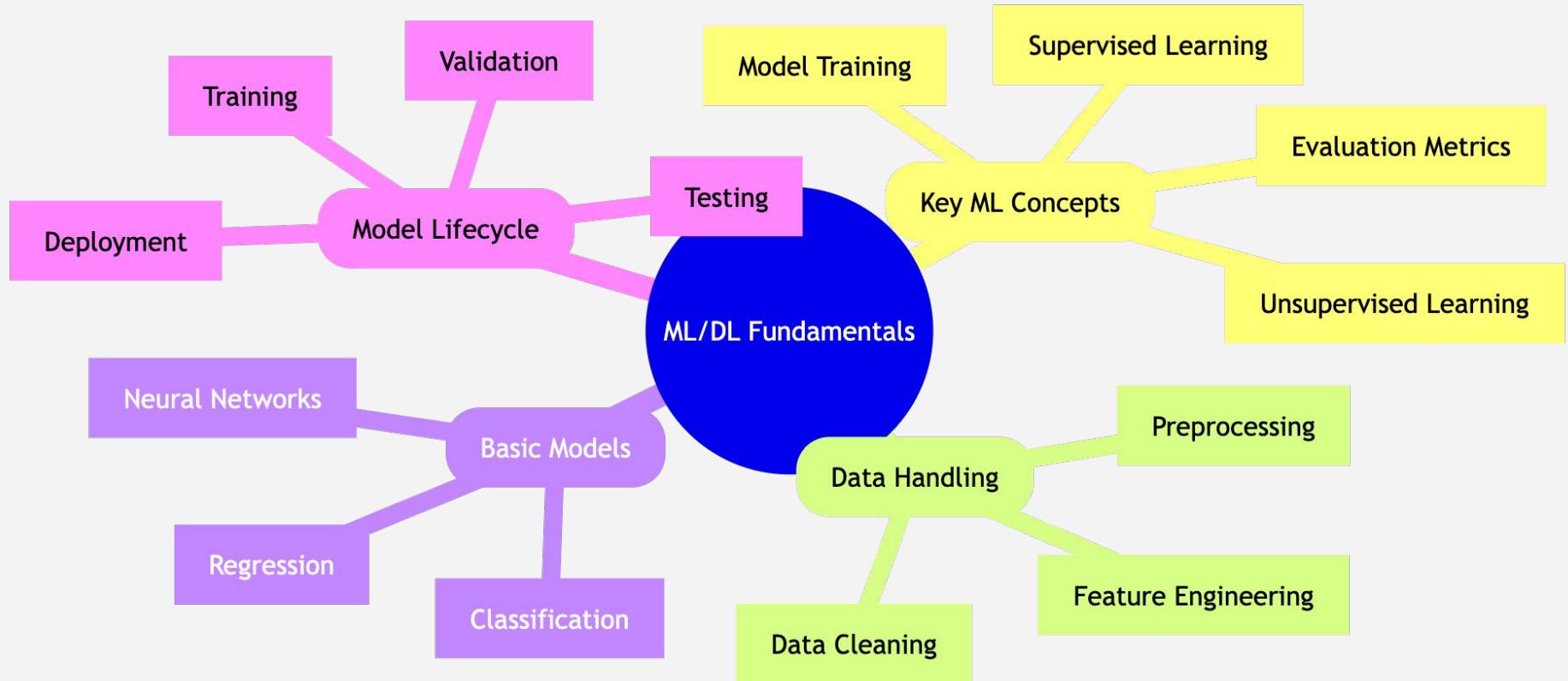
# Set up OpenAI API Account

***\*\* Please note*** that you will need an API key to use OpenAI services, and there may be some costs associated with using the API. However, these costs should be minimal.

# The core of AI - Deep & Machine Learning

# Machine Learning & Deep Learning *Crash Course*

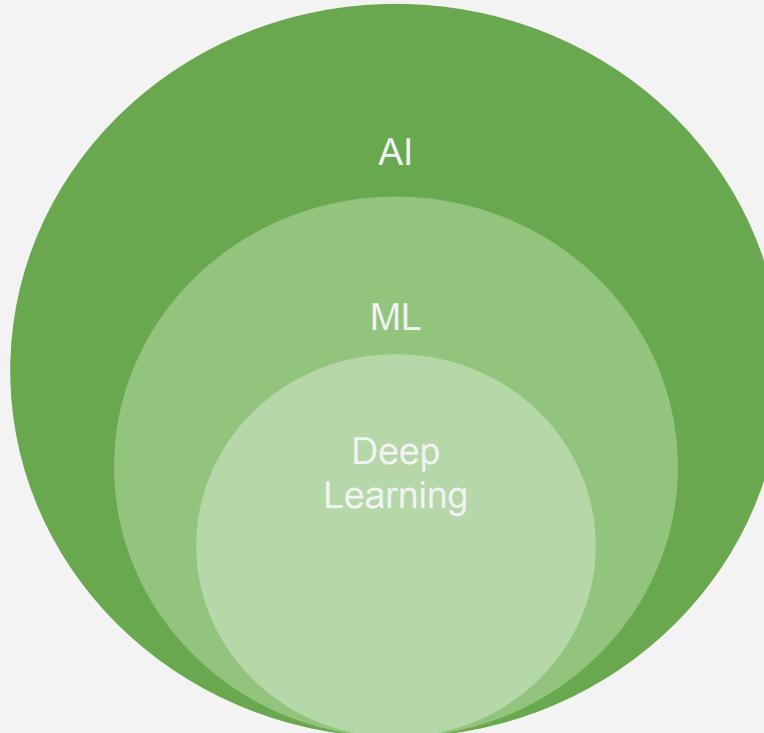
- *Motivation*
- *Key concepts*
- *Building blocks*



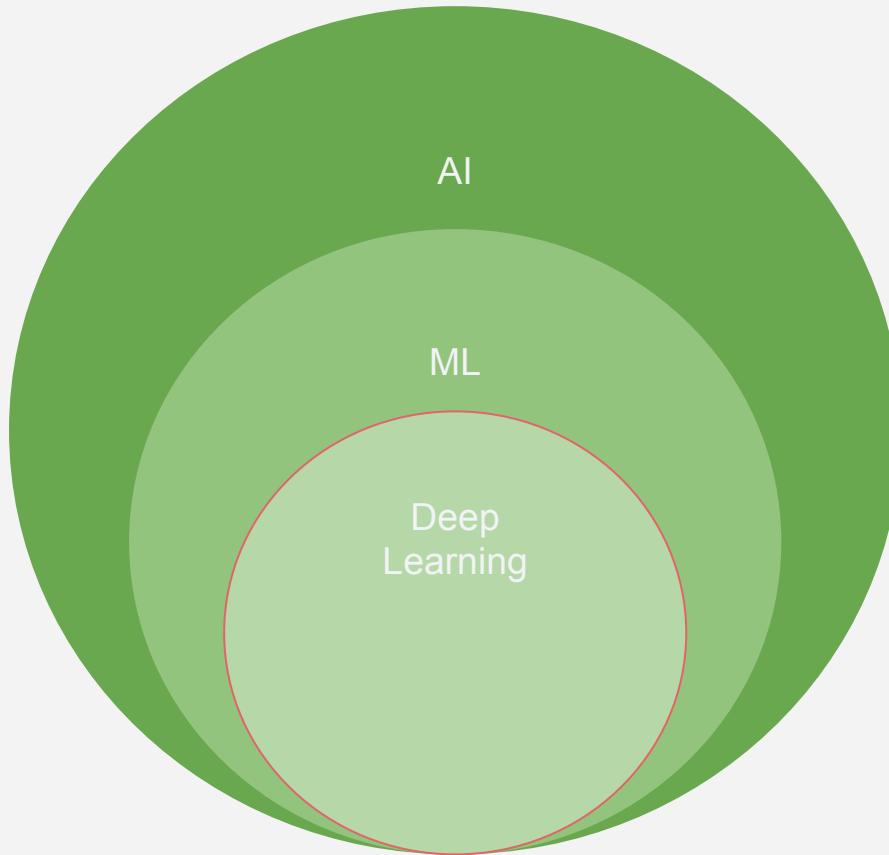
# Machine & Deep Learning

**AI** - Artificial Intelligence:

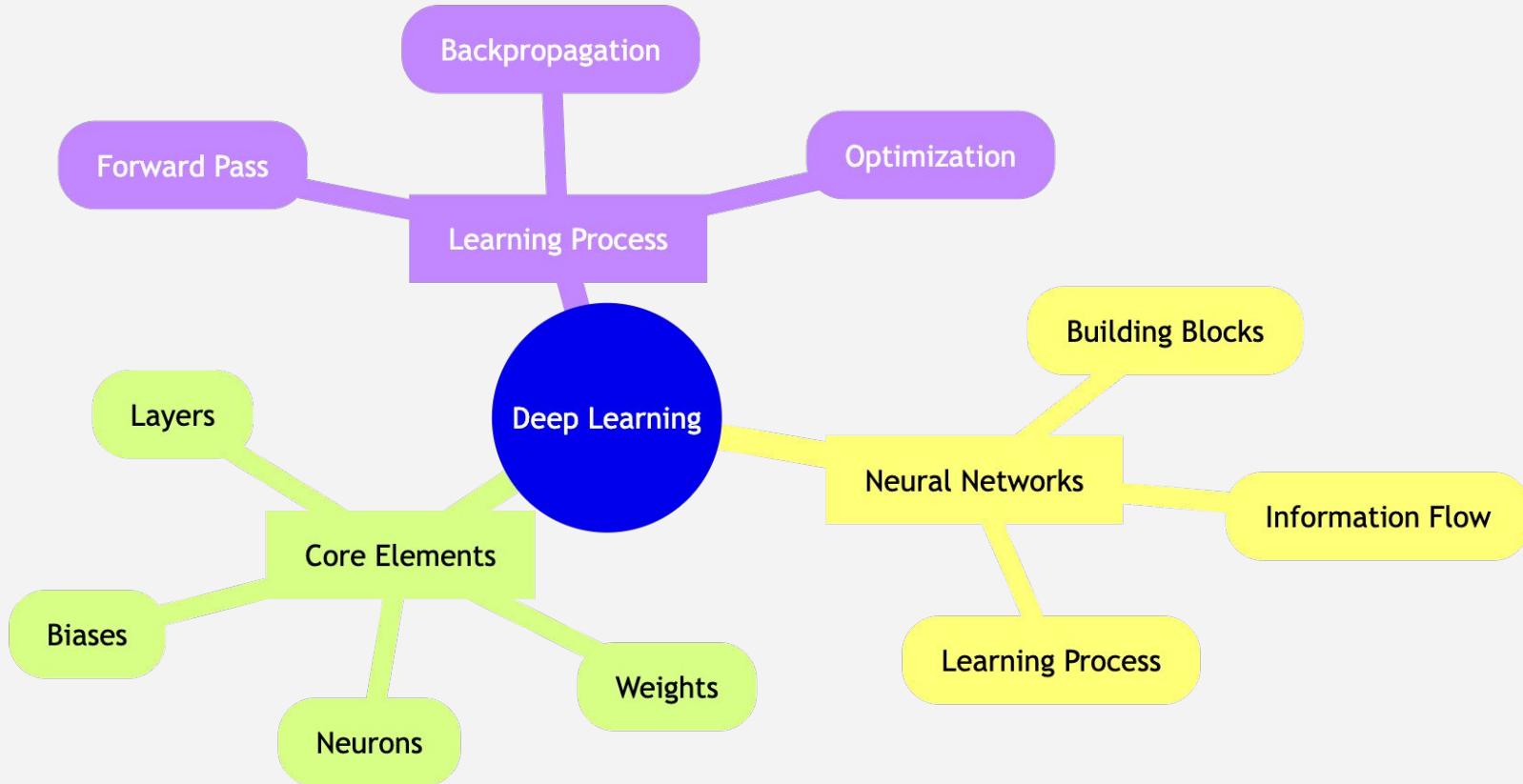
- Simulation of human intelligence by machines.



# Deep Learning



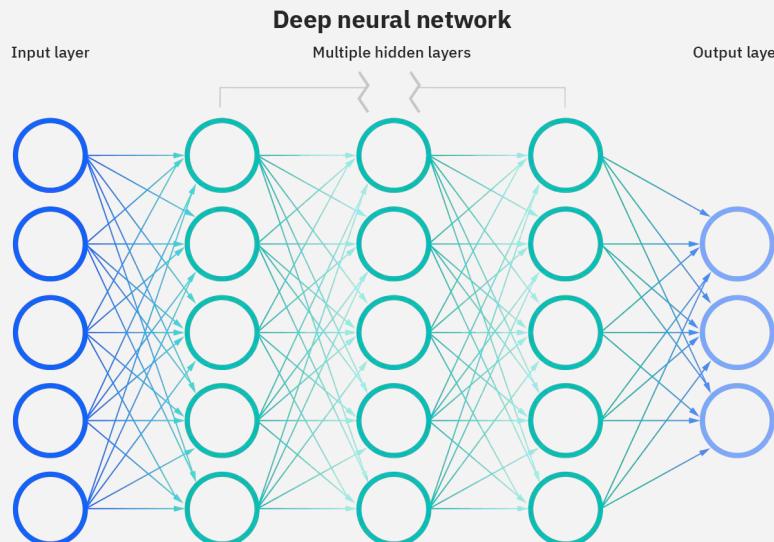
# Deep Learning Key Concepts



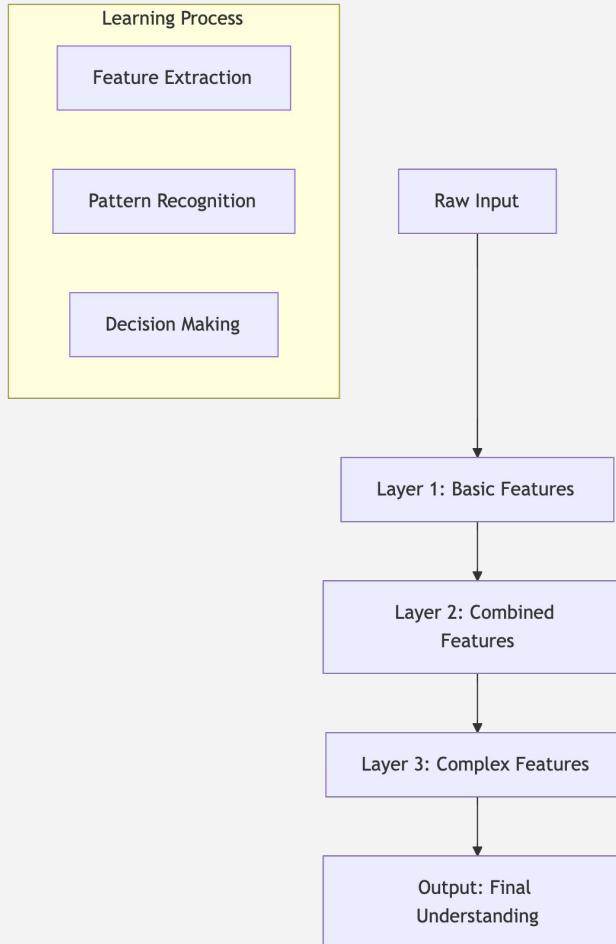
# Deep Learning Overview

Neural networks with multiple layers learn from data by:

- Automatically discovering features
- Learning hierarchical representations
- Transforming raw input into desired output

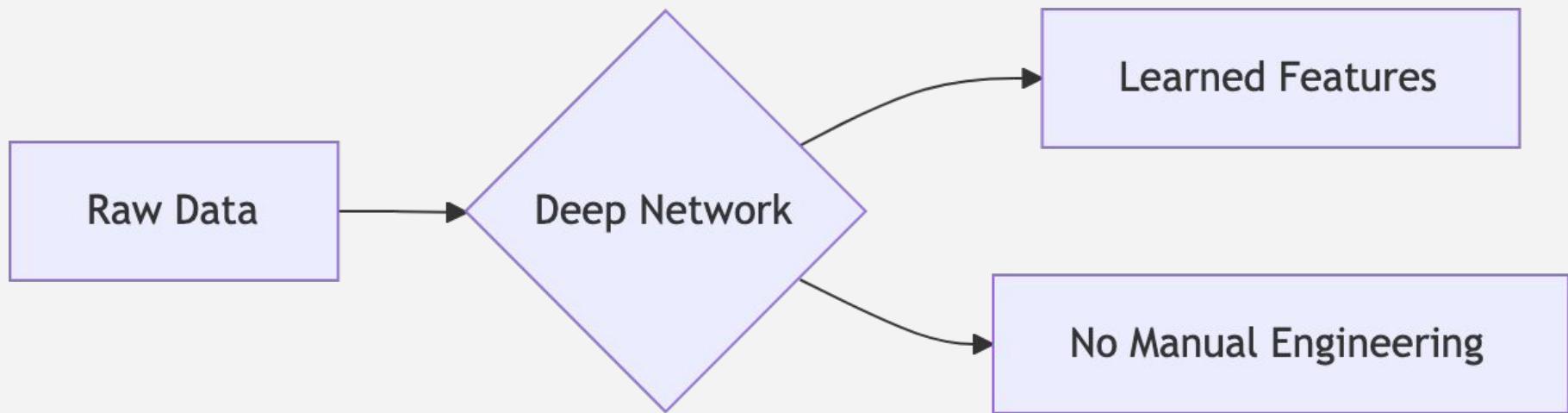


# Deep Learning - Breakdown



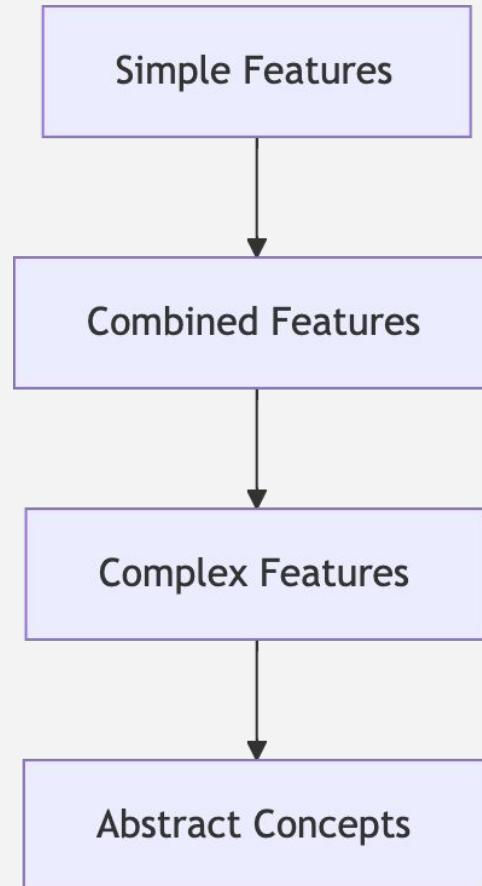
# Deep Learning - Key aspects

## 1. Automatic Feature Learning



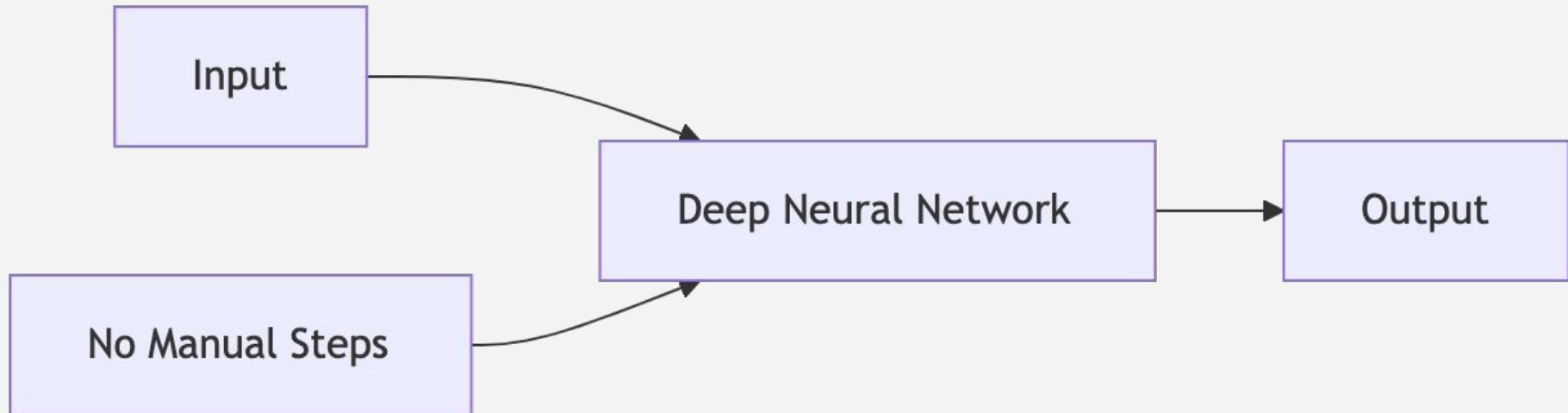
# Deep Learning - Key aspects

## 2. Hierarchical Structure



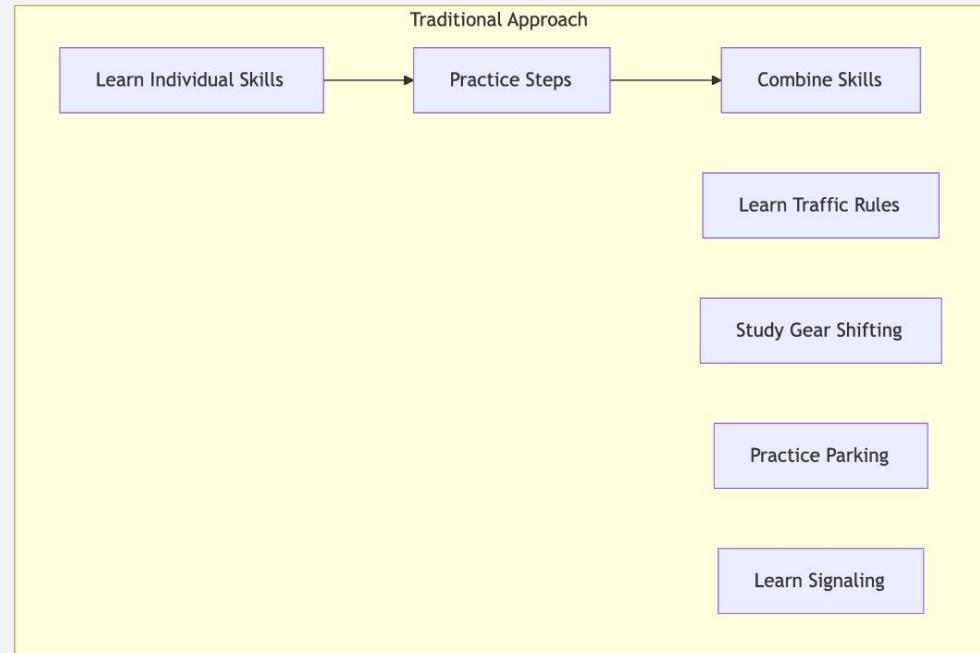
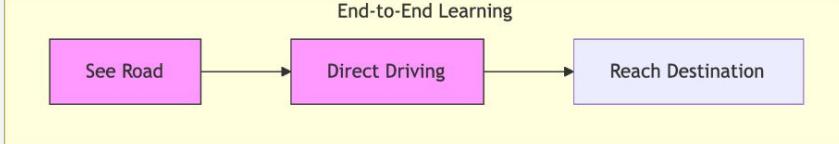
# Deep Learning - Key aspects

## 3. End-to-End Learning



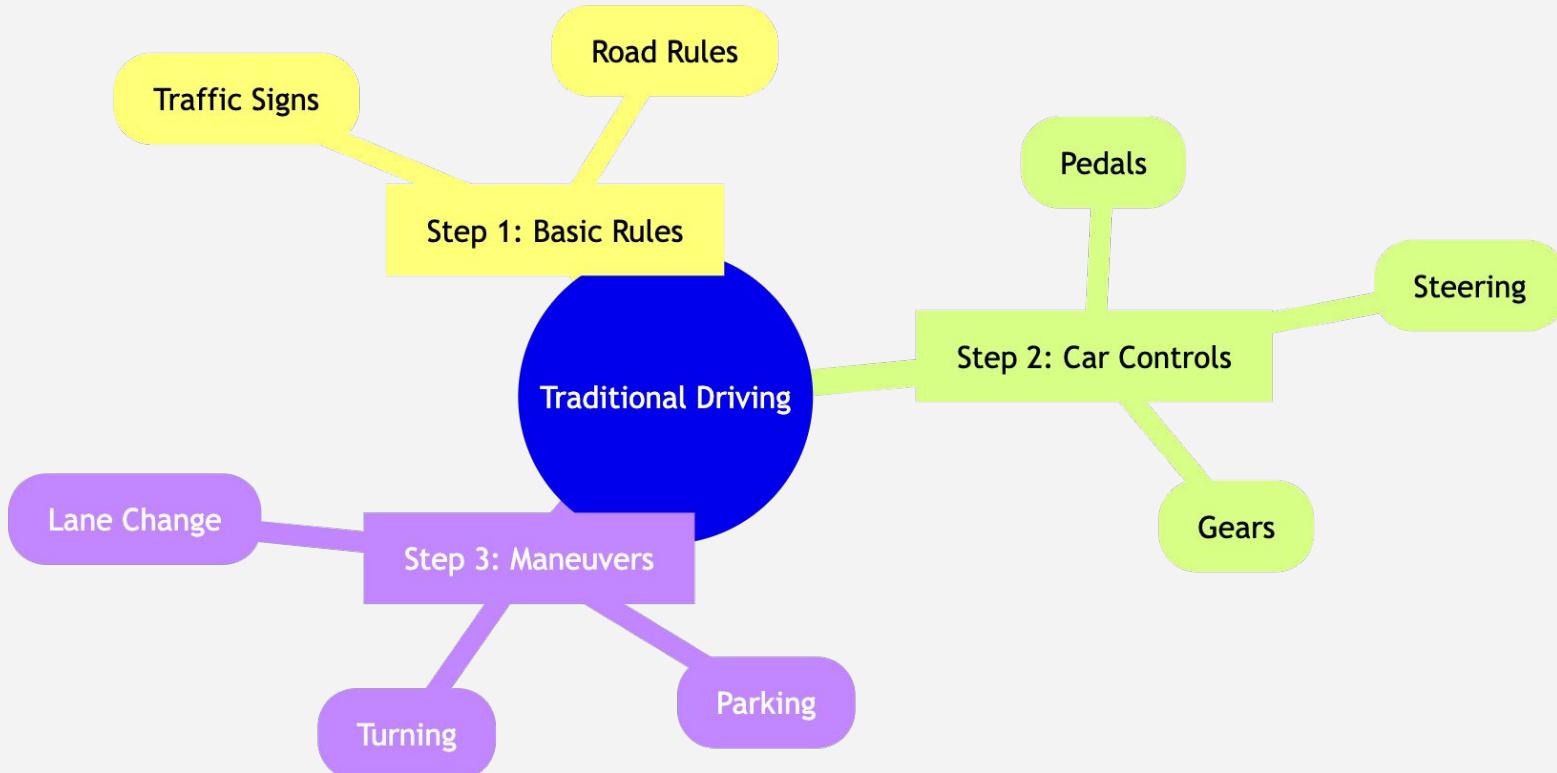
# Deep Learning - Key aspects

## 3. End-to-End Learning - (analogy) Understanding End-to-End Learning 🚗



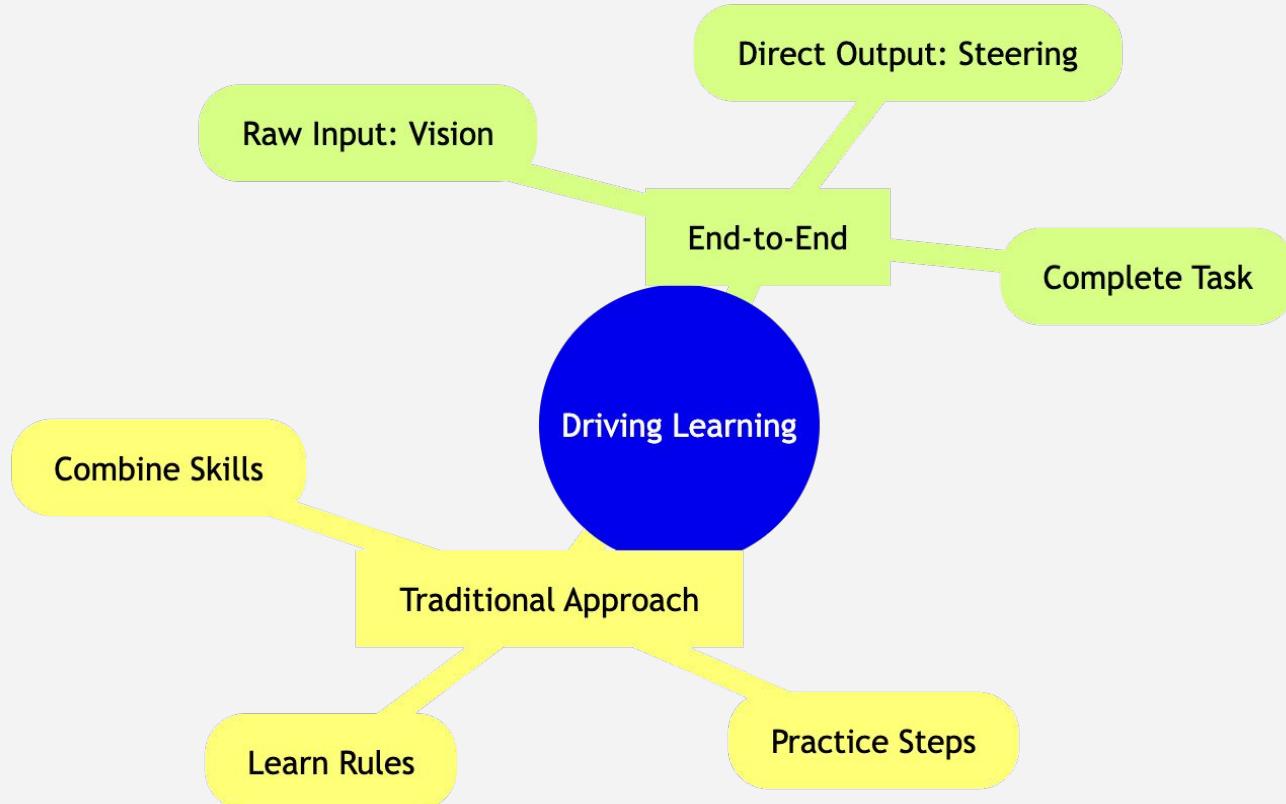
# Deep Learning - Key aspects

Traditional driving - Breakdown



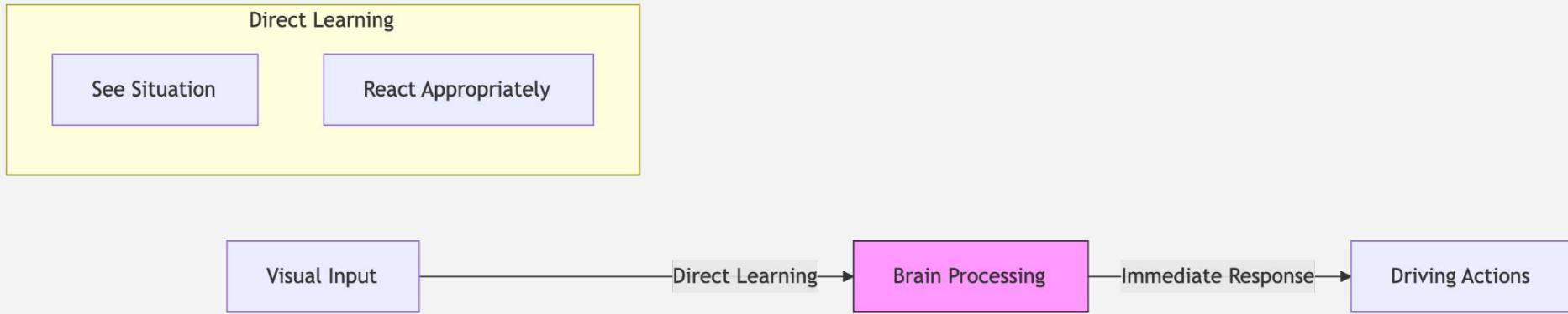
# Deep Learning - Key aspects

## 3. End-to-End Learning - (analogy) Understanding End-to-End Learning 🚗



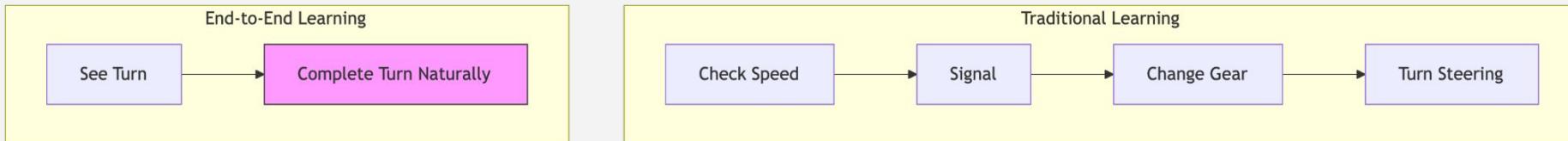
# Deep Learning - Key aspects

## 3. End-to-End Learning - (analogy) Understanding End-to-End Learning 🚗

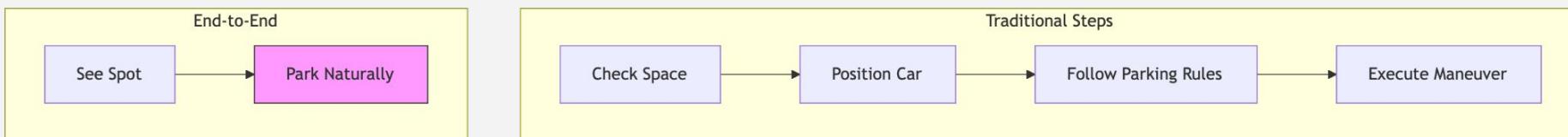


# Comparison Through Scenario

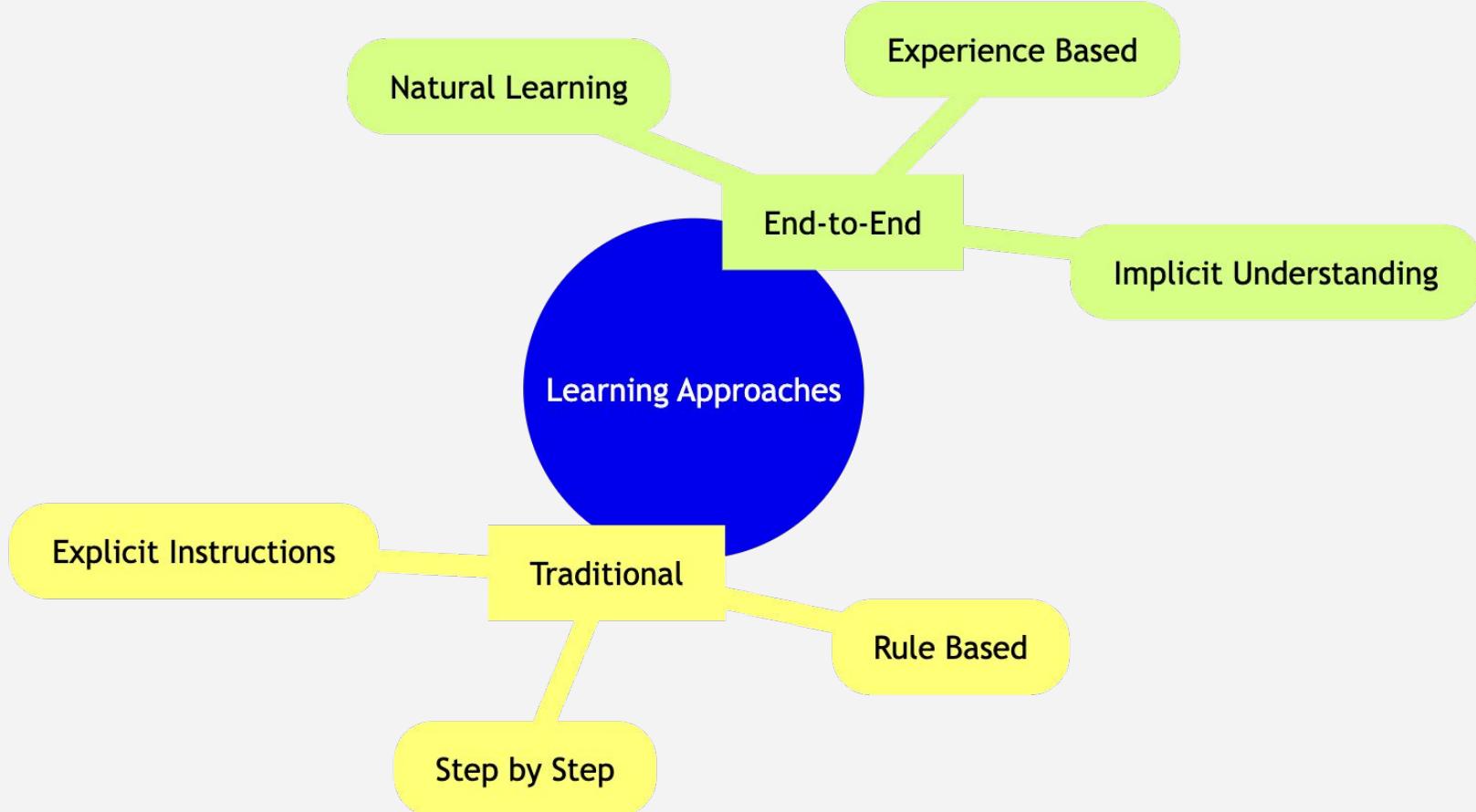
## Scenario 1: Approaching a Turn



## Scenario 2: Parking



# Key Differences

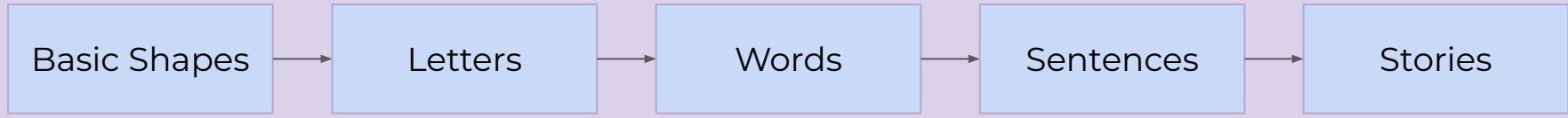


# Deep Learning - Real-World Analogy

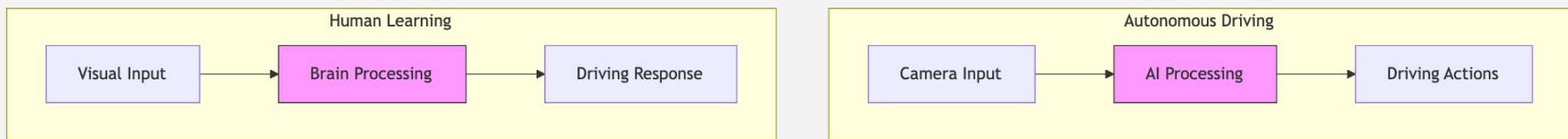
## Deep Learning Parallel



## Child Learning Process



# Deep Learning - Real-World Analogy



# **Key Takeaways**

## **1. Natural Learning**

- a. Direct input to output mapping
- b. No explicit rules required
- c. Natural pattern recognition

## **2. Efficiency**

- a. Faster response time
- b. More natural behavior
- c. Better adaptation

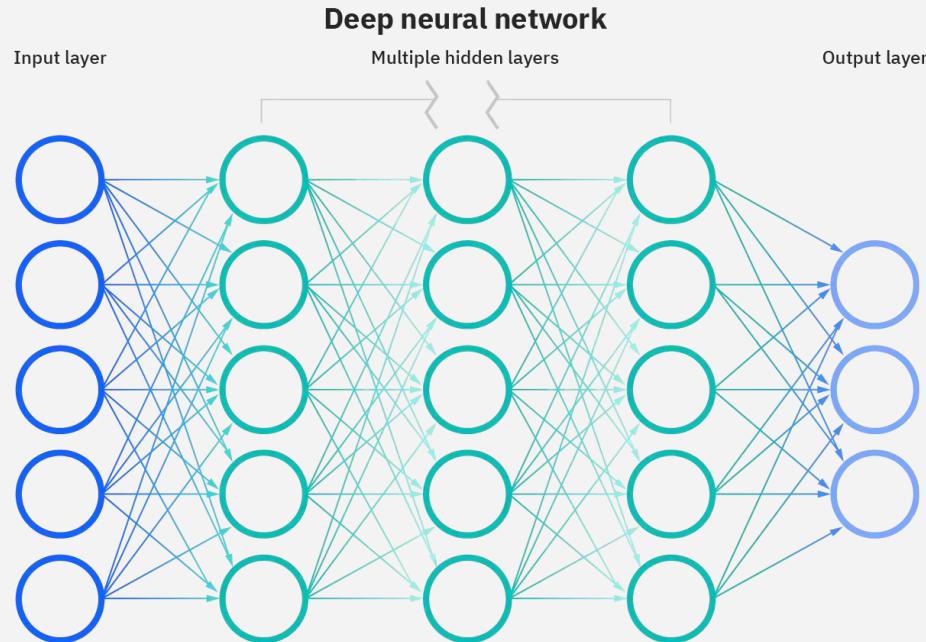
## **3. Integration**

- a. Seamless skill combination
- b. Contextual understanding
- c. Adaptive responses

# Deep Learning (DL)

- Uses neural networks (computational models inspired by the human brain).

**Deep Learning**

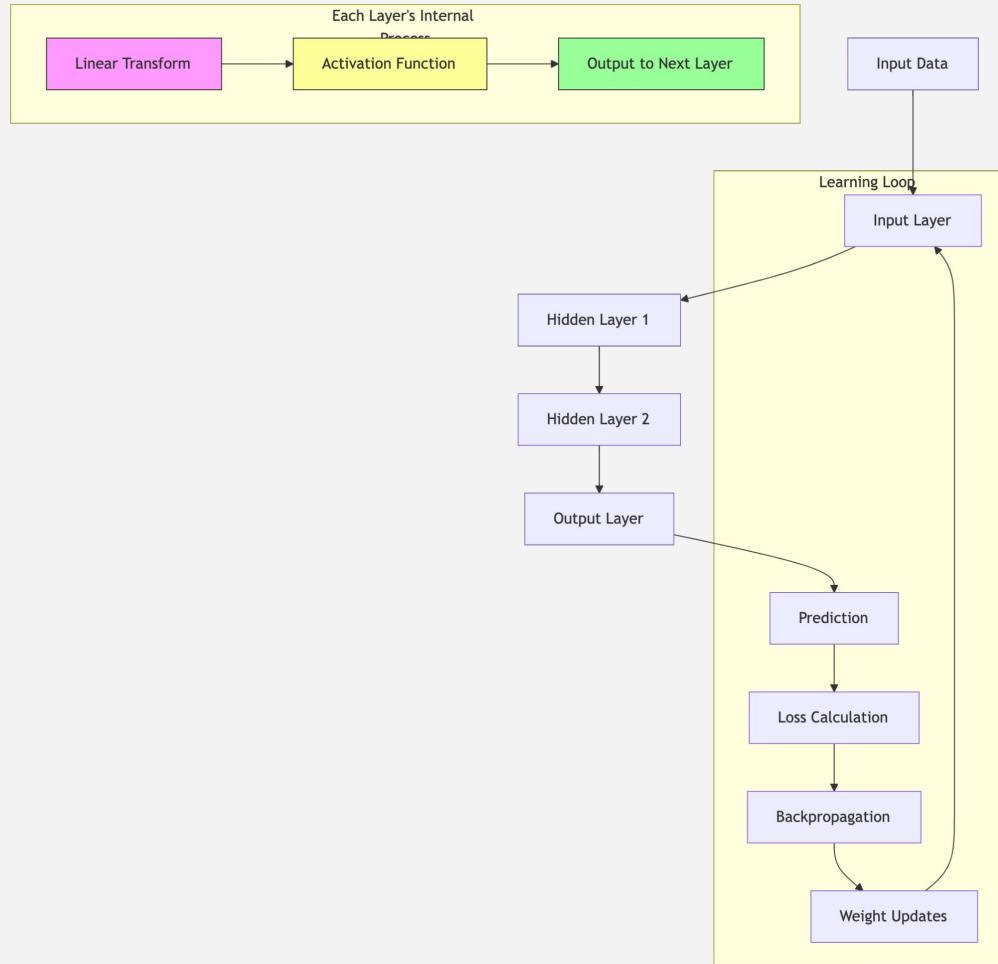


# Key Components of Neural Networks

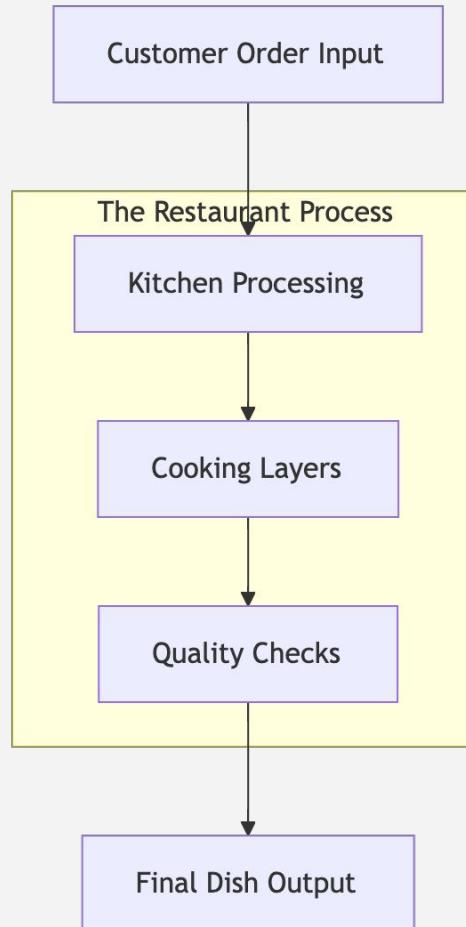
## Neural Networks

- **Neurons and Layers**
  - Input Layer: data receivers
  - Hidden layers: perform calculations to pass downstream
  - Output Layer: produces the final result (prediction/classification)
- **Activation Functions:** decision-makers (*is input important enough to pass forward?*)
- **Training with Backpropagation:** feedback and improvement

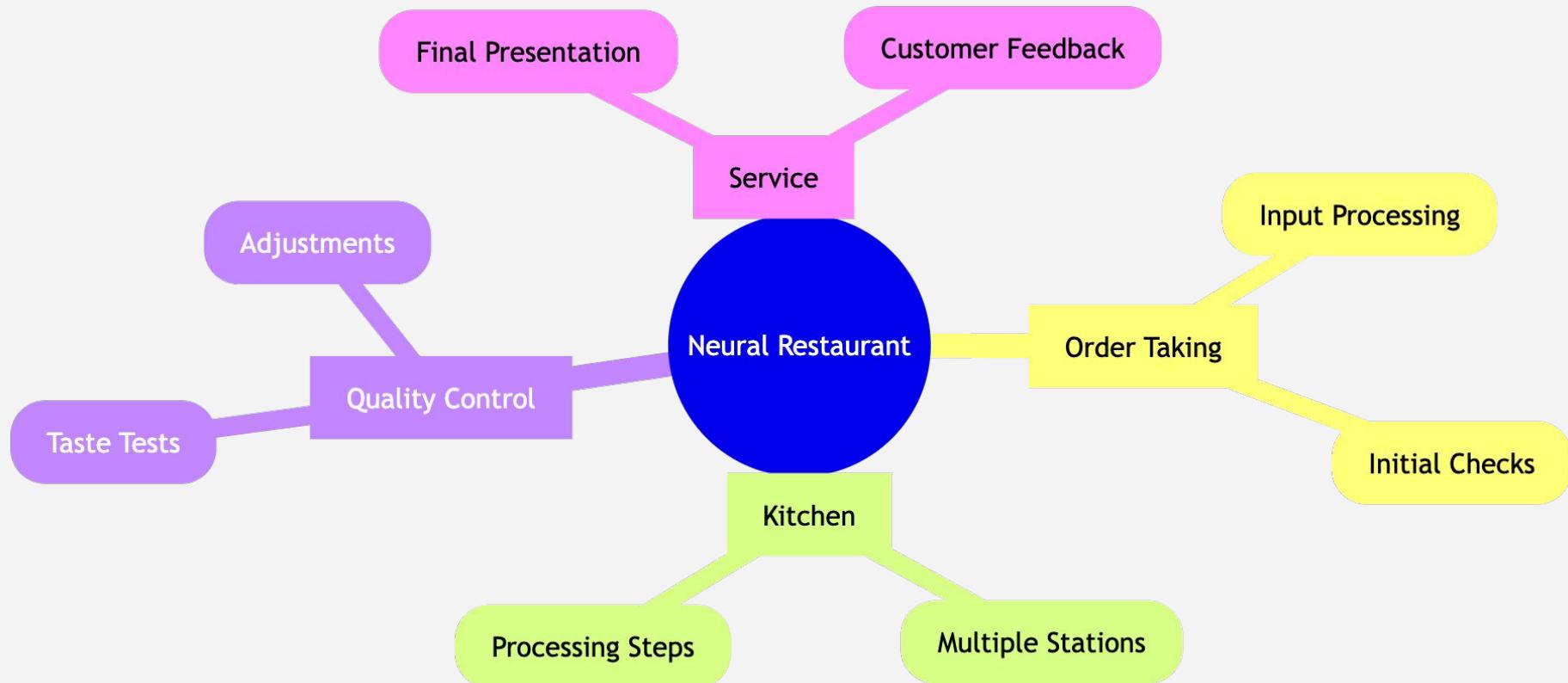
# The Neural Network - Deep Dive



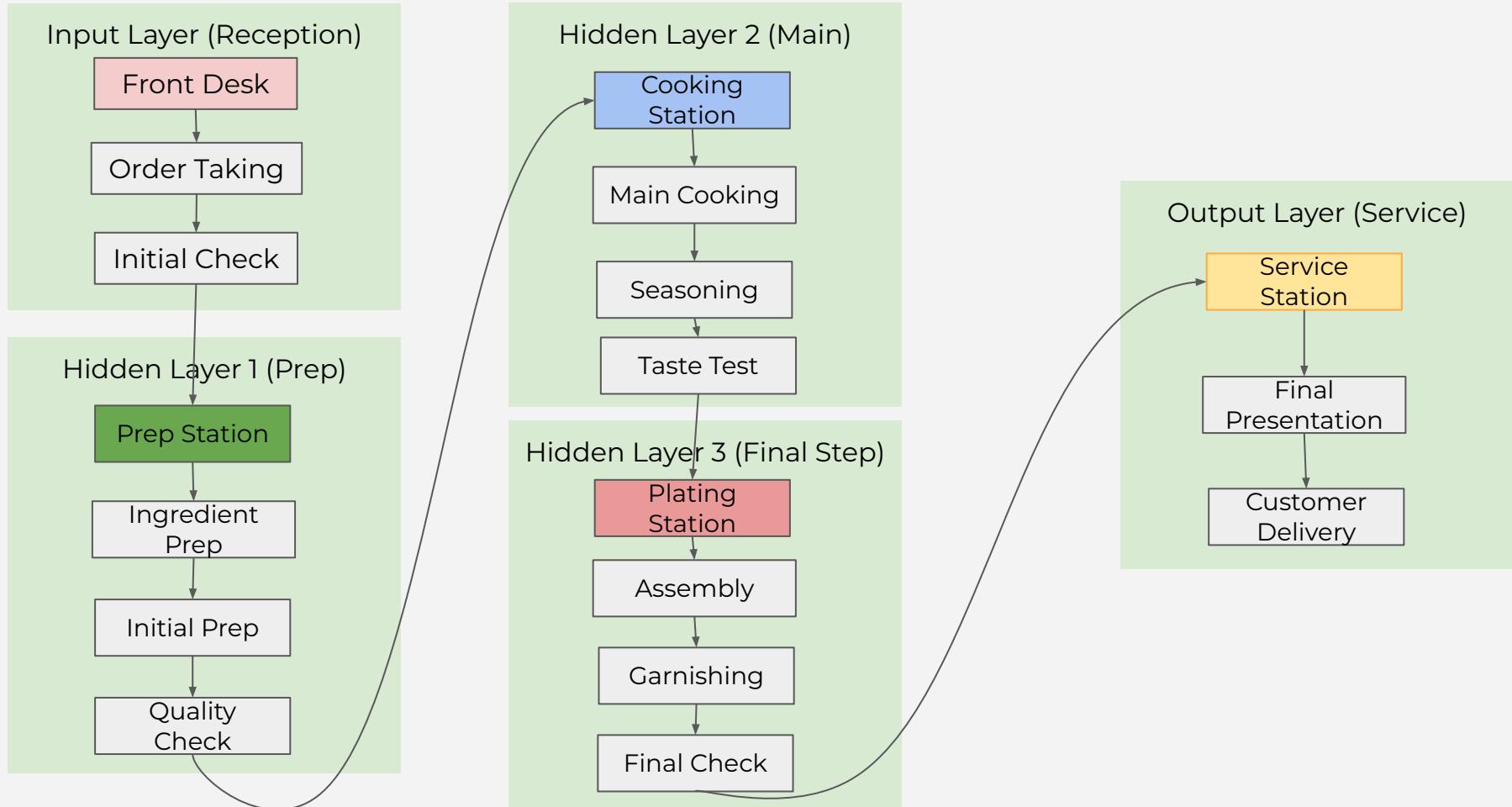
# The Neural Network - Deep Dive



# The Neural Network - Deep Dive



# Network Structure (Restaurant Layout)

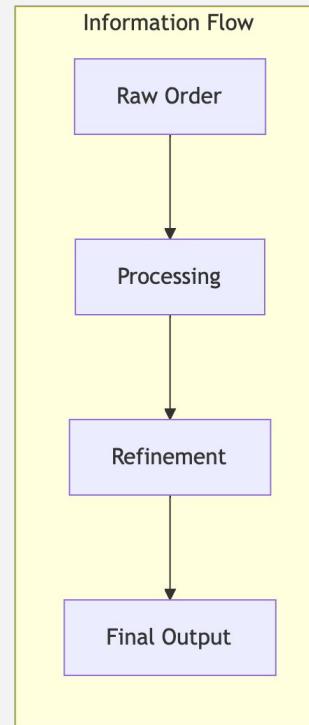


# Network Structure (Restaurant Layout)

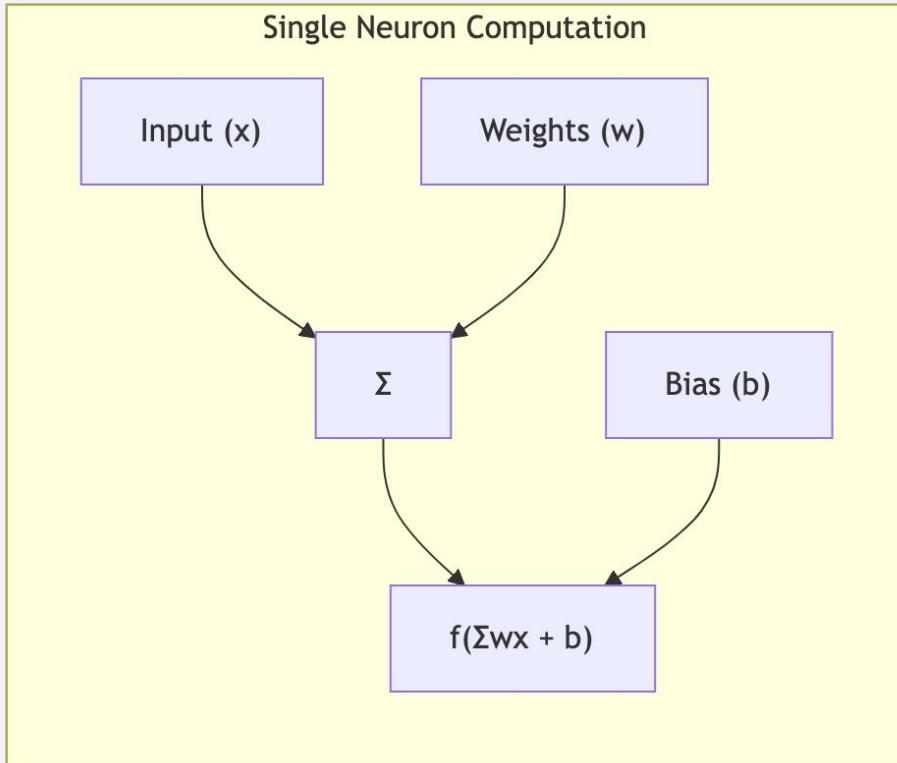
Key concepts:

- Bias = recipe proportions
- Weights = base seasonings
- Activation functions = quality controls

# Network Structure (Restaurant Layout)

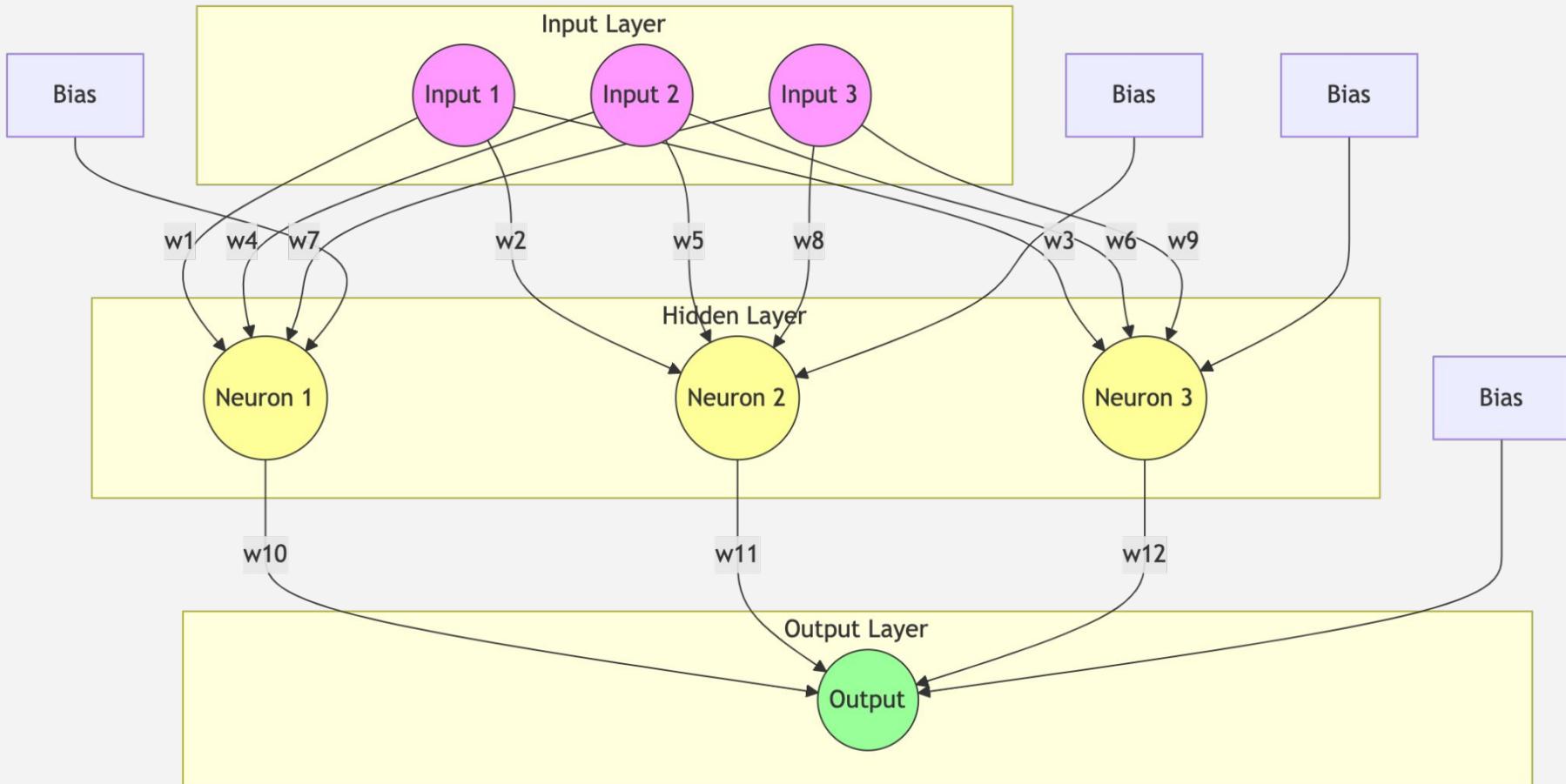


# Neural Network Mathematical Foundations: Weights & Biases

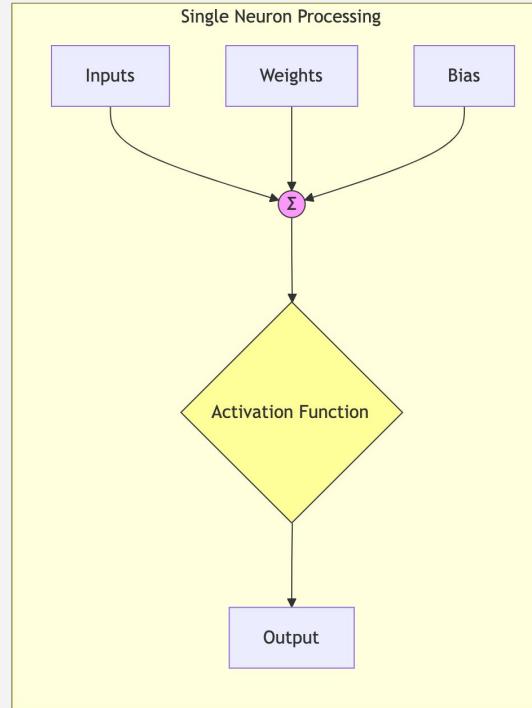
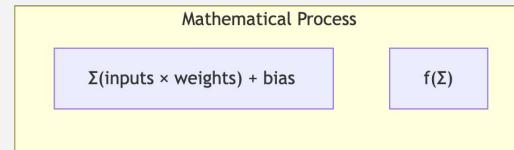


$$y = f\left(\sum_{i=1}^n w_i x_i + b\right)$$

# Neural Network Key Concepts (Restaurant Analogy)

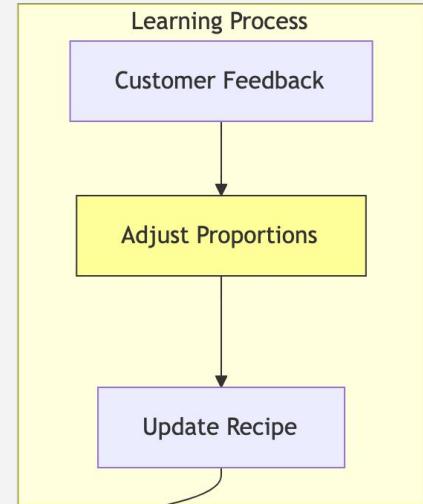
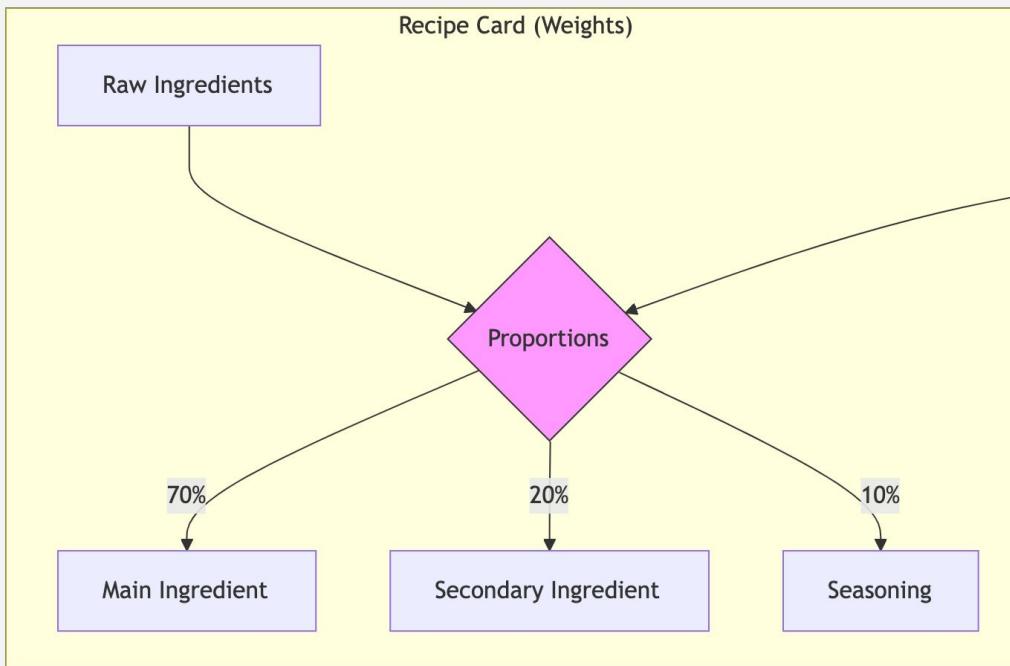


# Neural Network Key Concepts (Restaurant Analogy)

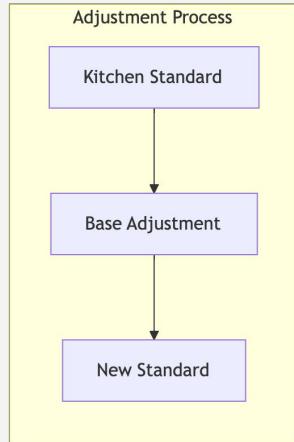


# Neural Network Key Concepts (Restaurant Analogy)

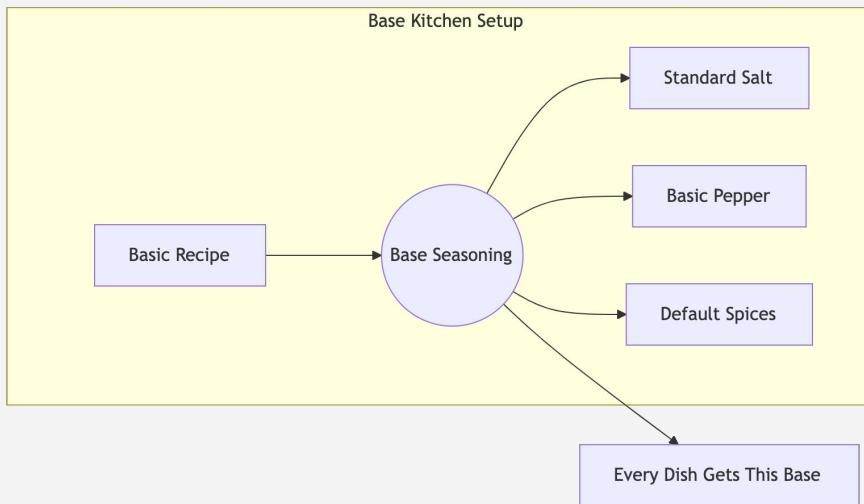
## Weights (Recipe Proportions)



# Neural Network Key Concepts (Restaurant Analogy)

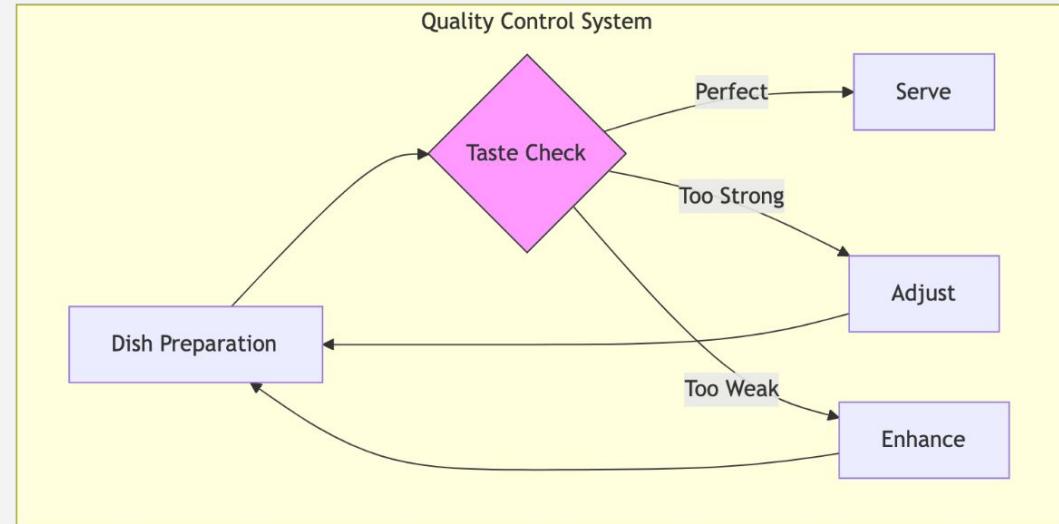
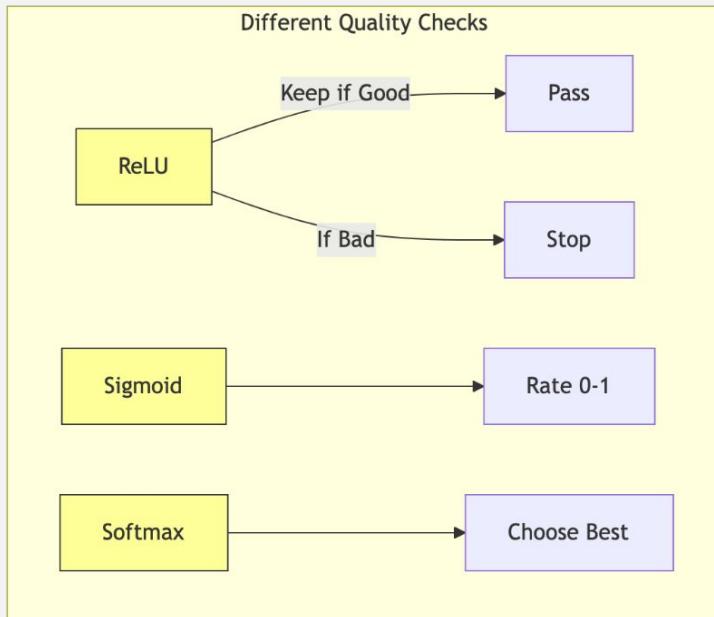


Weights (Recipe Proportions)

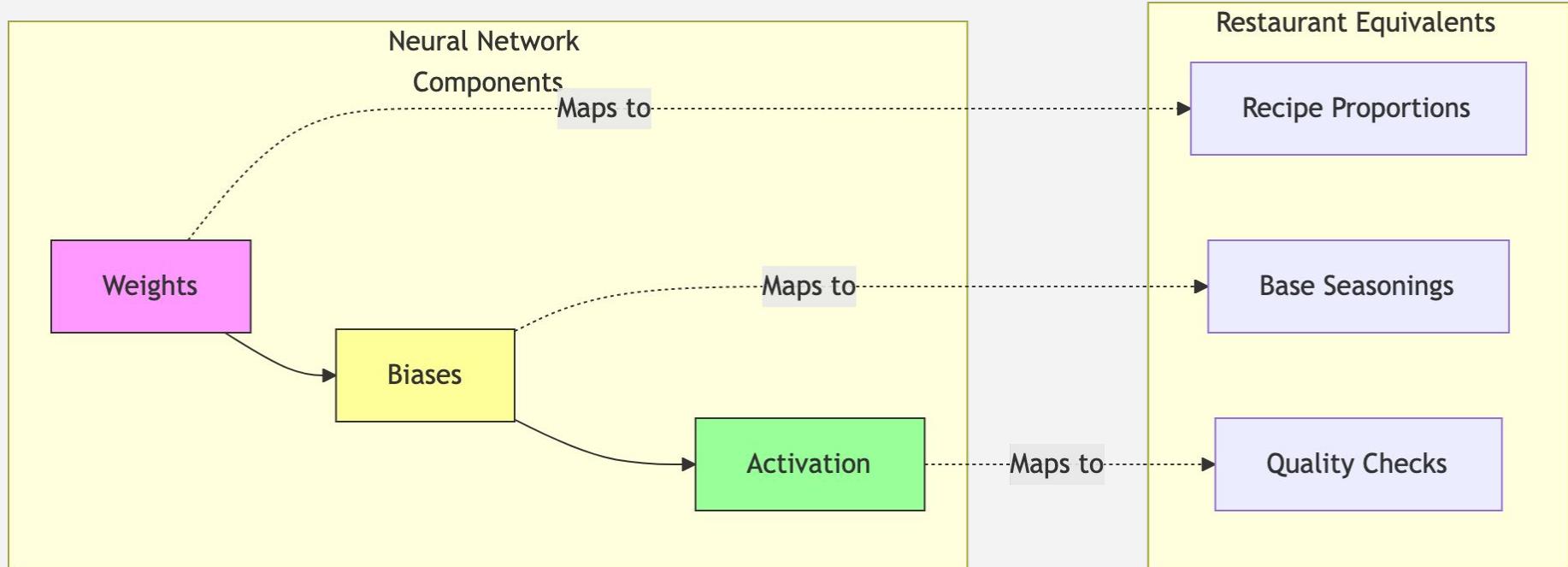


# Neural Network Key Concepts (Restaurant Analogy)

## Activation Functions (Quality Control)

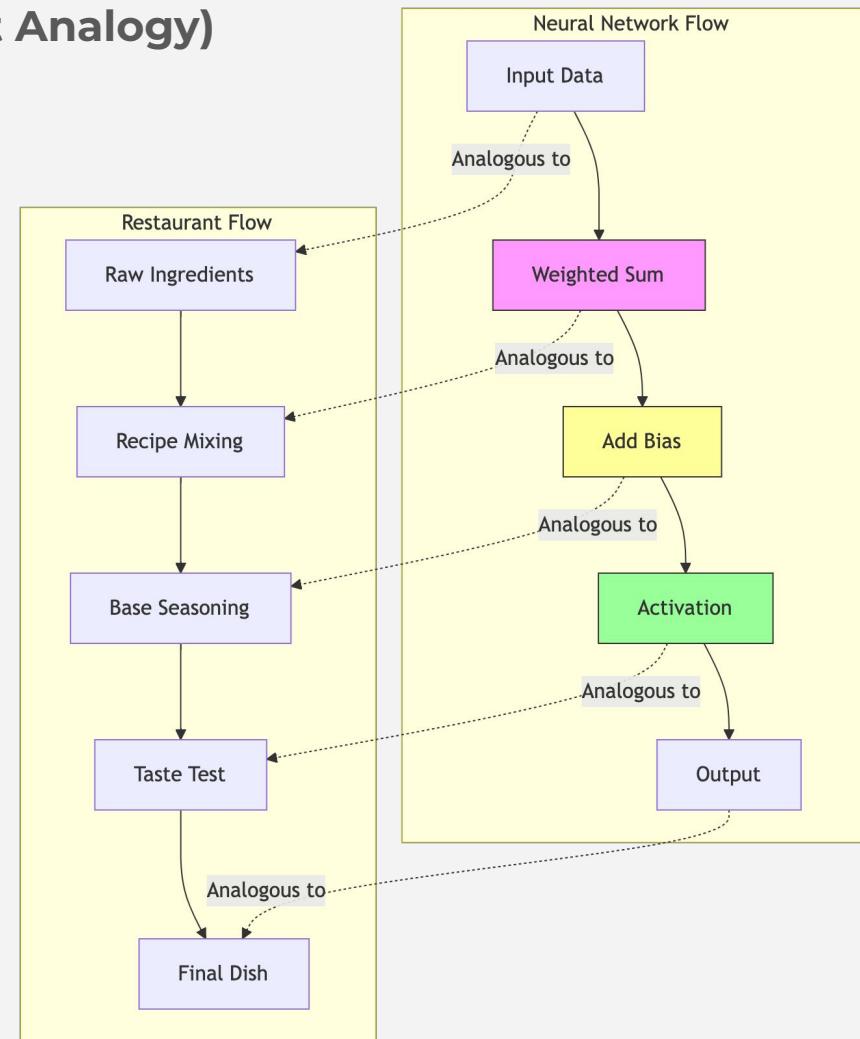


# Neural Network Key Concepts (Restaurant Analogy)



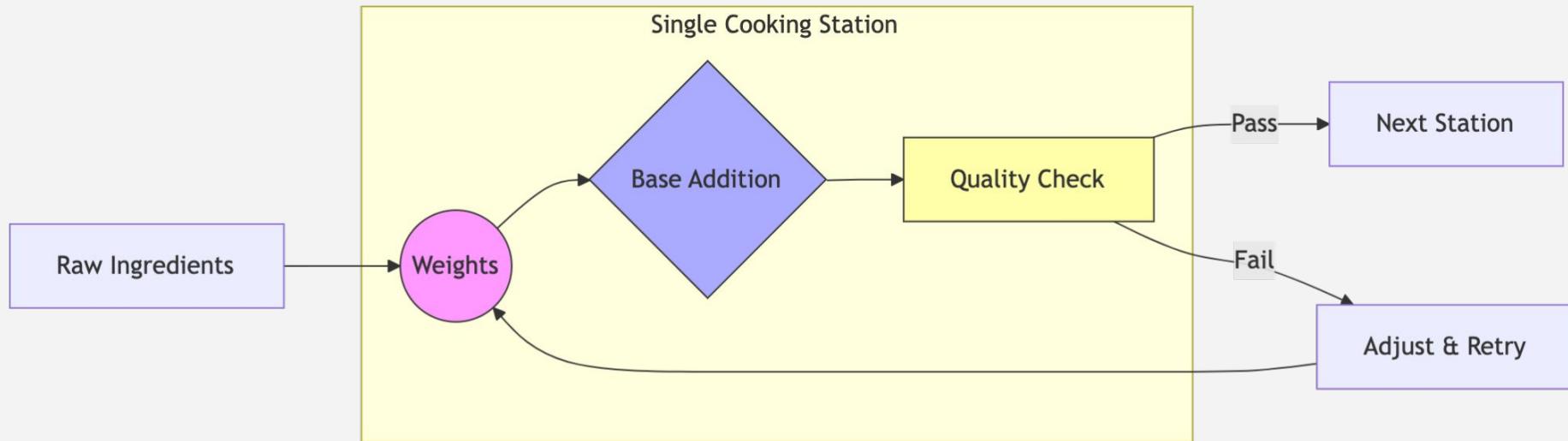
# Neural Network Key Concepts (Restaurant Analogy)

Information Flow Comparison



# Neural Network Key Concepts (Restaurant Analogy)

Real-world example

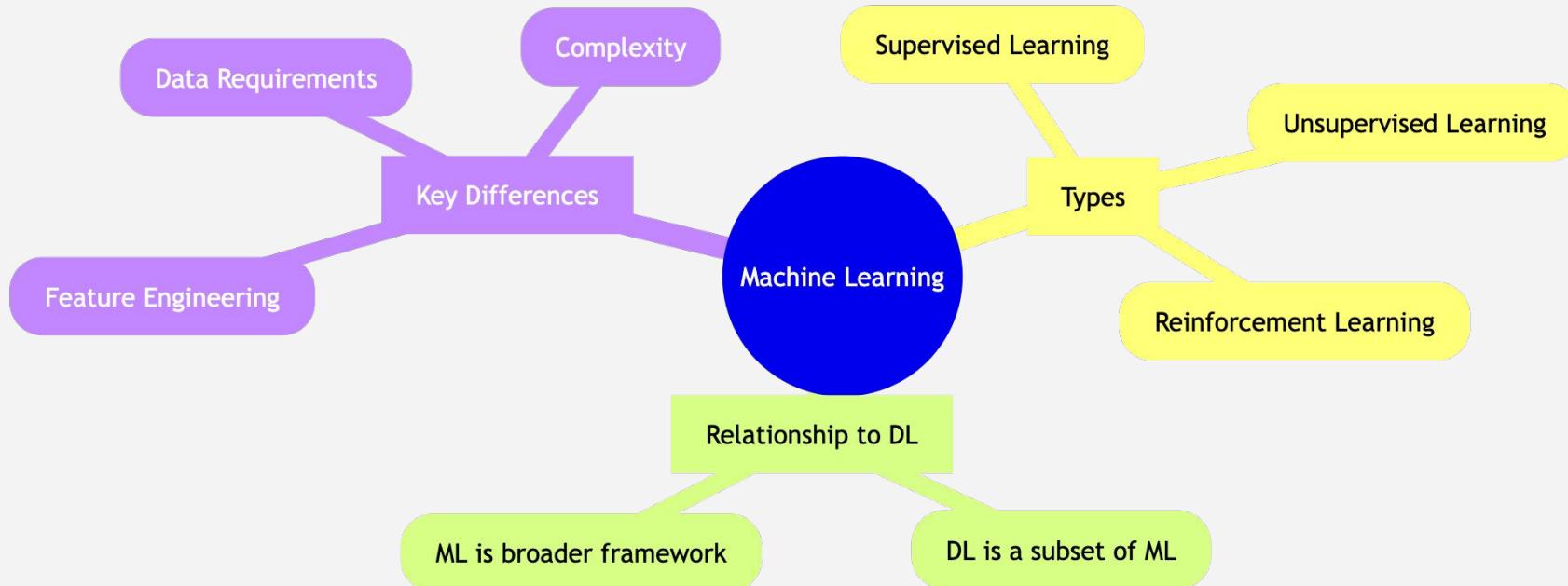


# Deep Learning - Summary

1. Deep Learning deep dive
  - a. Neural networks (input, hidden and output layers)
  - b. Bias and Weights, activation functions
2. Core ideas (DL)
  - a. Layer Structure
  - b. Weight/bias importance
  - c. Activation functions
  - d. Learning process



# Machine Learning Overview: Relationship with Deep Learning

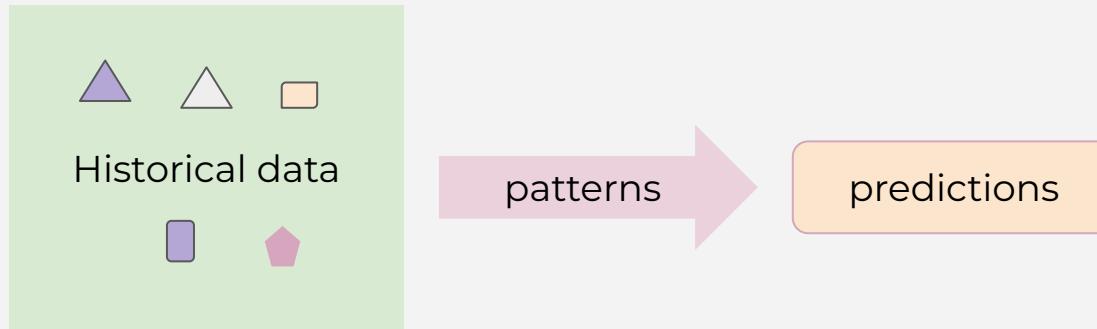




ML

- Algorithms that train models (systems). That data Improves over time.

Essentially, **Machine Learning** gives the computer the ability to learn without explicit programming



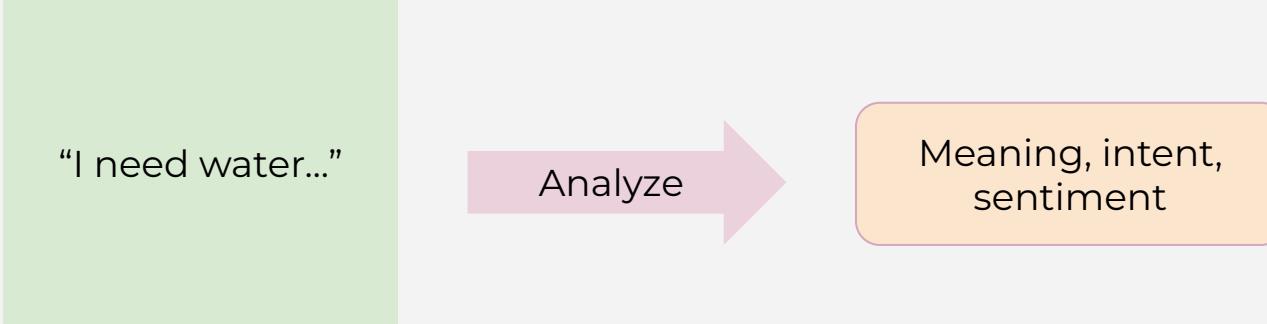
# Common categories of AI



NLP

- AI that understands and processes human language  
(Natural Language Processing)

Essentially, **Machine Learning** gives the computer the ability to learn without explicit programming



# ML vs DL

Deep Learning

Automatic Feature  
Learning

Large Data Sets

Black Box Model

Machine Learning

Manual Feature Engineering

Smaller Data Sets

Human Interpretable

# Learning Types (with Analogies)

Three Main Types

Supervised

Like Teacher & Student

Learning with Answers

Unsupervised

Like Puzzle Solving

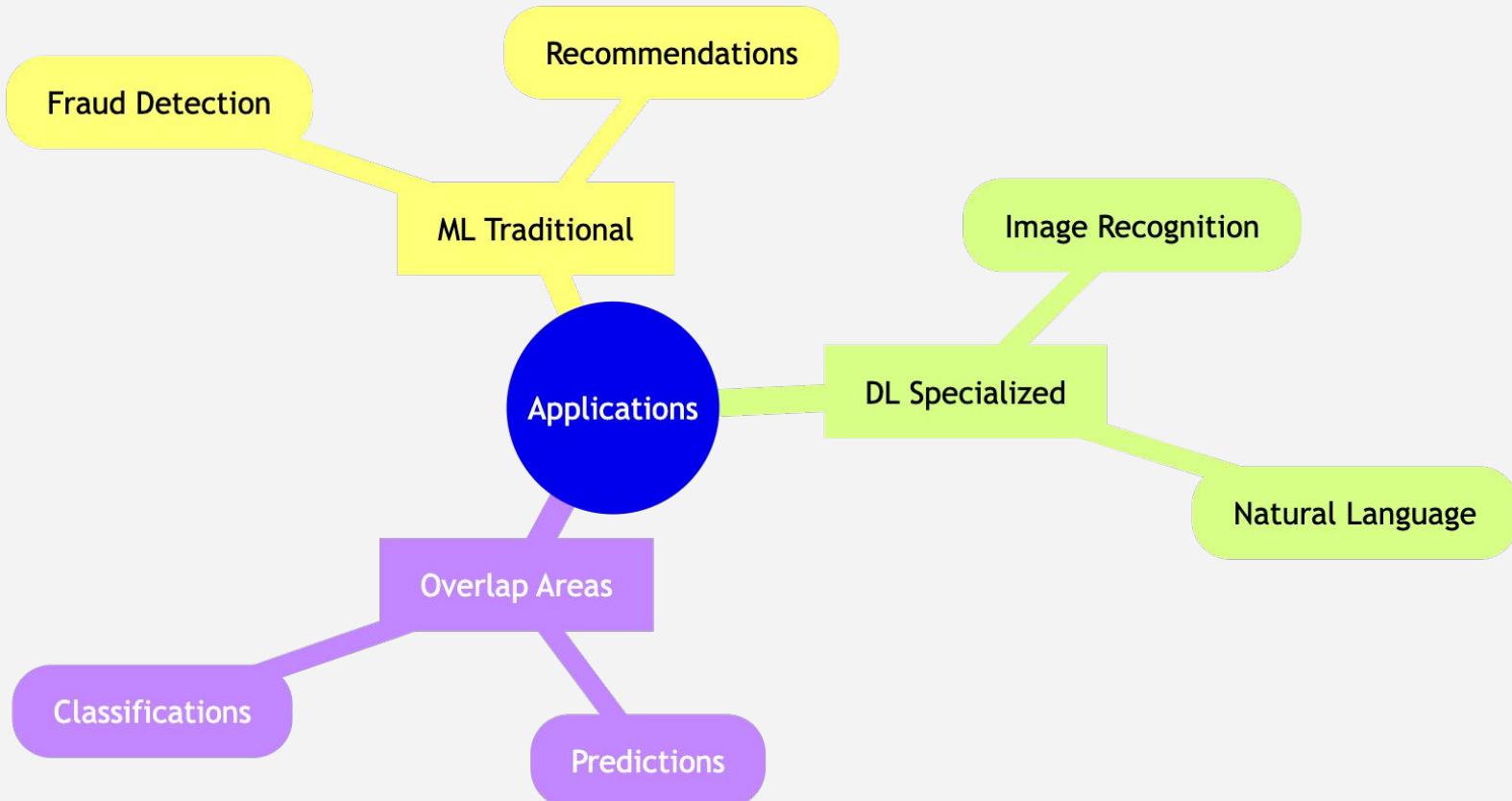
Finding Patterns

Reinforcement

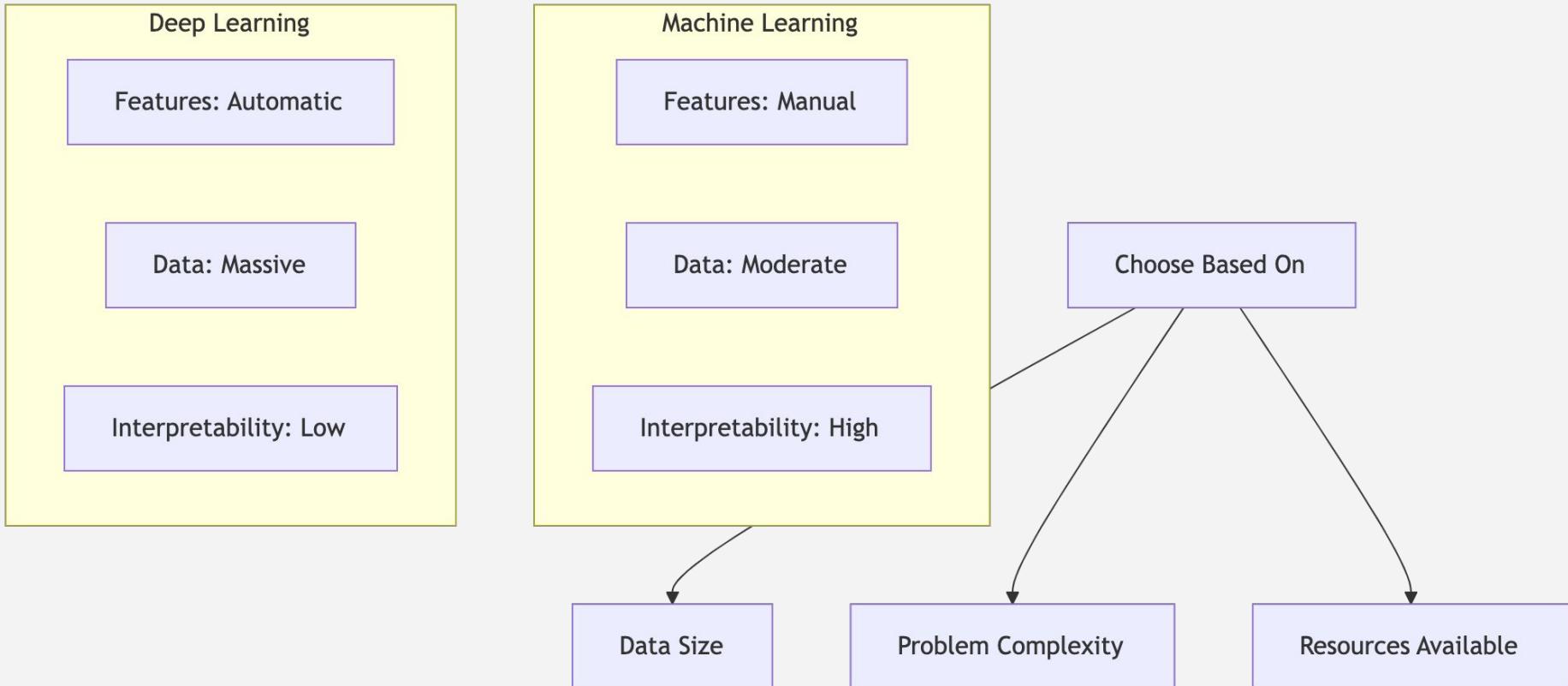
Like Training Dog

Learning from Experience

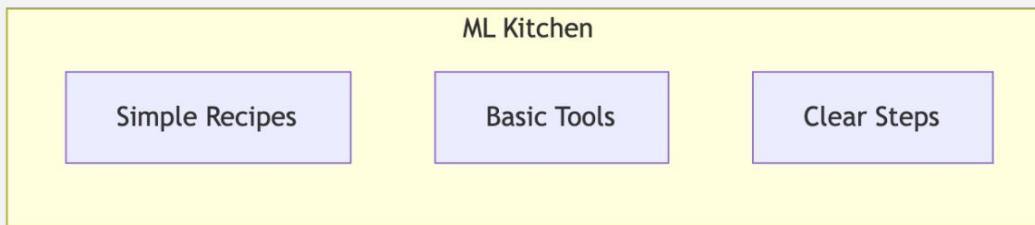
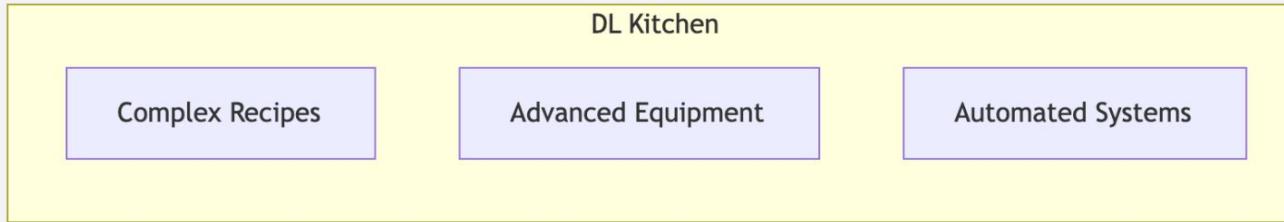
# Common Applications



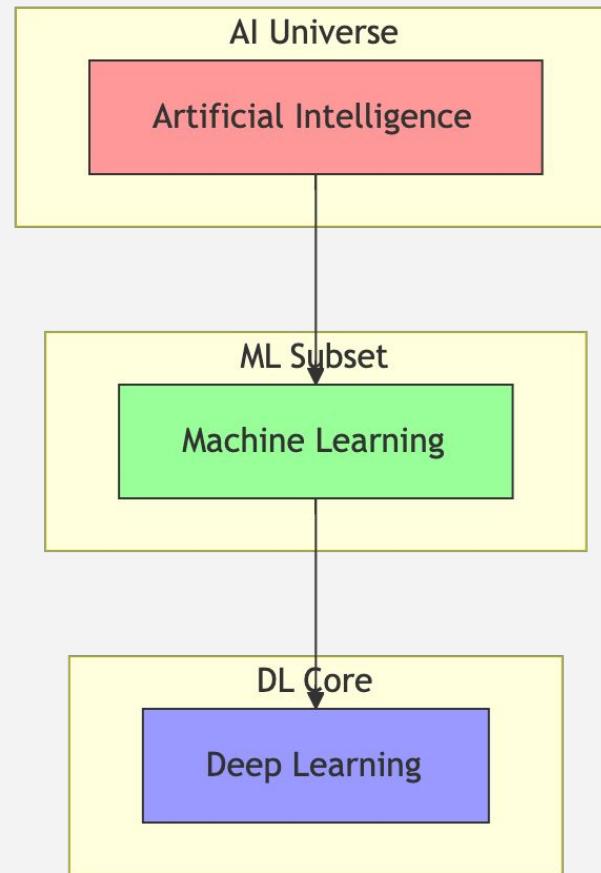
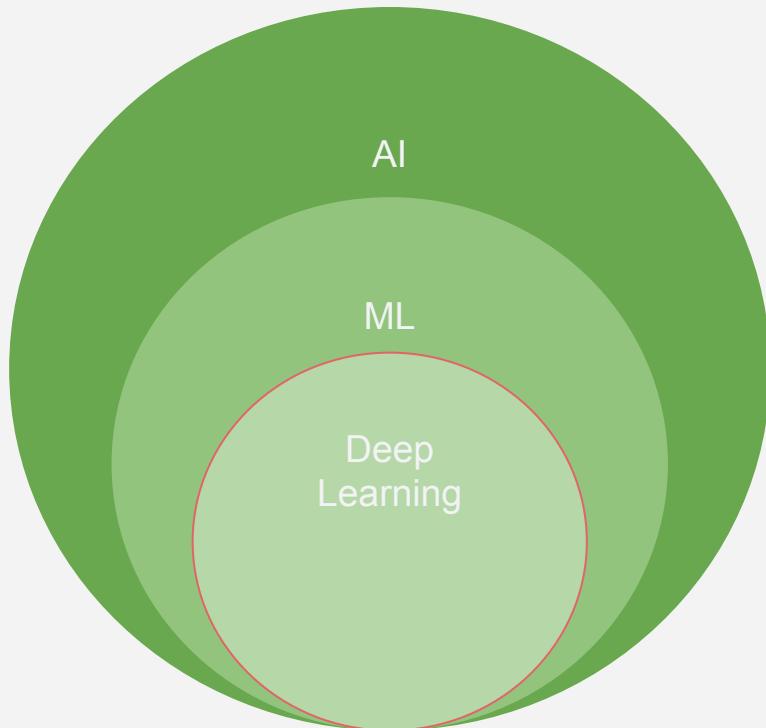
# Key Differences



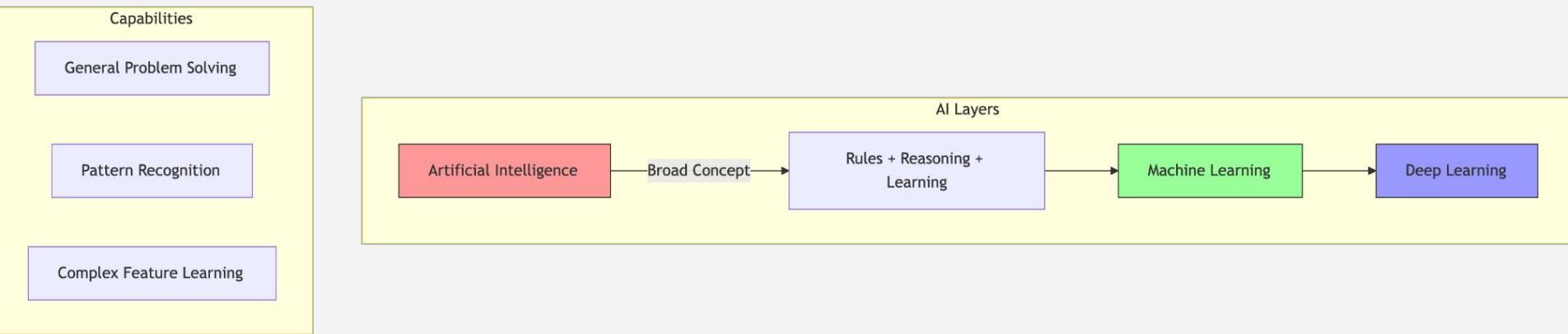
# Restaurant Analogy



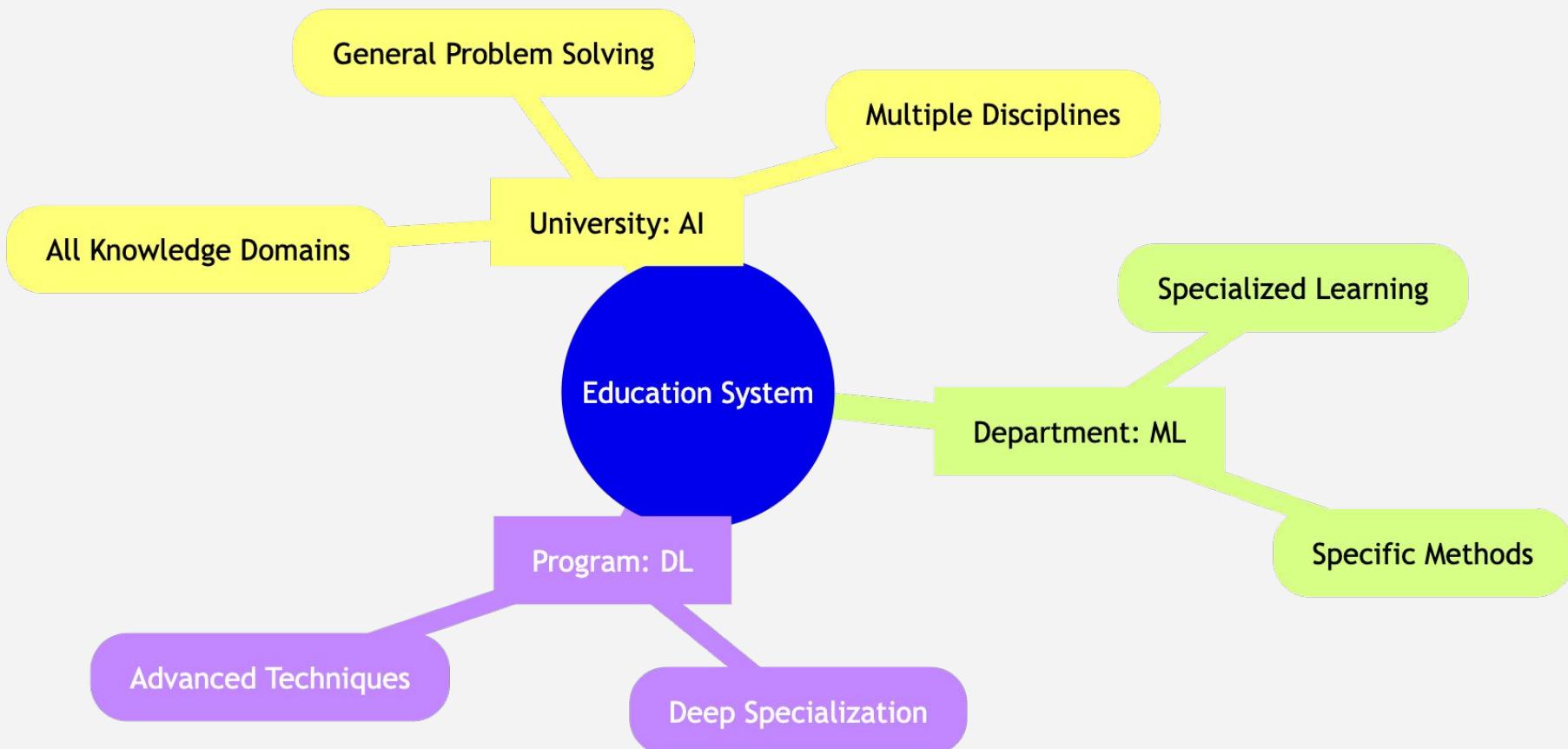
# The AI Hierarchy: From General AI to Deep Learning



# The AI Hierarchy: From General AI to Deep Learning

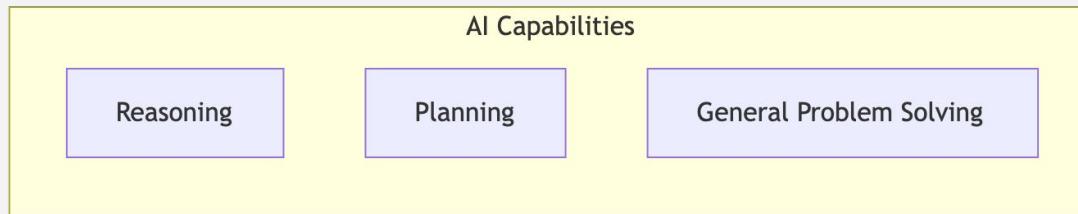
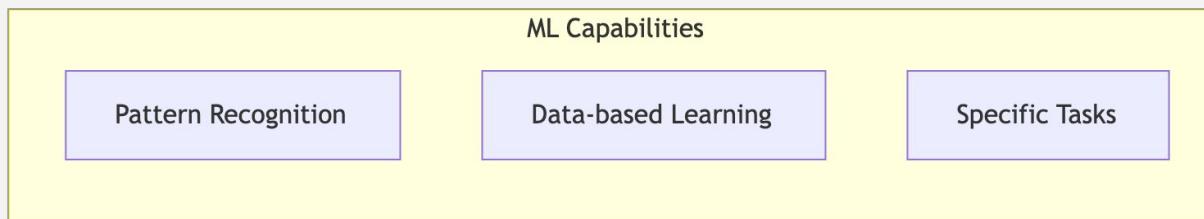
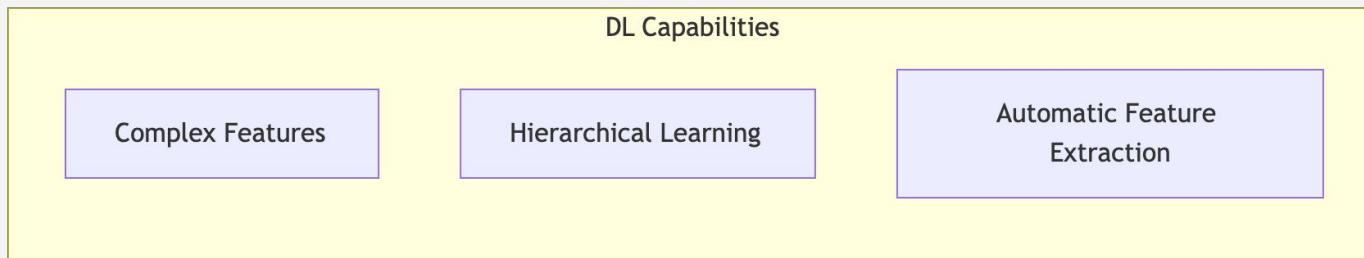


# The AI Hierarchy: From General AI to Deep Learning



# The AI Hierarchy: From General AI to Deep Learning

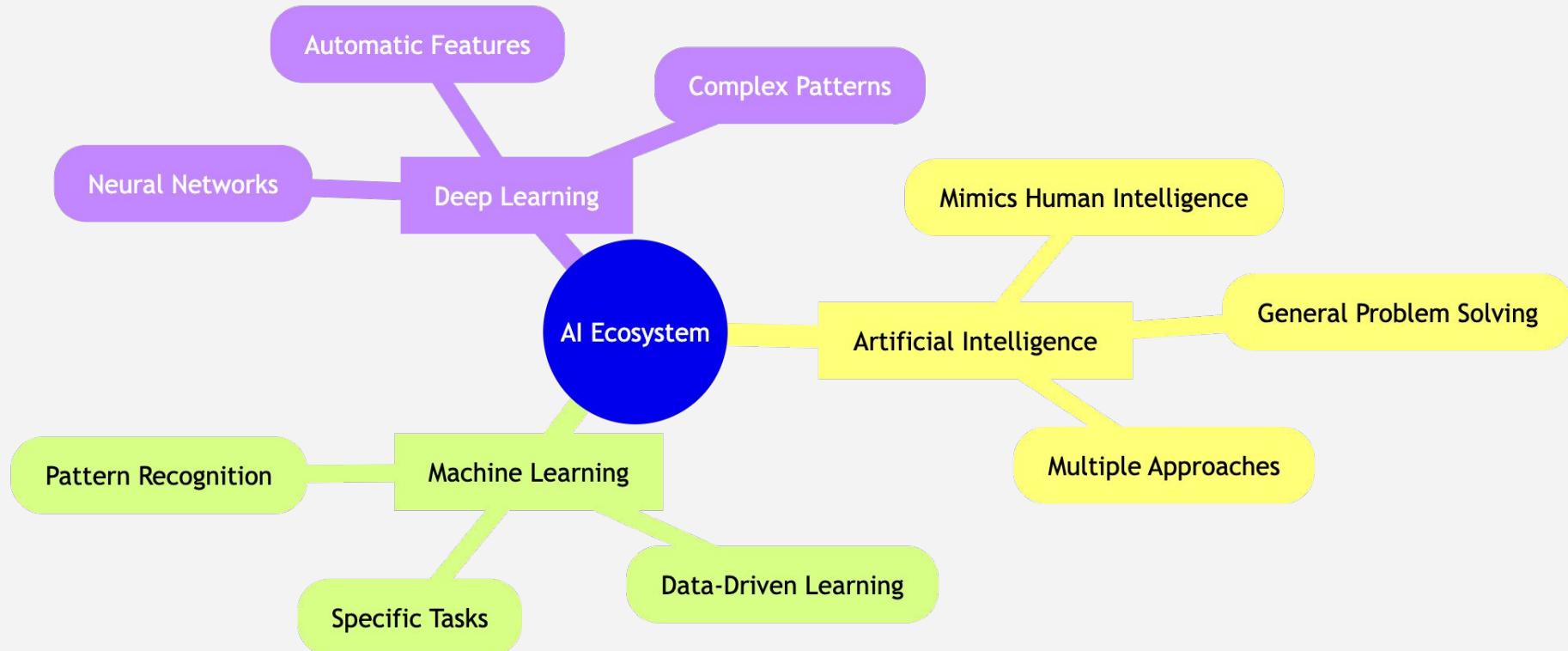
Comparative capabilities



# The AI Hierarchy: Kitchen Analogy



# The AI Hierarchy: Key Distinctions



# The AI Hierarchy: Application Spectrum

## DL Applications

Image Recognition

Natural Language

Speech Processing

## ML Applications

Predictions

Classifications

Recommendations

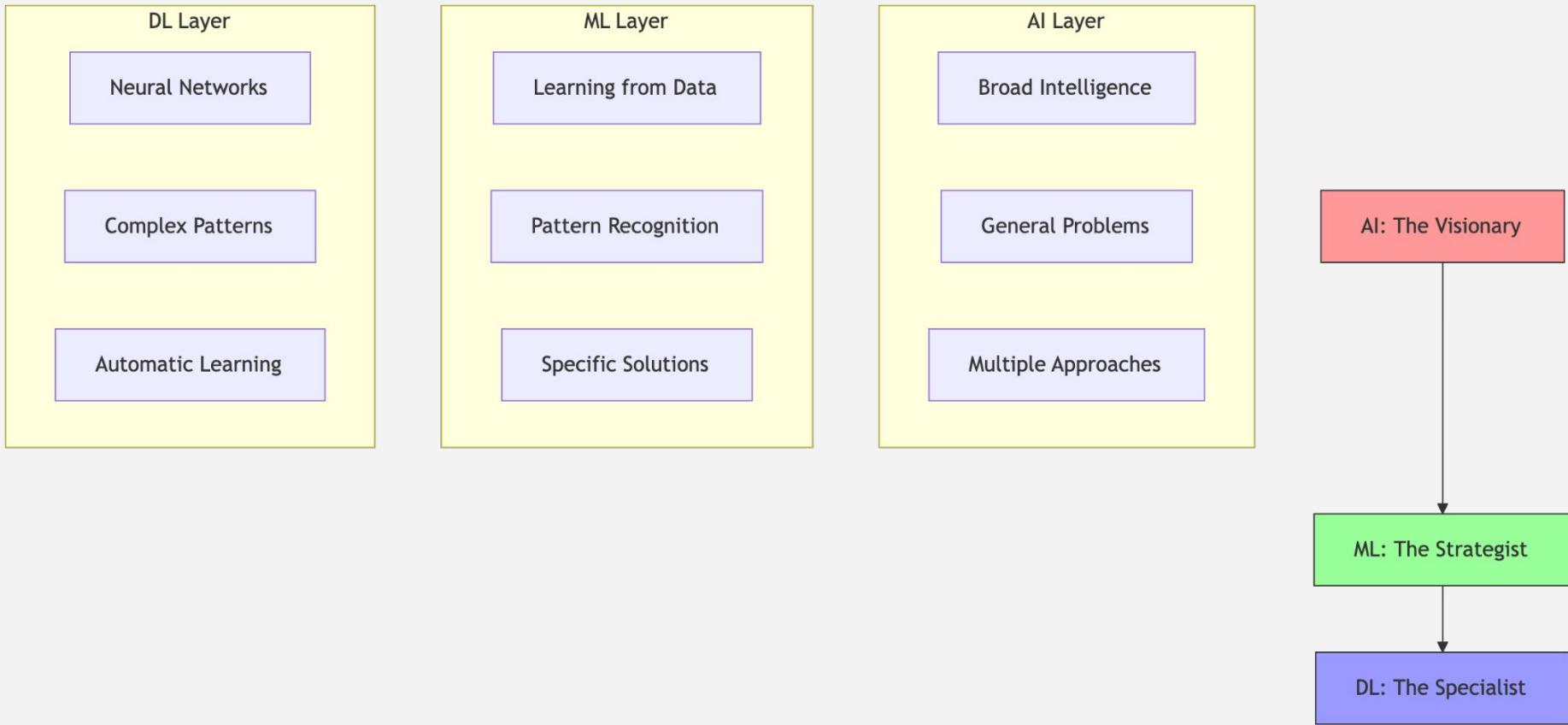
## AI Applications

Autonomous Systems

Robotics

Expert Systems

# The AI Hierarchy: Summary - Three-Layer Framework



# Generative AI

## GenAI

- Introduction
  - What is it?
  - How it works?
- Key concepts and Fundamentals

# What can I help with?

Brainstorm ideas|



Brainstorm ideas for my next vacation

Brainstorm ideas for a brand campaign

Brainstorm ideas for a new workout routine

Brainstorm ideas for a dinner party menu

Let's brainstorm ideas for my next vacation. Start by asking what time of year I want to travel.



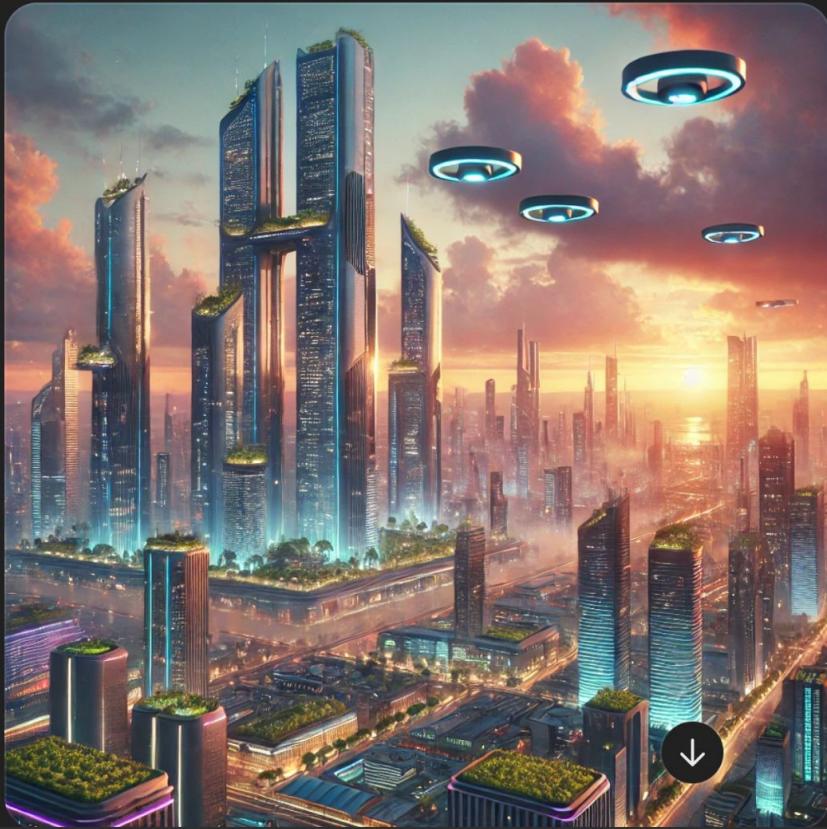
Sounds good! To kick things off:

**When do you want to travel?**

- Are you thinking of escaping winter's chill or enjoying a summer getaway?
- Maybe you have a specific holiday period in mind?
- Also, let me know if you prefer peak season buzz or quieter off-season vibes.

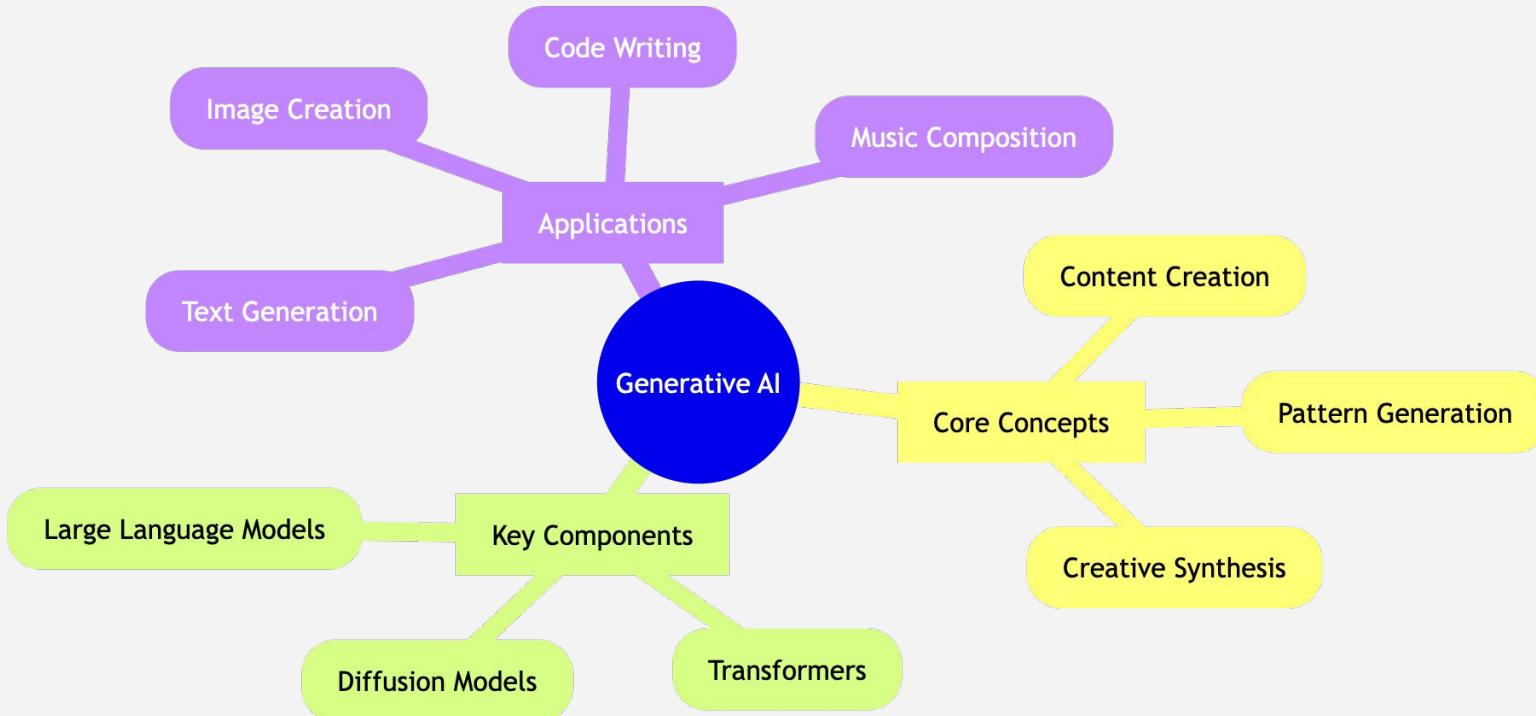
Let's get started! 😊

A futuristic city skyline at sunset

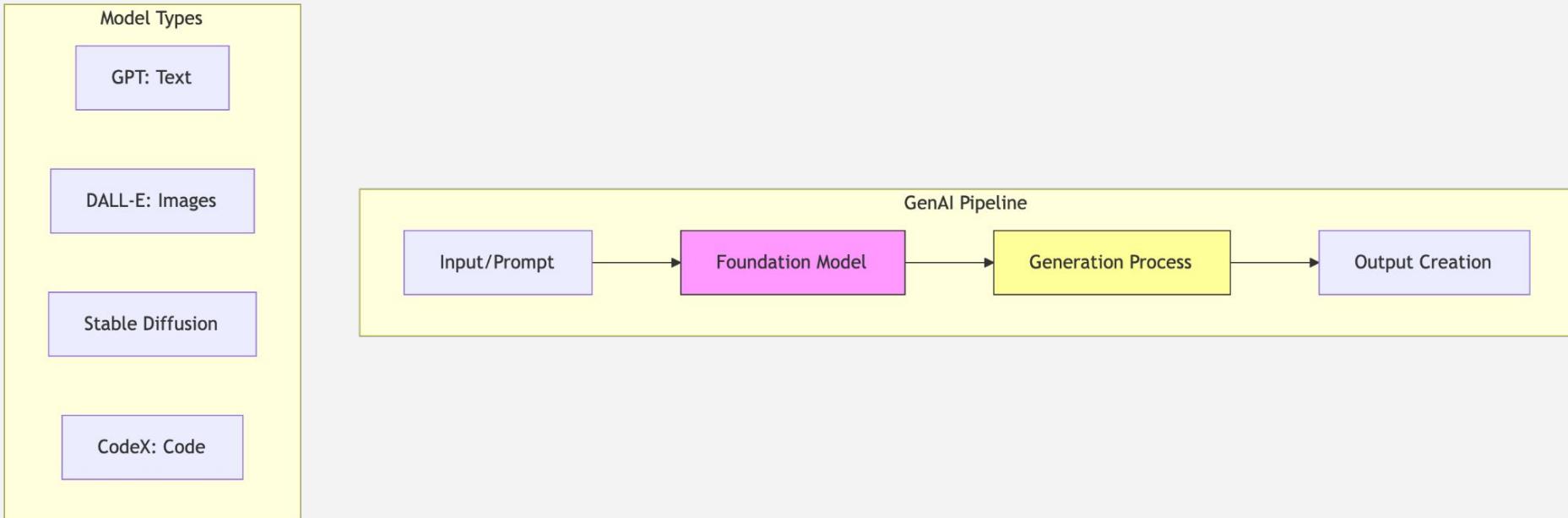


# Generative AI (GenAI) - Deep Dive

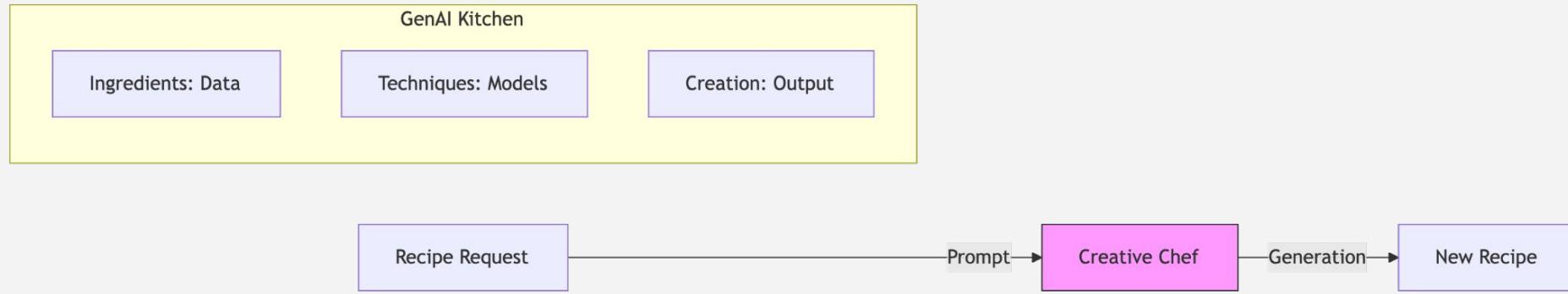
*“Artificial intelligence systems that can create new content (text, images, code, etc...)”*



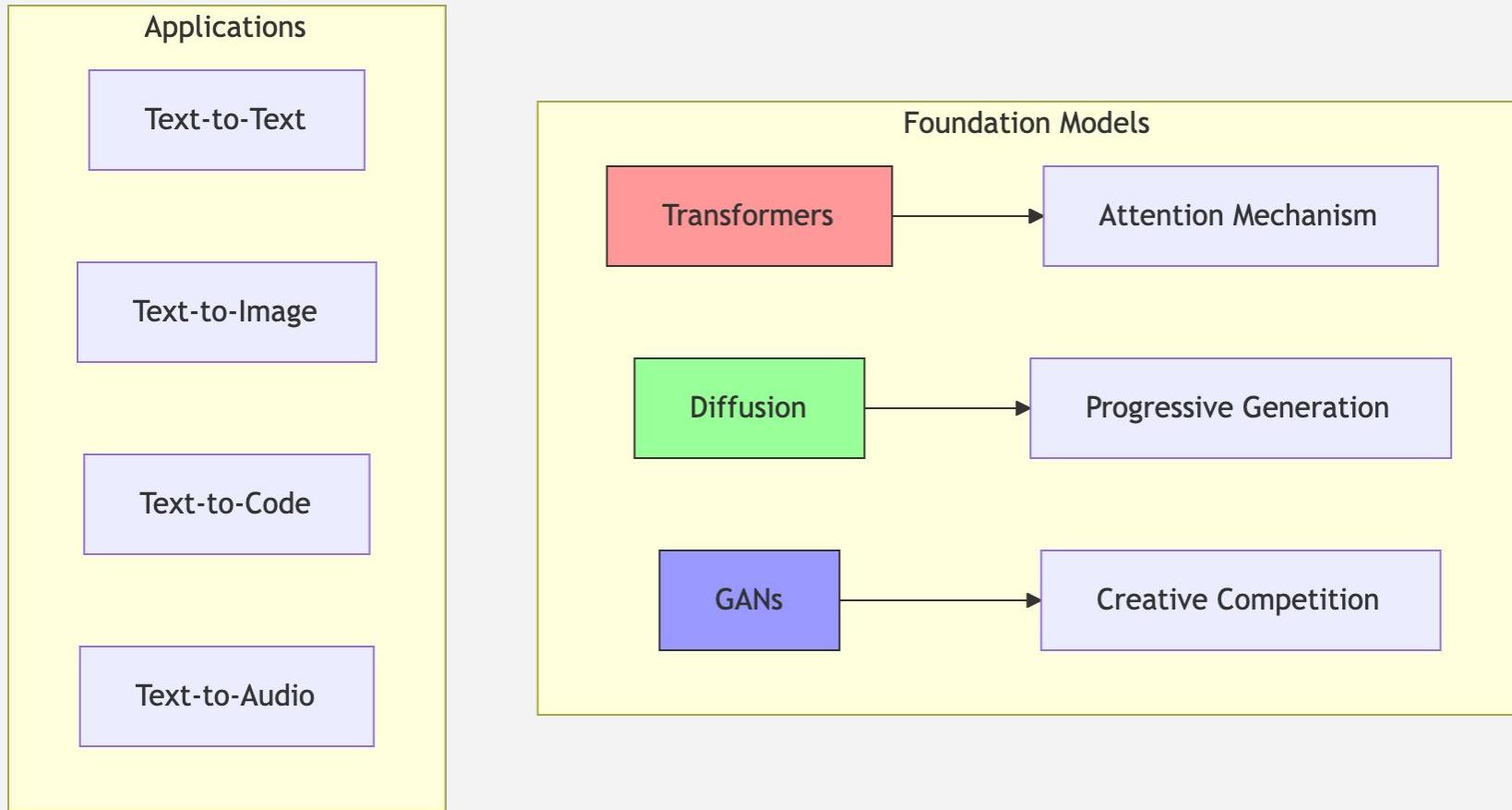
# Generative AI (GenAI) -Architecture Overview



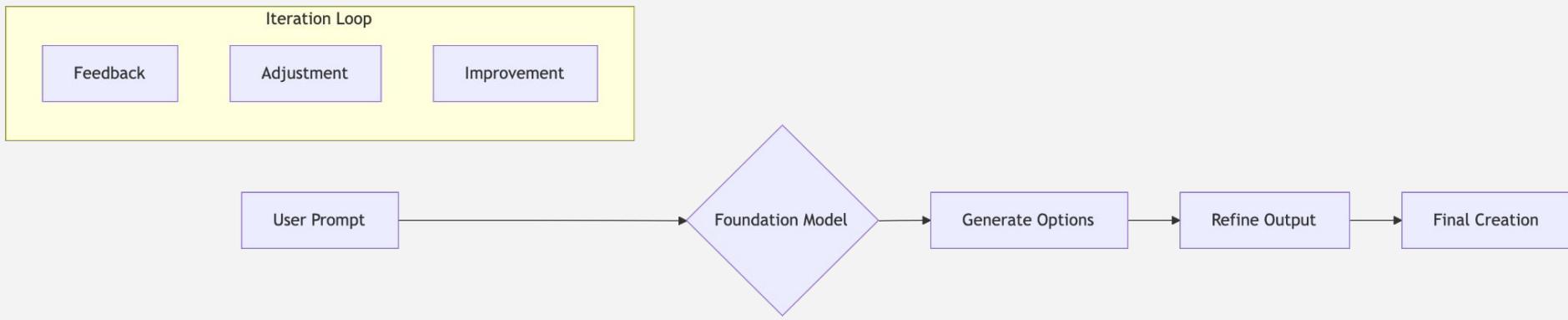
# Generative AI (GenAI) -Architecture Overview



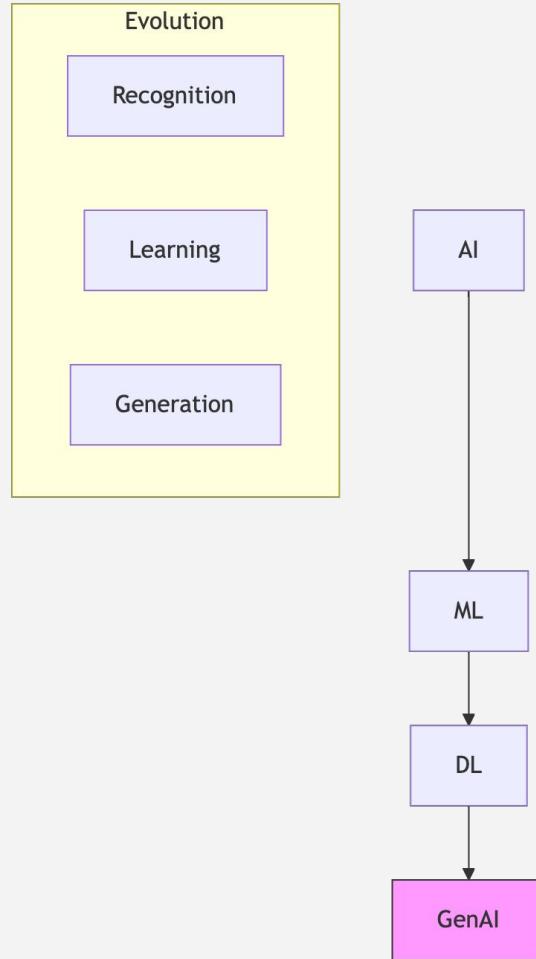
# Generative AI (GenAI) - Key Technologies



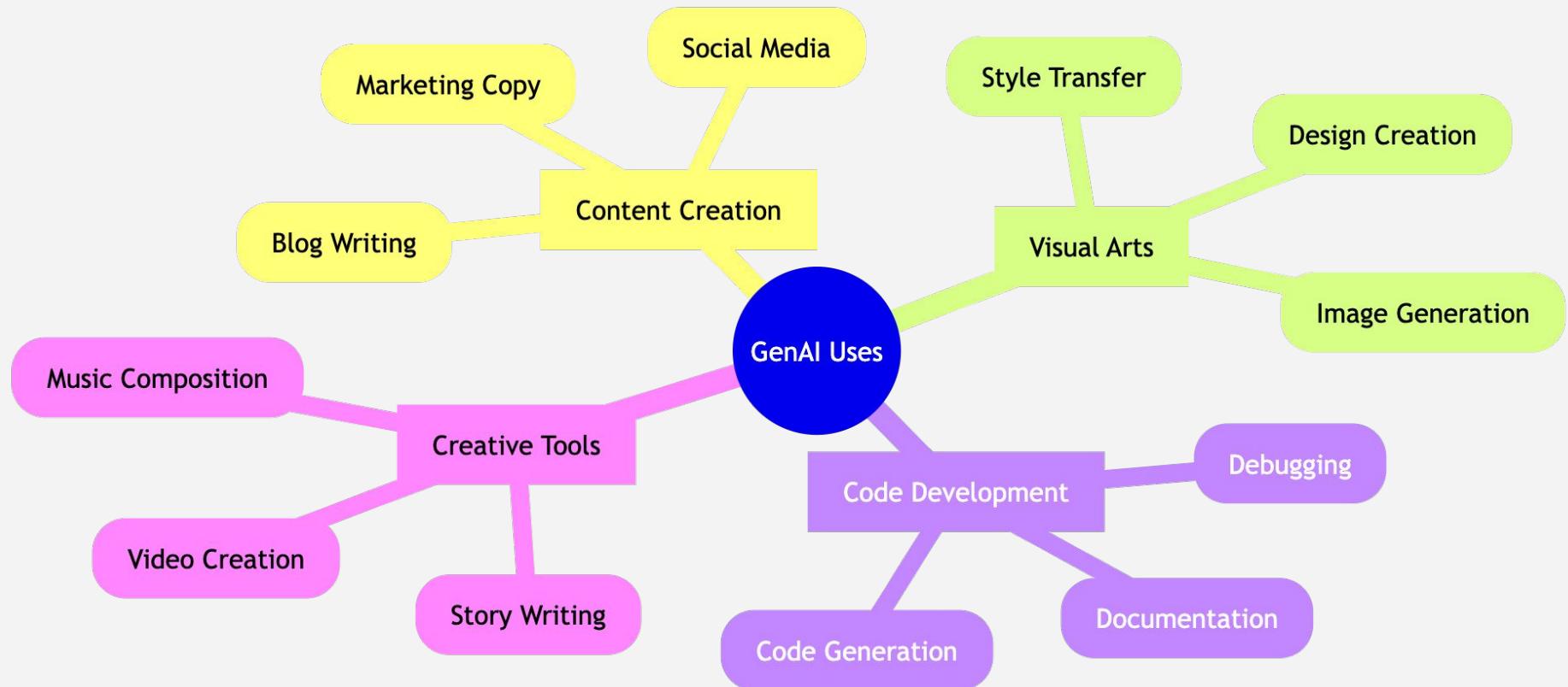
# Generative AI (GenAI) - The Creative Process



# Generative AI (GenAI) - GenAI in AI Ecosystem



# Generative AI (GenAI) - Practical Applications



# Generative AI (GenAI) - Unique Features

## Traditional AI

Pattern Recognition

Classification

Prediction

## GenAI Characteristics

Creative Ability

Pattern Understanding

Context Awareness

# Generative AI (GenAI) - Limitations & Challenges

## Solutions

Better Training

Human Oversight

Guidelines

## Challenges

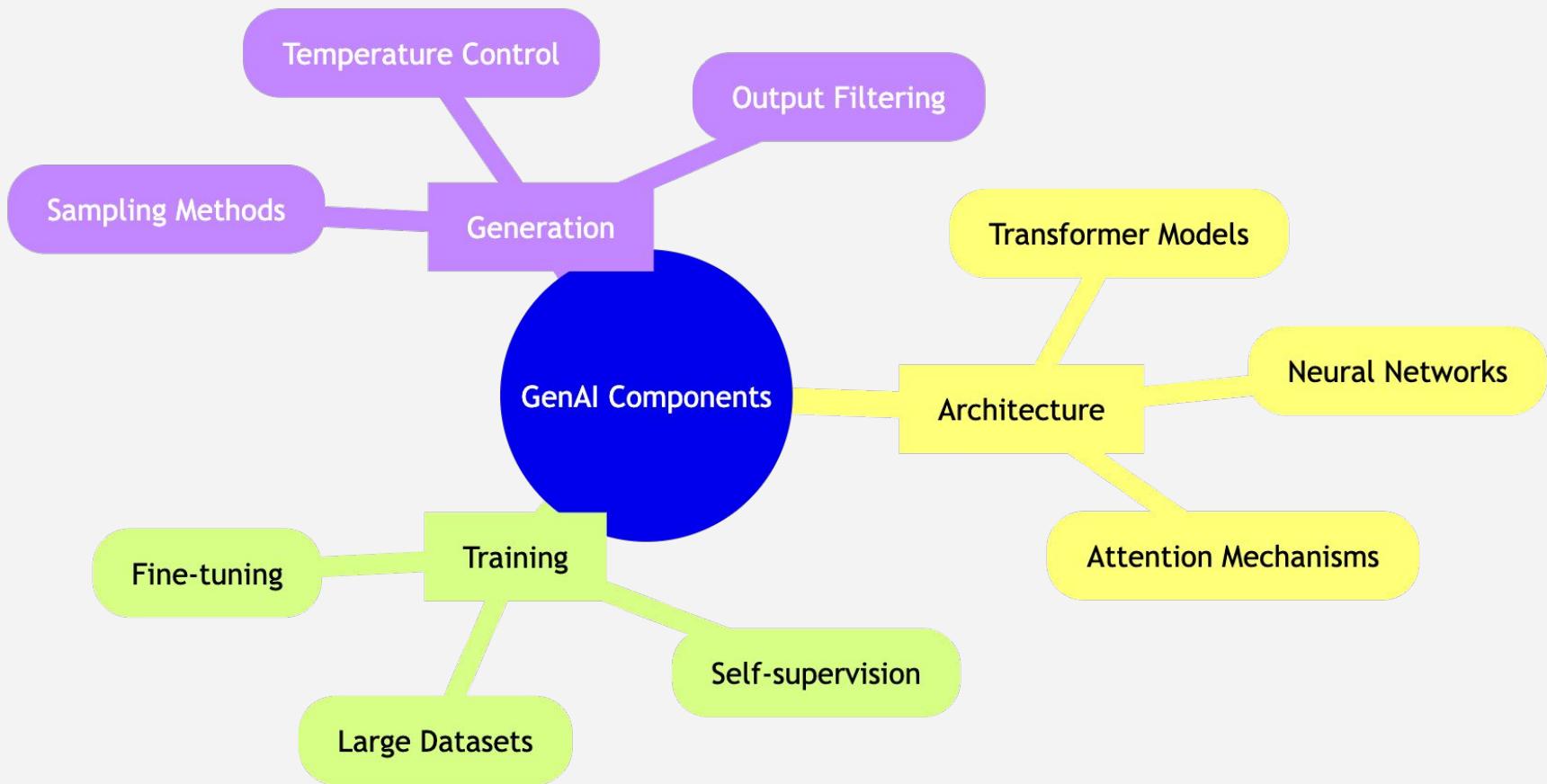
Hallucination

Bias

Control

Ethics

# Generative AI (GenAI) - Key Components



# Fundamentals of AI and LLMs

- Introduction to AI
  - What is it?
  - How it works?
- LLMs
  - Fundamentals
  - Main concepts
- Build a product with LLMs
  - Hands-on

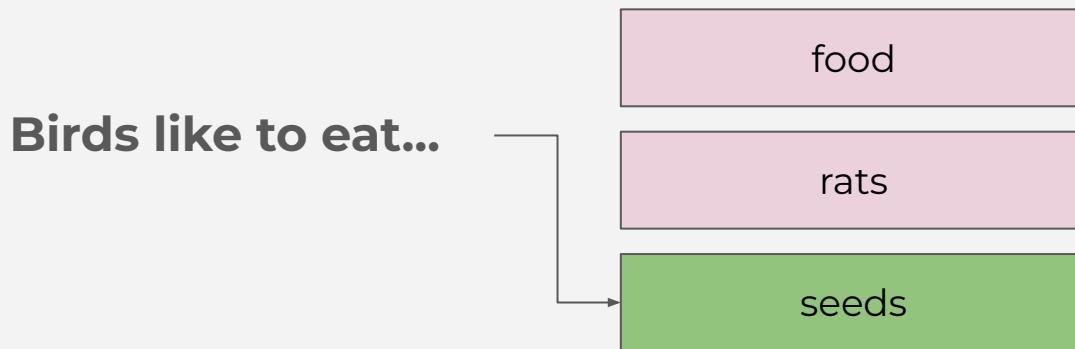
# LLMs

# Fundamentals

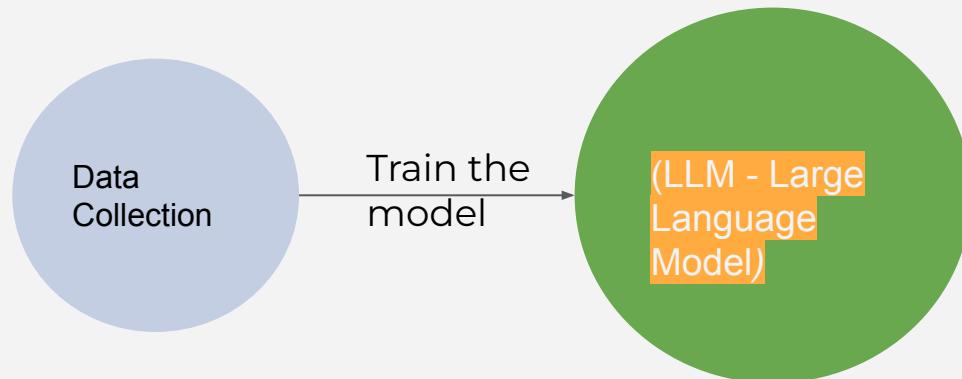
- What are LLMs
- How LLMs work
- Key concepts

# What are LLMs

Powerful models that generate **human-like text** by predicting the next word...

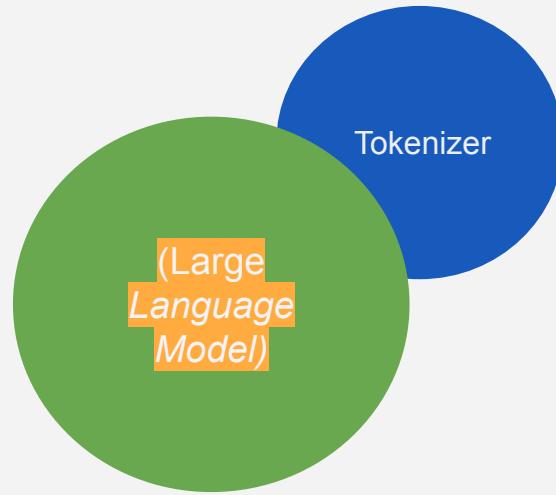


# Data Collection and Training...



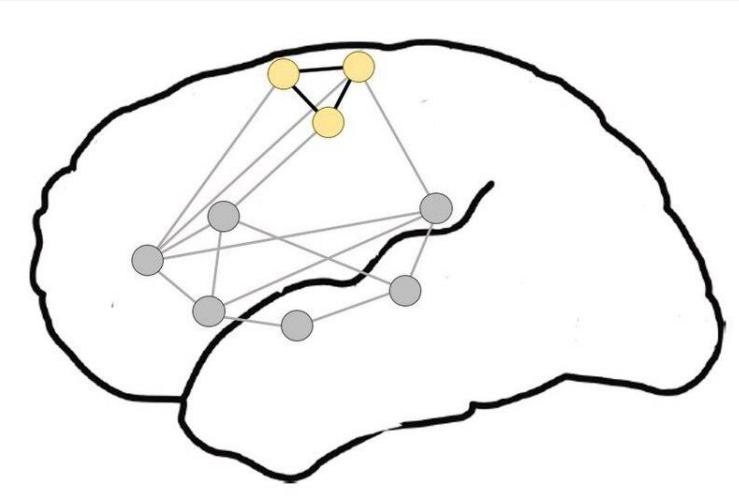
Collection of lots of data & train the model: learn patterns in human language (*articles, studies, news...*)

# Tokenizer



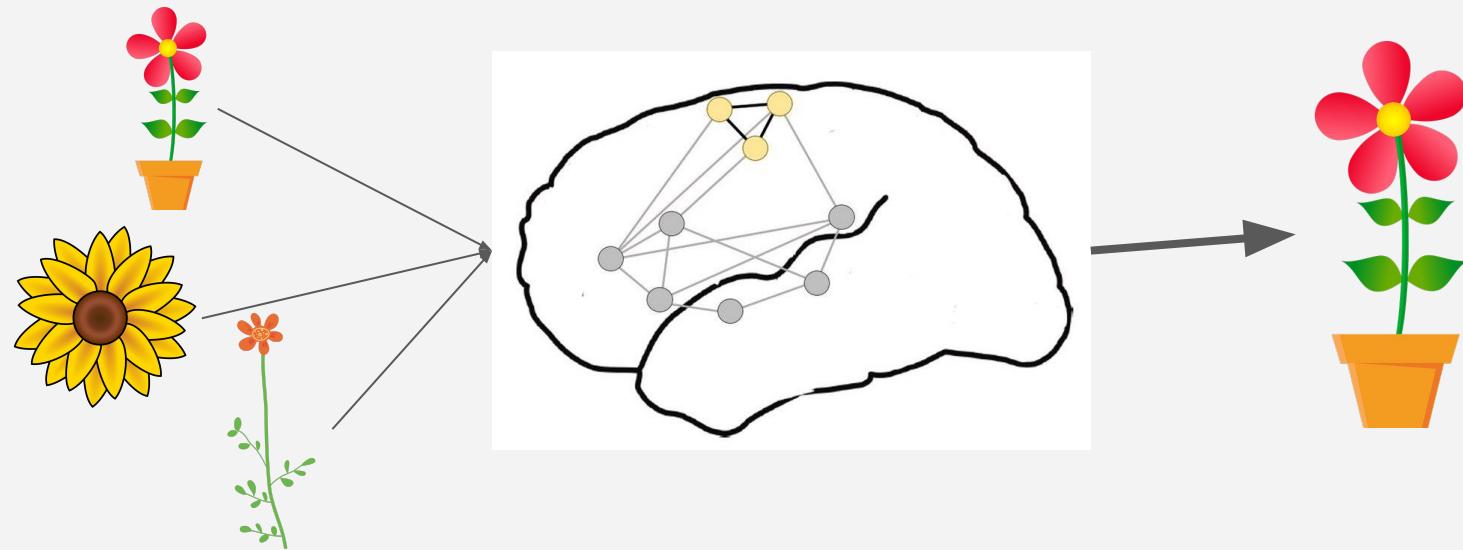
The **Tokenizer** splits the input text into smaller units - **tokens**: transforms the *input* into a format the model can understand.

# Neural Network - What is It?



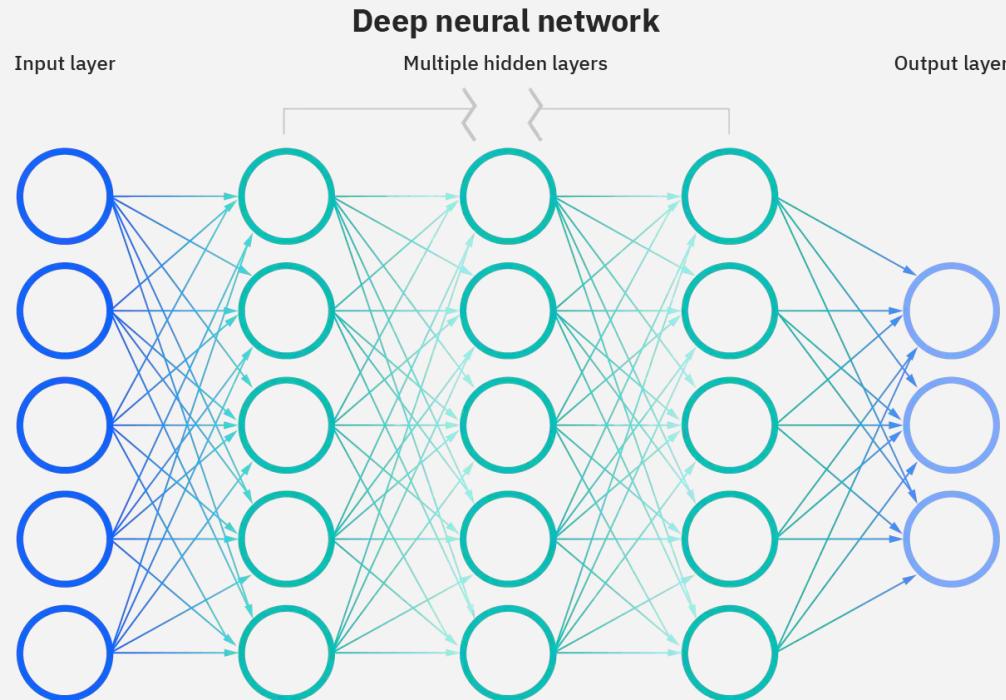
Our brain has a network of neurons which are interconnected - they process information - make sense of the world around us.

# Making Sense of the World...



For the brain to “know” what kind of flower/plant, it needs to be trained with lots of different flowers/plant types...

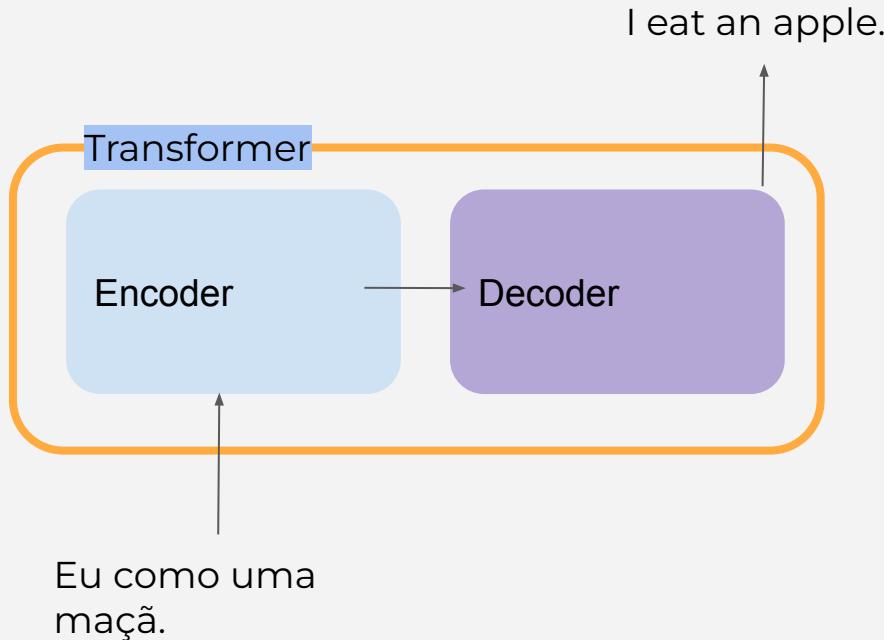
# Neural Network - AI



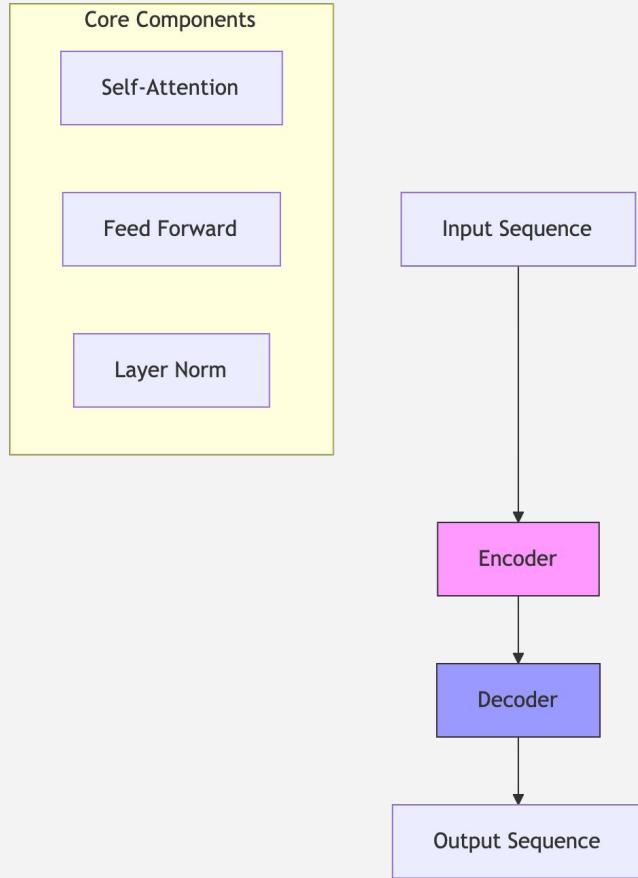
An AI Neural Network resembles the Brain Neural network!

# The Transformer Architecture - Overview

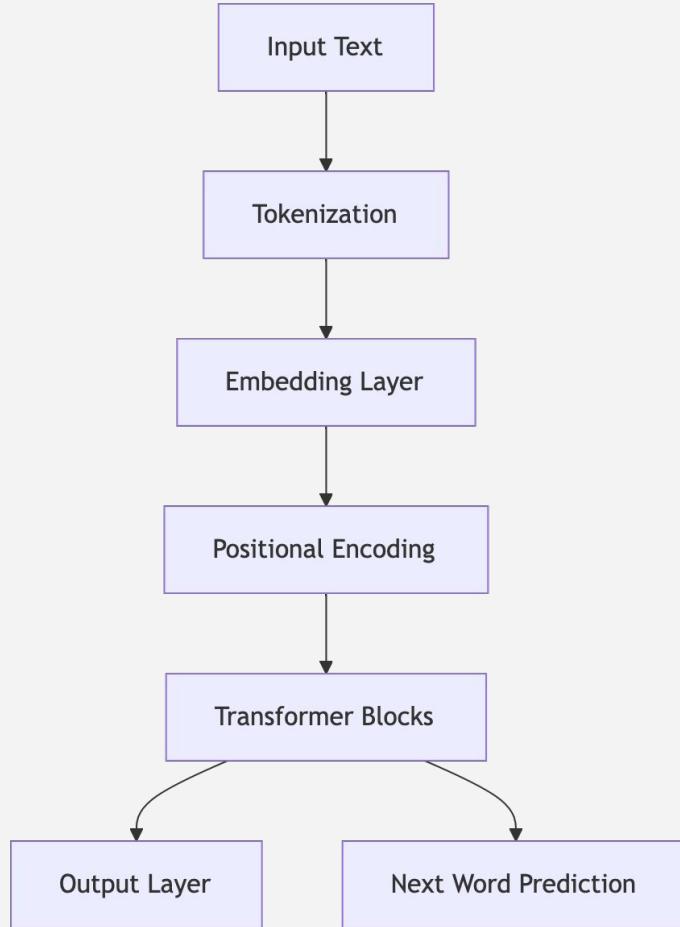
The **Transformer Architecture** is a *Neural Network* best suited for text and natural language processing.



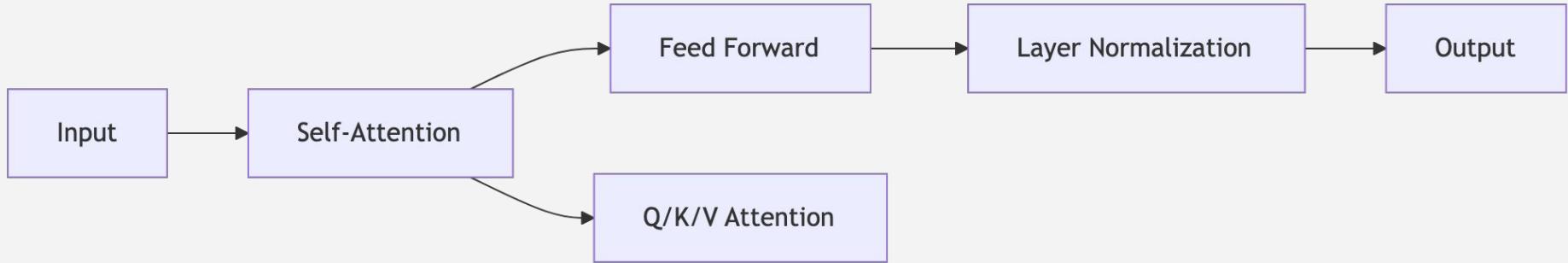
# The Transformer Architecture - Overview



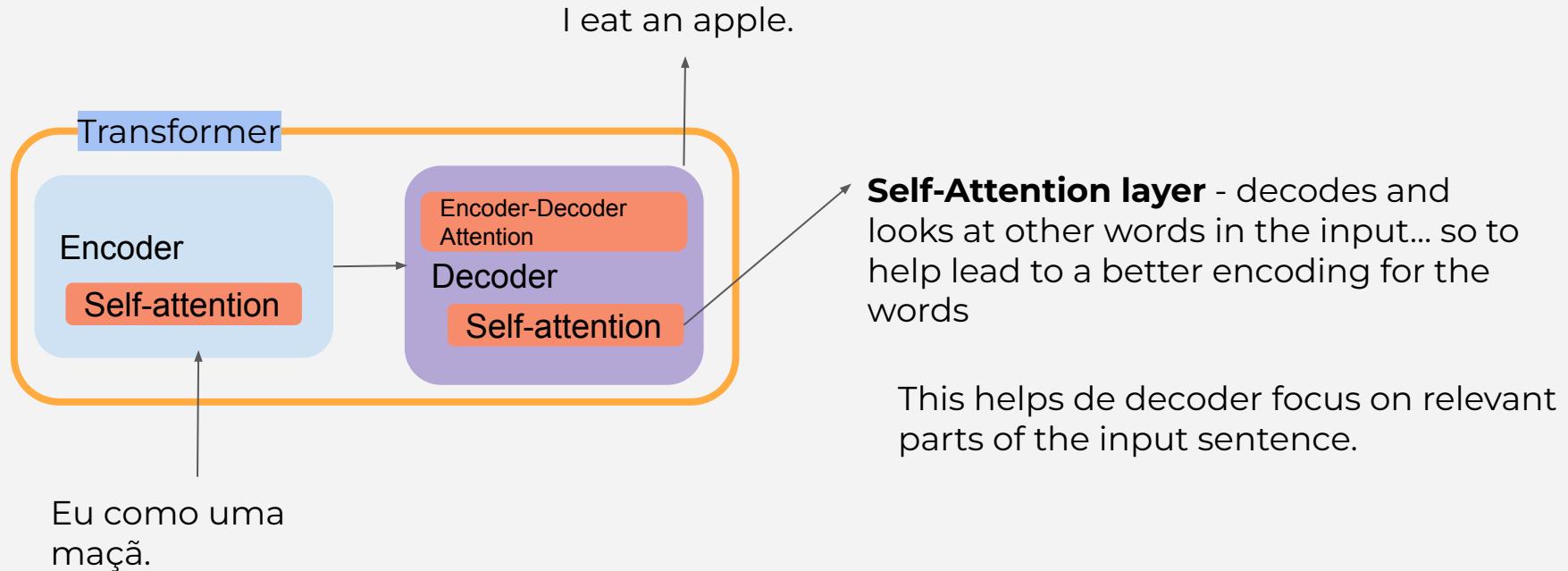
# The Transformer Architecture - Overview



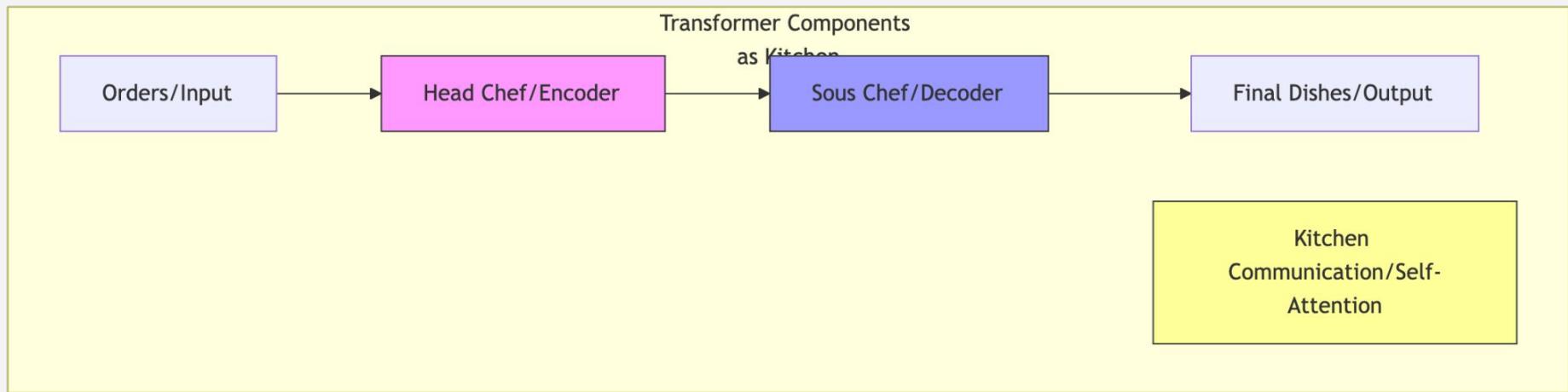
# The Transformer Architecture - Overview



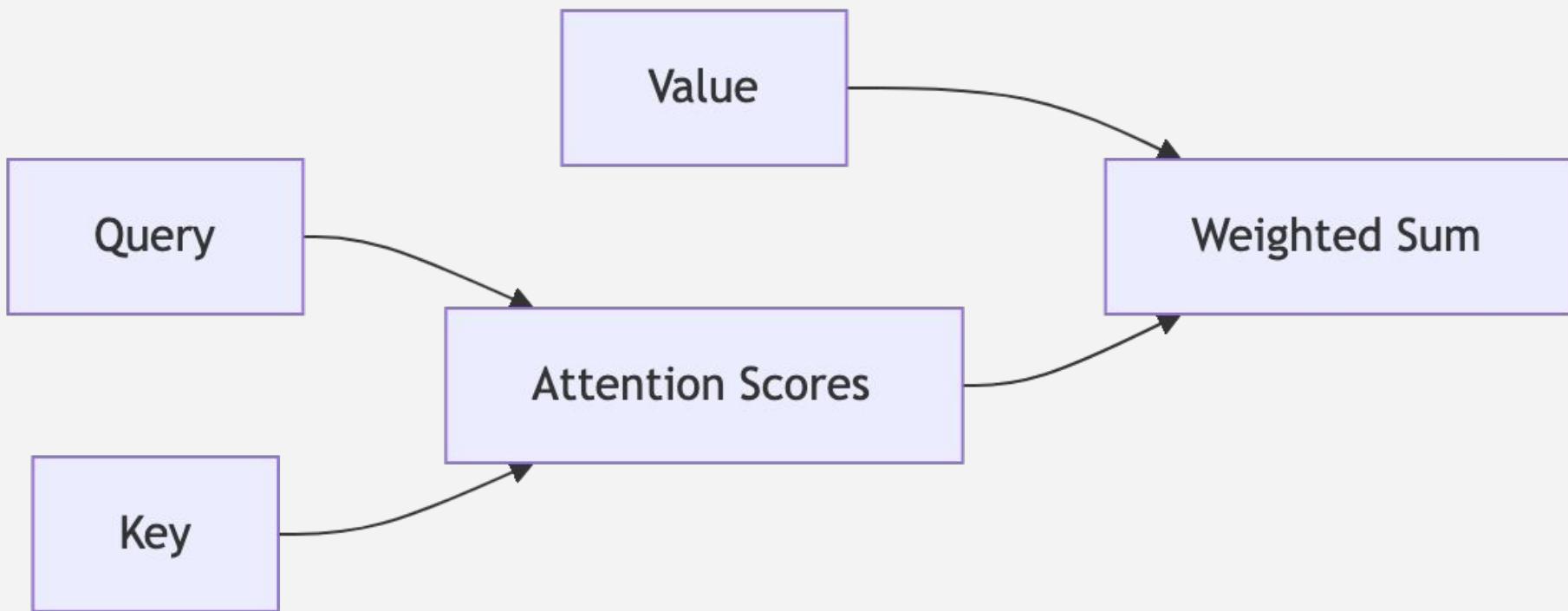
# The Transformer Architecture - Overview



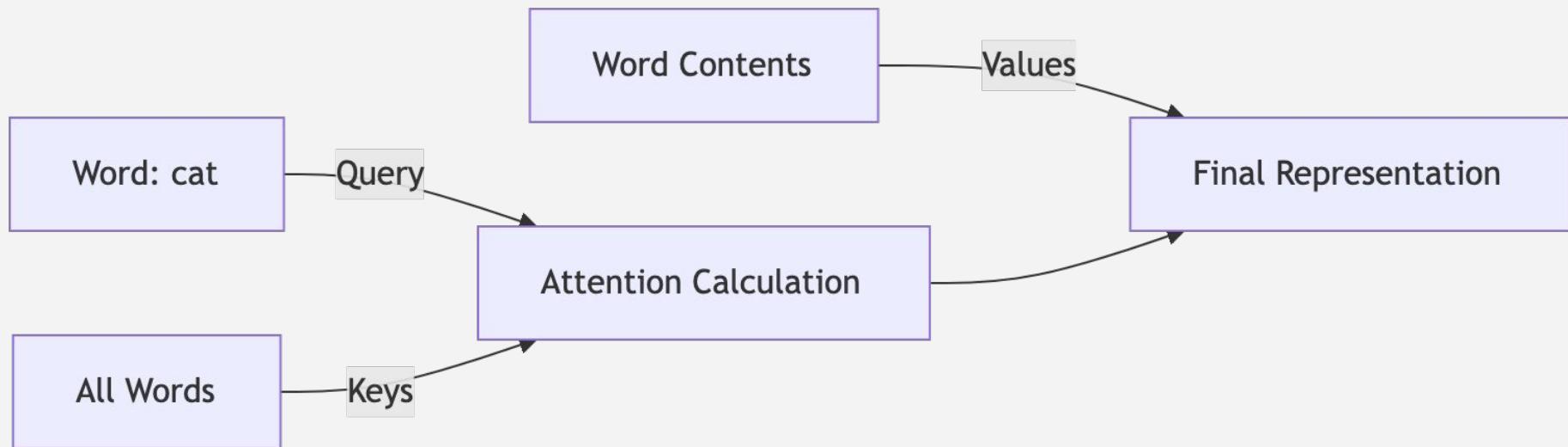
# Restaurant Kitchen Analogy



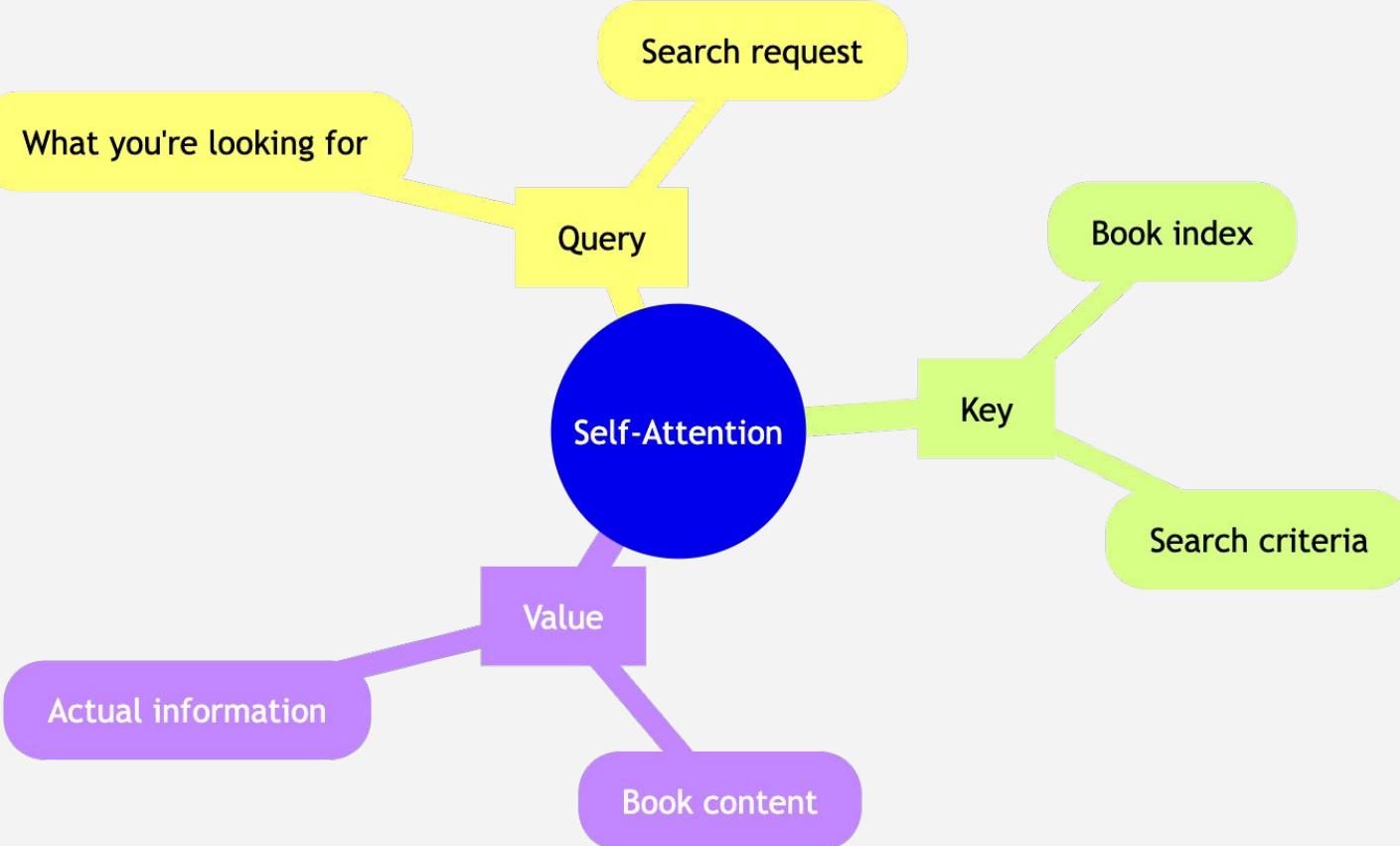
# Self-Attention Deep Dive



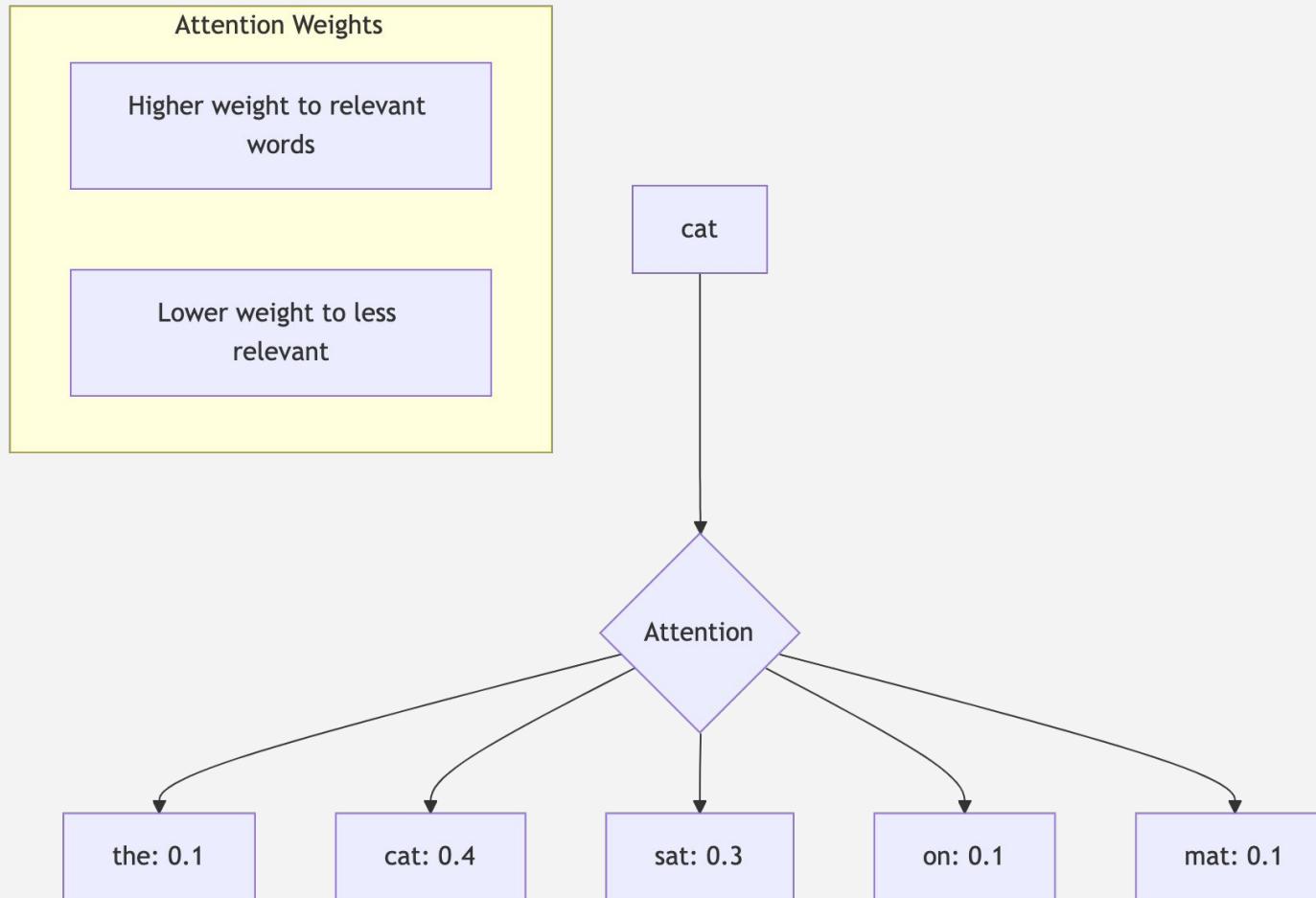
# Real Example



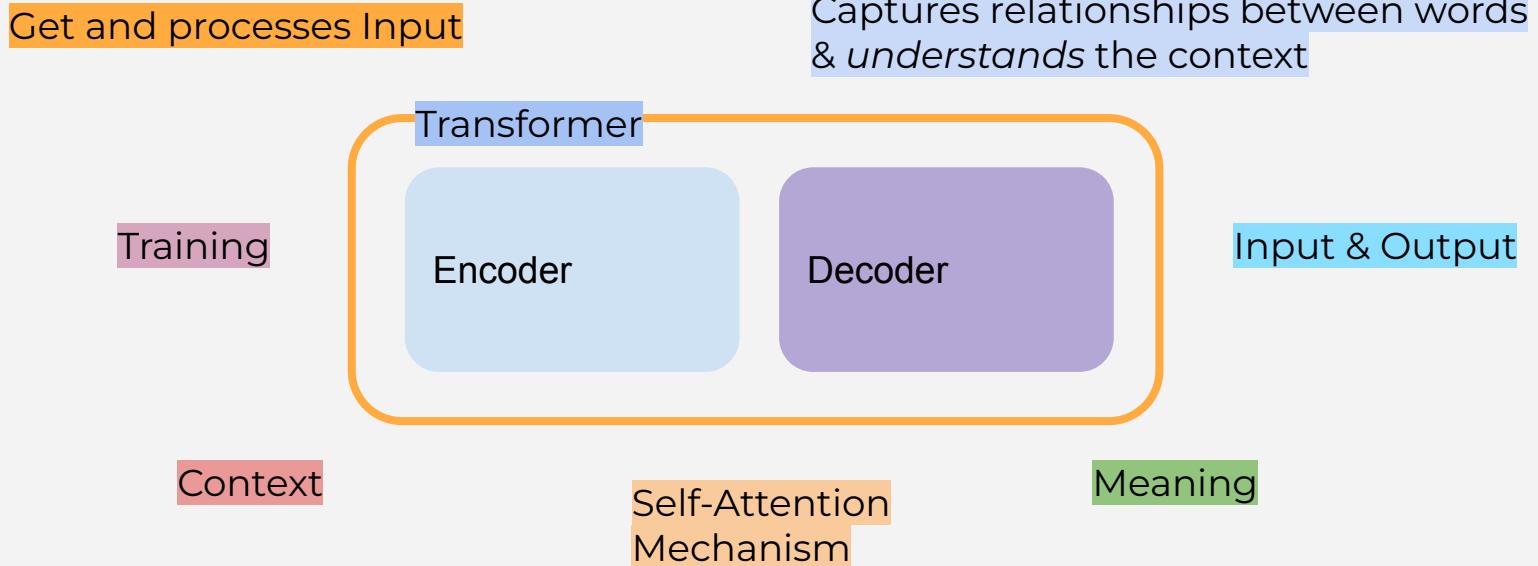
# Library Analogy



# Practical Example with Sentence



# The Transformer Architecture - Overview



# Understanding Tokens & LLMs

Large Language Model don't "Understand" text.

They "Understand" *tokens* - smaller pieces of text

GPT-3 Codex

I eat apples.

**Tokens**

4

**Characters**

13

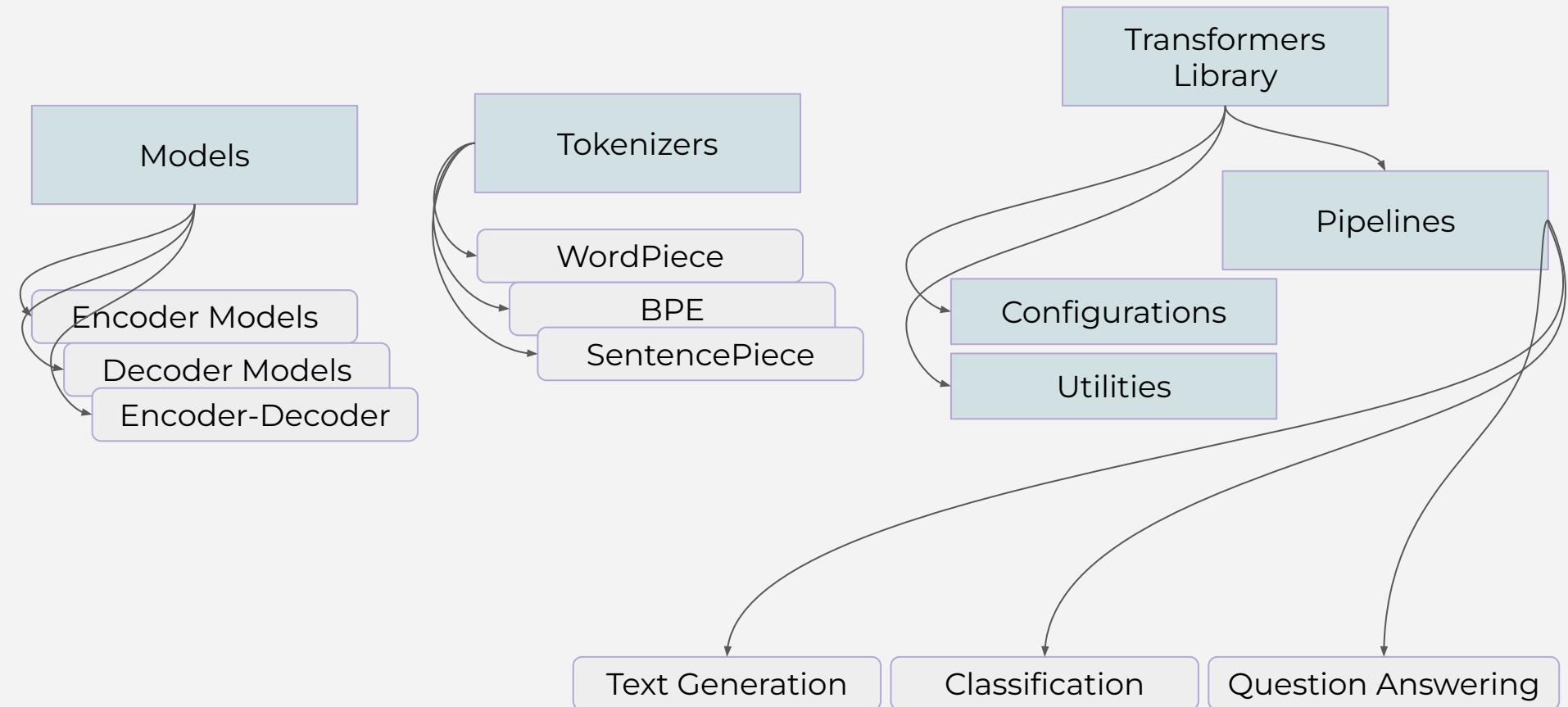
I eat apples.

(

# The Transformers Library

1. A comprehensive, modular toolkit for transformer models:
  - a. Tokenization
  - b. Model configuration
  - c. High-level pipelines

# The Transformers Library - Core Components

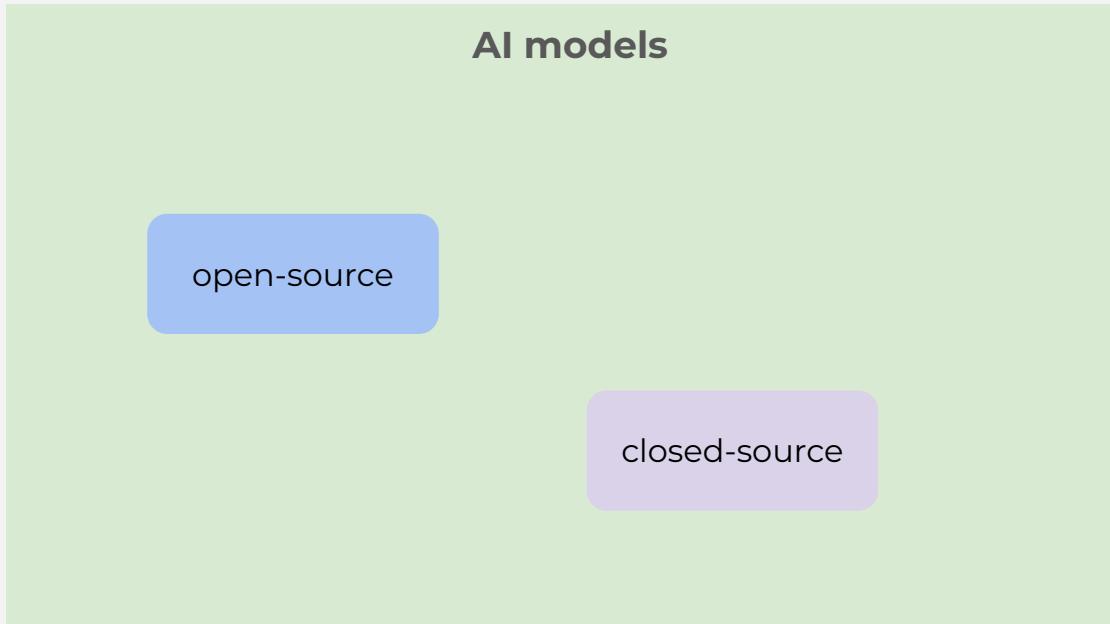


# Modern AI Application - Popular LLMs

The past a couple of years, we've seen a massive explosion of LLMs in the market...

- OpenAI Models (GPT-4...)
- Meta's LLaMA Models
- Anthropic
- Google's Models
- Open Source
  - DeepSeek, Mistral, Falcon...

# Open vs. Closed Source LLMs



# Open-source LLMs

open-source

Transparency

Cost Control

Customization  
flexibility

Community  
collaboration

**Challenges!**

- Initial setup complexity
- Hardware requirements
- Maintenance responsibility
- Performance optimization needs

# Closed-source LLMs

Closed-source

Reliability

Performance

Ease of Use

Enterprise  
Features

**Challenges!**

- Usage **costs**
- Data **privacy** concerns
- Limited customization
- **Vendor lock-in**

# OpenAI Models (Closed-source)

Closed-source

GPT-4

03

01

GPT-4o

...

- Usage **costs**
- Data **privacy** concerns
- Limited customization
- **Vendor lock-in**

# Key Decision Factors

- Budget constraints
- Privacy requirements
- Technical resources
- Performance resources
- Customization requirements

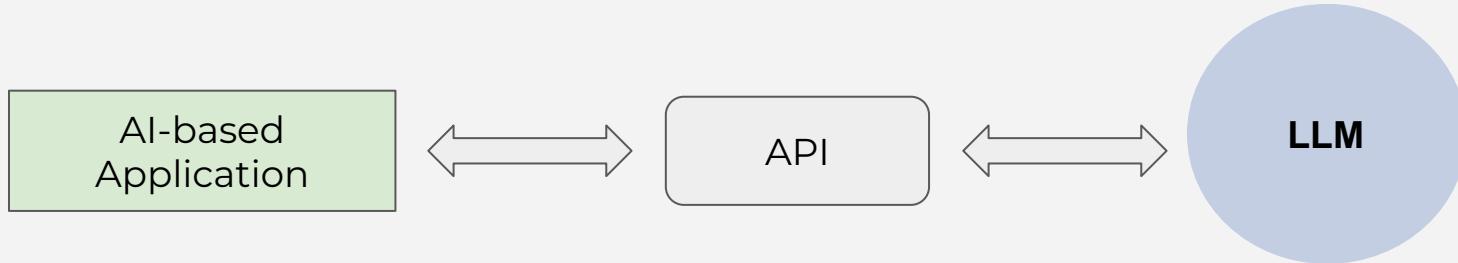
# Simple LLM Demo

1. We'll use the Transformers library for models
2. Build a simple LLM
3. Run it

# API Integration Basics (Hands-on)

- i. Setting up environment
- ii. API keys management
- iii. First API call
- iv. Basic prompt testing
- v. Cost monitoring

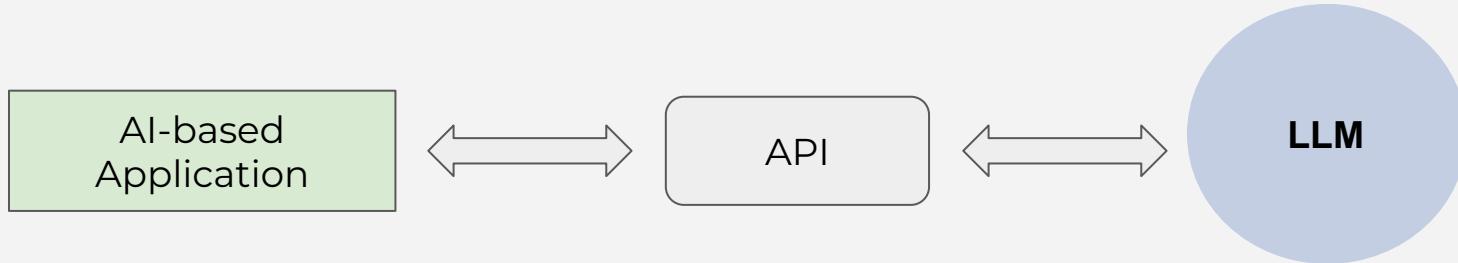
# Using APIs Effectively in AI Projects



**API** - Application Programming Interface

- **Faster development**
- **Access to powerful tools** (models like GPT4, Llama...)
- **Scalability**

# Using APIs Effectively in AI Projects



## Authentication:

- API's require an API Key to authorize requests
- Use HTTP requests (like GET or POST) to send and receive data

Example: “send a prompt to GPT and get a response”

# Open-source Models - Ollama

Introduce Ollama as an alternative to  
closed-source  
*Re-use ollama course content fully*

# Prompting

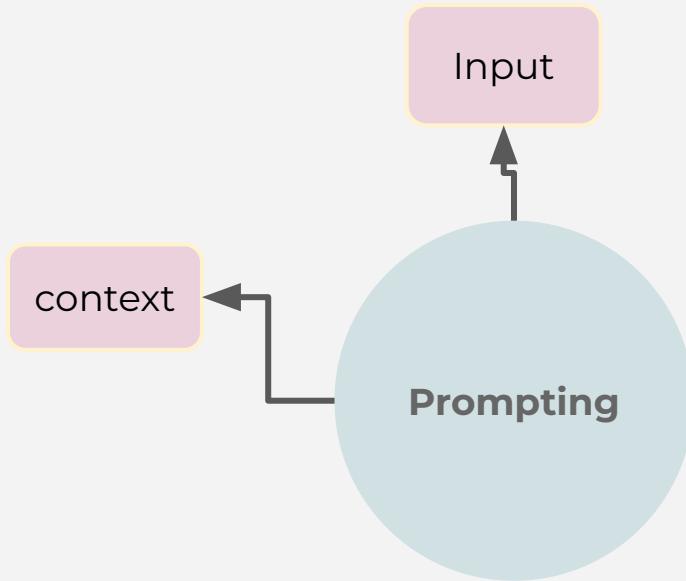
- *Best way to communicate with LLMs*

- Prompt engineering
- Basic components
- Prompting techniques
- Key concepts
-

# Prompt Engineering

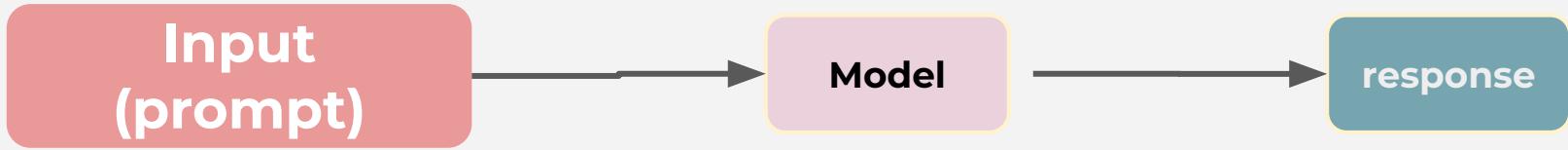
The art of writing clear and specific prompts (*instructions*) to get the desired output from an LLM

# Prompting



## What is it?

- Provide **input** (prompt) to an AI model for a desired **response**.



# Key concepts

## Input-output Relationship

Input goes into the model which generates the output/response based on the model's training

## Contextual Understanding

Model infers, from the input, the **task**, **tone** and desired **format** of the **response**.

## Flexibility

Prompts can range from **simple** to **complex**.

# Prompt engineering

Prompt Engineering

The **art** and **science** of designing effective prompts to achieve specific outcomes from AI models.

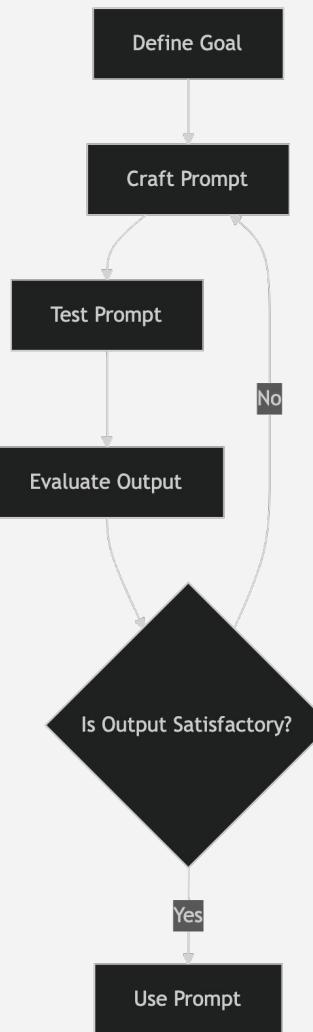
- Crafting prompts to **maximize** the model's performance, **accuracy** and **relevance**.

# Prompt engineering - Why it matters?

## Prompt Engineering

- **Improves output quality:**
  - Well designed prompts lead to more accurate, relevant and useful responses
- **Reduces ambiguity:**
  - Clear prompts minimize misunderstanding and irrelevant outputs
- **Enables Control:**
  - Allows users to guide the model's behavior and tailor outputs to specific needs.

# Prompt engineering workflow



# Types of prompts

Direct prompts

Open-ended prompts

- **Explicit instructions**
  - **Example:** “What is the capital of Benin?”
  - Use case: simple, factual queries
- **Encourage creative or explanatory responses**
  - **Example:** “tell me a story about a dragon and a knight”
  - **Use case:** creative writing, brainstorming...

# Types of prompts

Instructional prompts

- **Detailed instructions for a specific task.**
  - **Example:** “Write a 300-word summary of the benefits of exercise, using bullet points”
  - Use case: Structured tasks like summarization or analysis
- **Assigning a role or persona to the model.**
  - **Example:** “You are a historian. Explain the causes of World War I.”
  - **Use case:** Context-specific responses.

Role-base prompt

# Types of prompts

## Chain-of-thought prompts

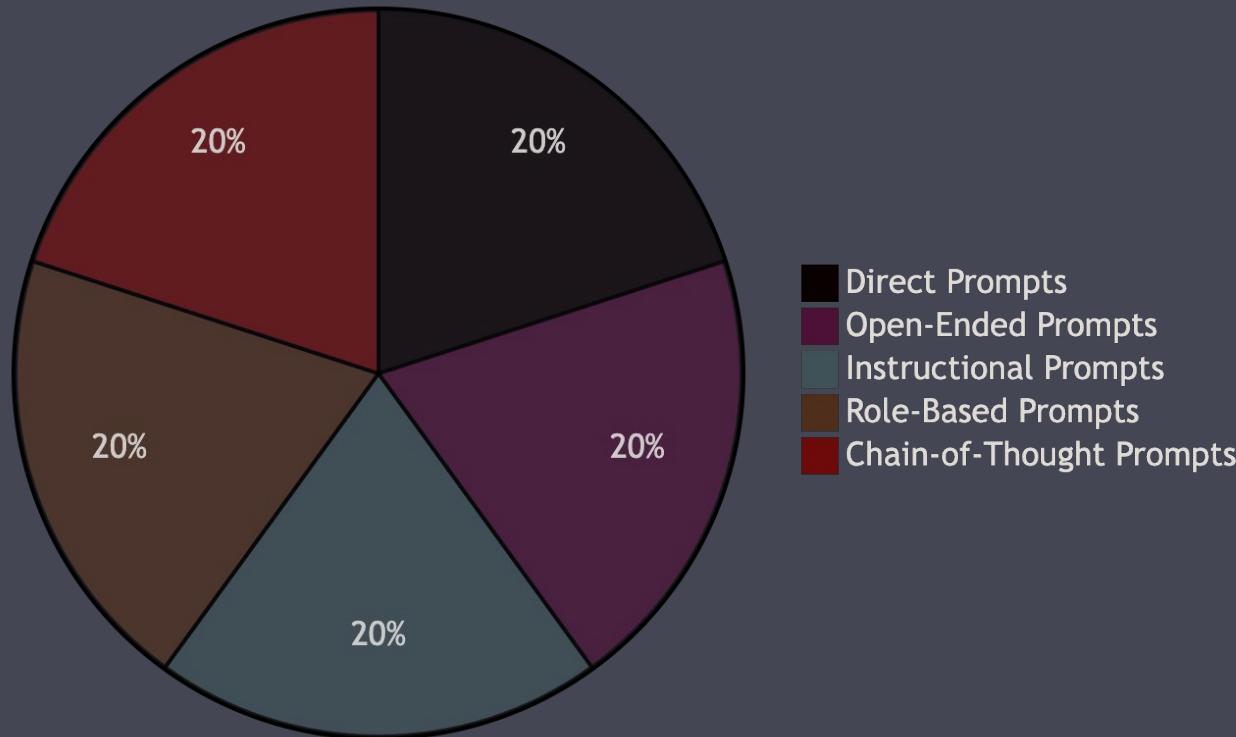
- **Encouraging the model to "think step by step."**
  - **Example:** “Solve this math problem step by step:  $3x + 5 = 20$ .”
  - **Use case:** Complex problem-solving.

## Role-base prompt

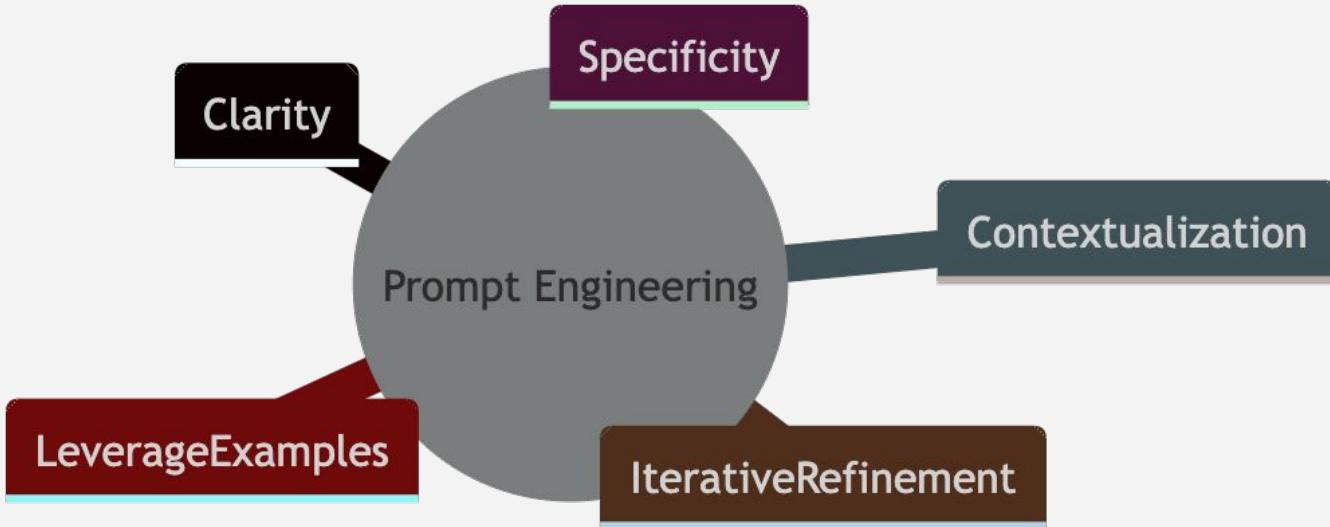
- **Assigning a role or persona to the model.**
  - **Example:** “You are a historian. Explain the causes of World War I.”
  - **Use case:** Context-specific responses.

# Types of prompts

Types of Prompts

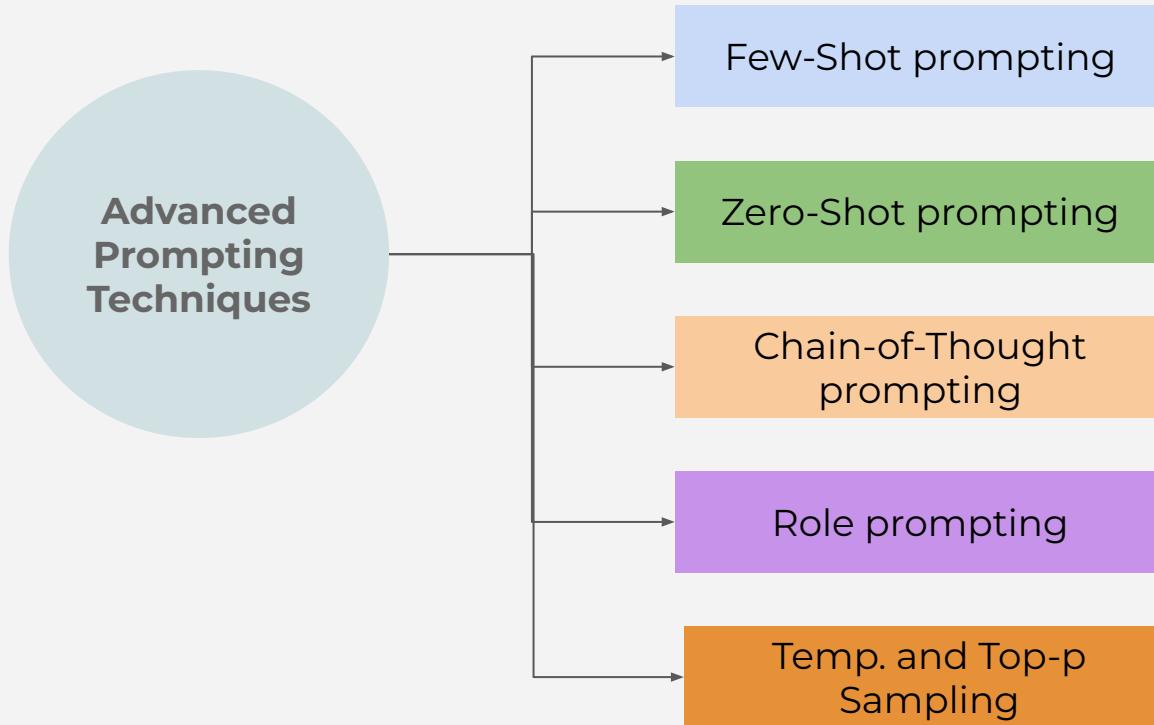


# Principles of effective prompts engineering

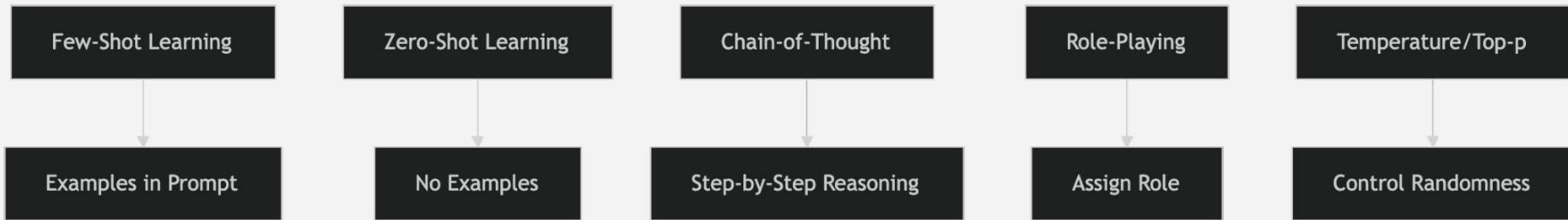


# Advanced prompting techniques

# Advanced prompting techniques



# Advanced prompting techniques



# Advanced prompting techniques

## Few-Shot prompting

Provide a few examples in the prompt to guide the model.

Example: "Translate these sentences: 'Hello' -> 'Hola', 'Goodbye' -> 'Adiós'. Now translate: 'Thank you'!"

# Advanced prompting techniques

## Zero-Shot prompting

Ask the model to perform a task without examples

Example: "Translate 'Good morning' into Spanish."

# Advanced prompting techniques

Chain-of-Thought  
prompting

Encourage the model to **break down** complex tasks into steps.

Example: "Solve this problem step by step: If John has 5 apples and gives 2 to Mary, how many does he have left?"

# Advanced prompting techniques

## Role prompting

Assign a role to the model to tailor its responses.

Example: "You are a career coach. Give advice to someone switching careers."

# Advanced prompting techniques

Temp. and Top-p  
Sampling

Adjust parameters like temperature (creativity) and top-p (diversity) to control output randomness.

Example: Use low temperature for factual responses and high temperature for creative writing.

# Challenges in prompt engineering

## Ambiguity

- Poor prompts can lead to irrelevant or incorrect results
- Solution: be specific and **test multiple** variations

## Bias

- Prompts can inadvertently introduce bias into the model's response
- Solution: use open-ended prompts for creative tasks

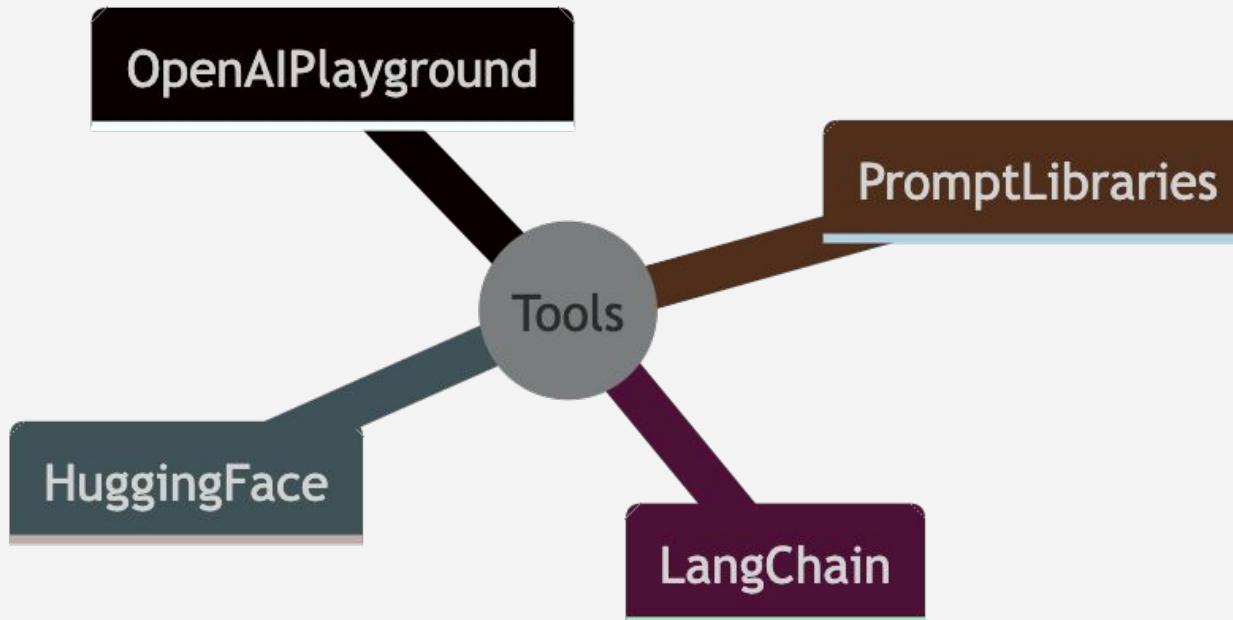
## Scalability

- Crafting effective prompts for large-scale applications can be time-consuming.
- Solution: develop reusable templates and automate where possible

# Challenges in prompt engineering



# Tools and Frameworks for Prompt Engineering



# Conclusion: Key takeaways

- **Prompting and prompt engineering** are essential skills for leveraging the full potential of AI models.
- By understanding the principles, techniques, and challenges, you can craft prompts that:
  - Unlock **accurate** responses.
  - Enable **creative** outputs.
  - Ensure **context-aware** results.
- Whether you're a **developer**, **writer**, or **business professional**, mastering prompt engineering will:
  - Empower you to achieve your goals more effectively.
  - Enhance your ability to interact with AI systems.

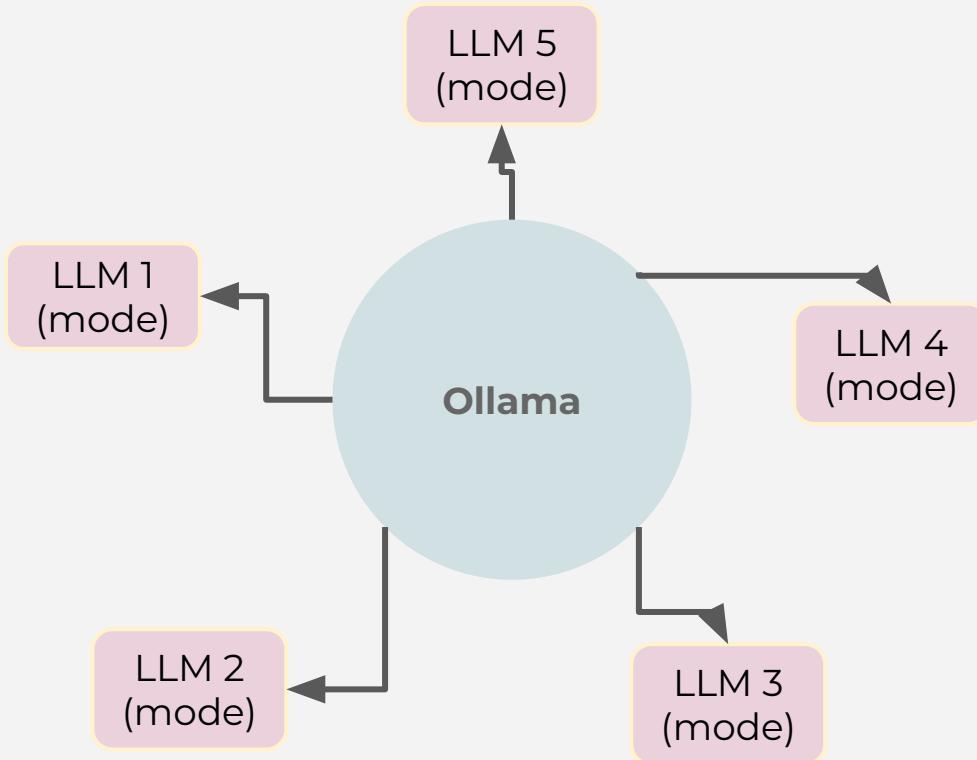
# Hands-on: Prompt Engineering

# Ollama setup

# *Ollama* *Deep Dive*

- What is it?
- Why (motivation)?
- Advantages

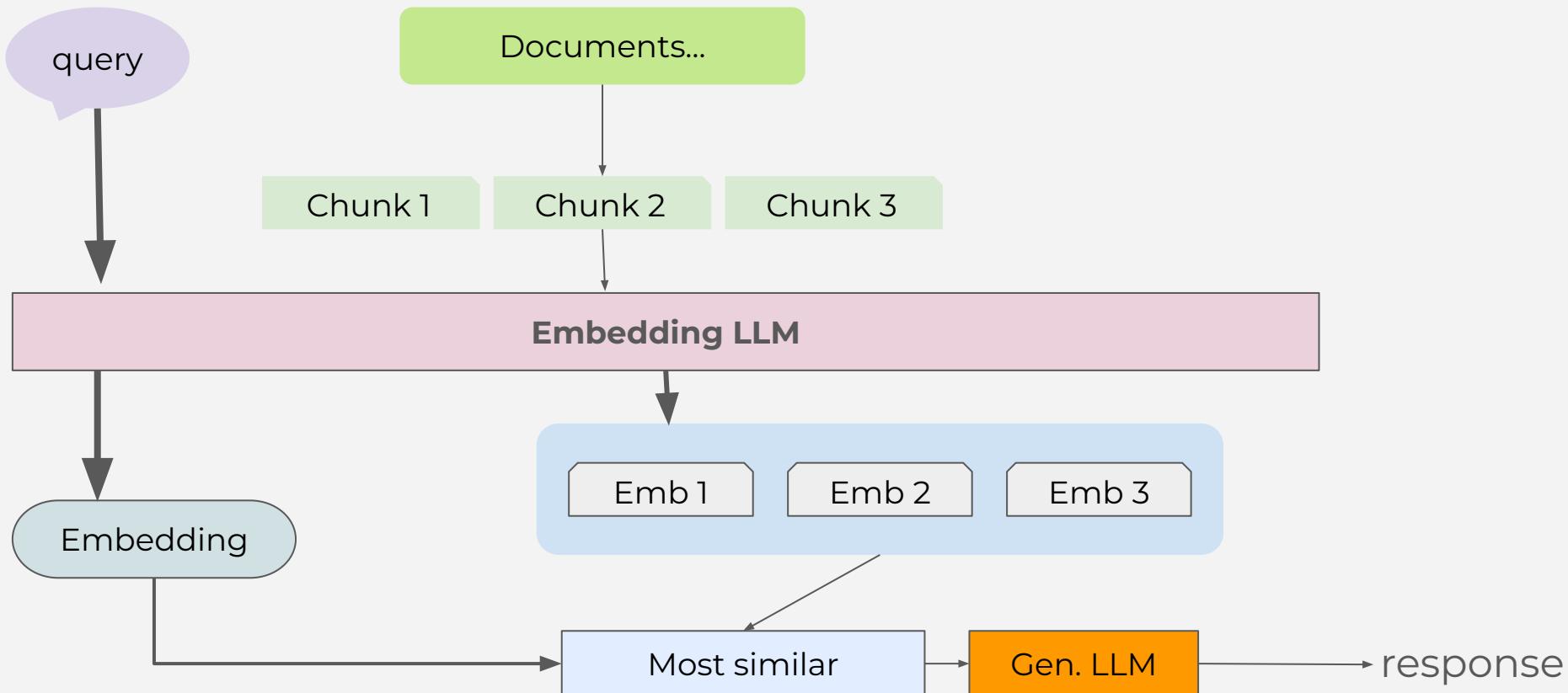
# Ollama



At its core:

- CLI - command-line interface (*manages installation and execution of LLMs locally*)

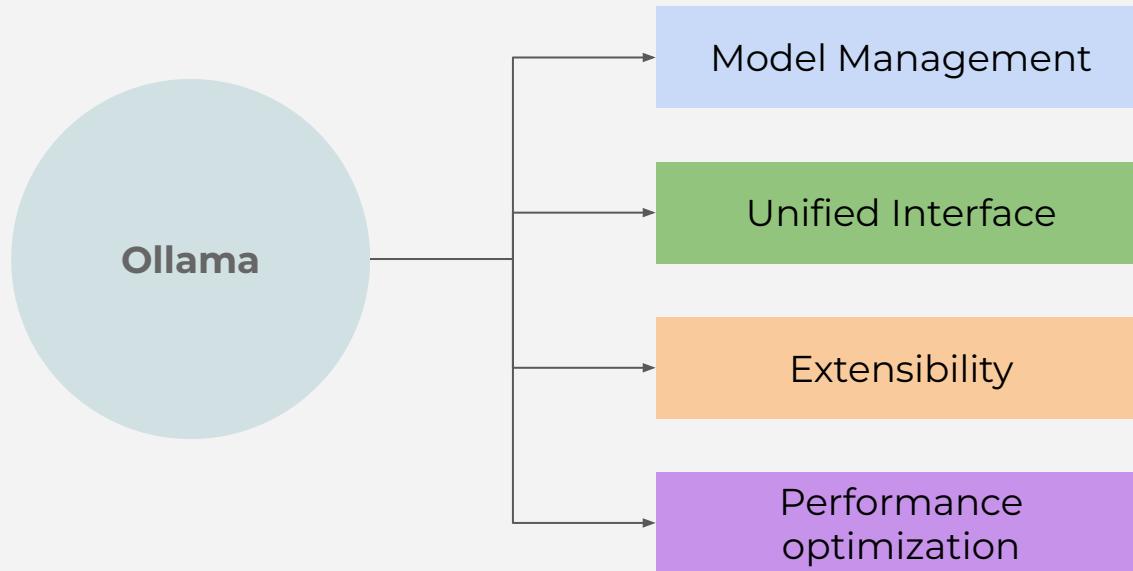
# What problem does it solve?



# What problem does it solve?

1. **Privacy concerns** - running LLMs locally provides a lot of security.
2. **Ease of use** - setting up LLMs can be challenging. With Ollama, it's an easy process
3. **Cost efficiency** - no more cloud-based services (which can be costly)
4. **Latency reduction** - local execution reduces the latency issues
5. **Customization** - greater flexibility in customizing our models

# Key features



# Use cases

Development & testing

Education and  
Research

Secure applications

# *Ollama* **setup**

- Installation
- Setup

# System requirements

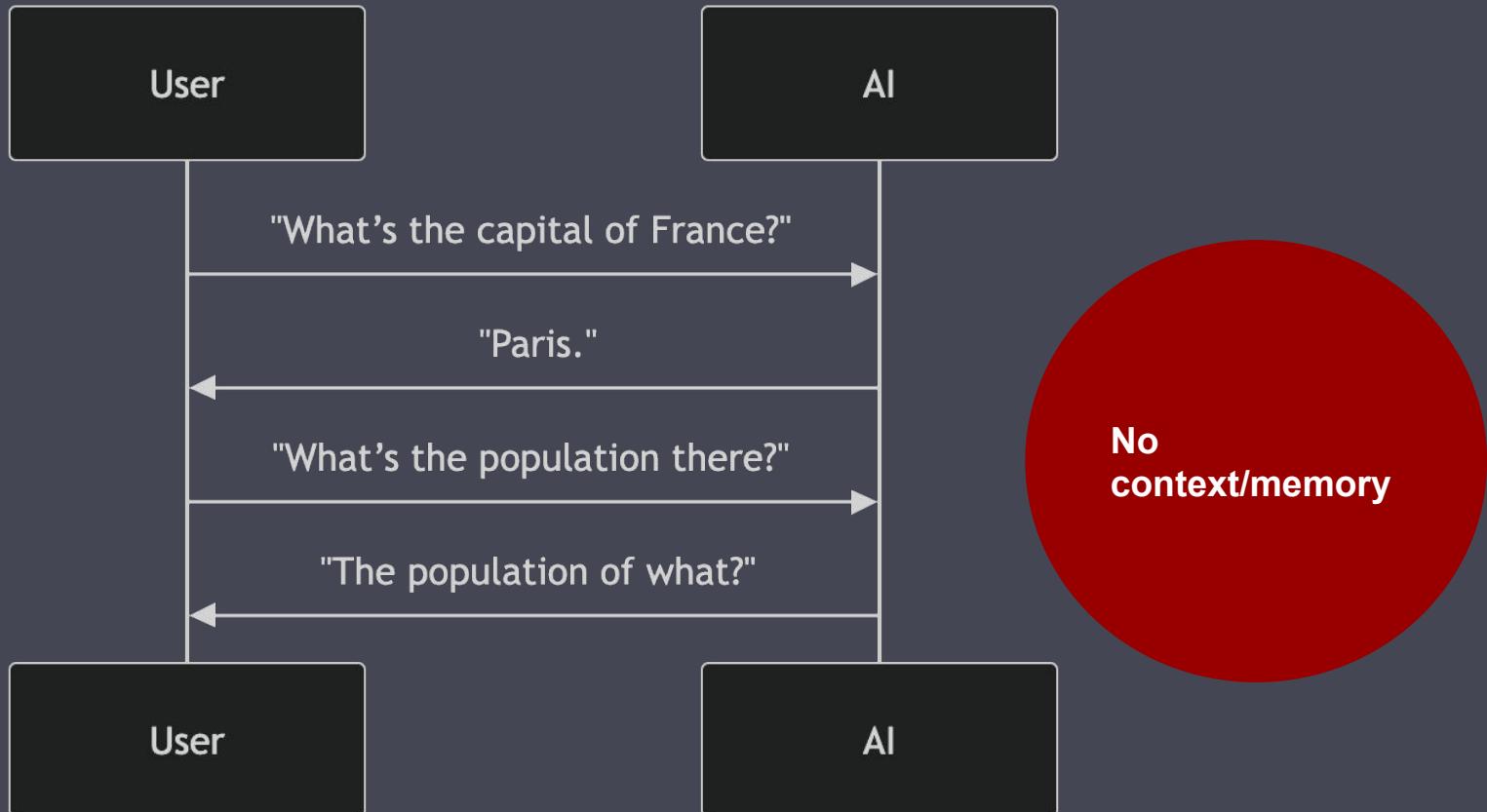
OS: Mac, Linux & Win

Storage (at least ~10GB  
free)

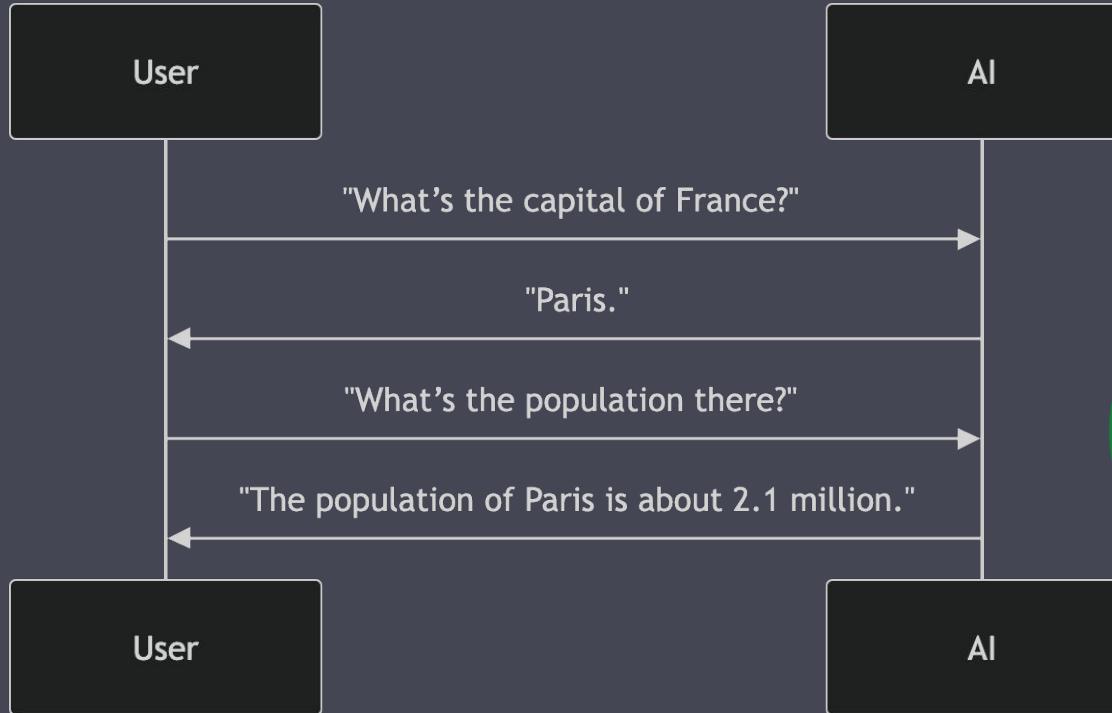
Processor: modern CPU

# **Context & Memory Management - A Deep Dive**

# Motivation (Memory & Context)



# Motivation (Memory & Context)



# What is Context and Memory Management?

*A way to make sure that the AI “remembers” previous context...*

## Context

Information/state that's relevant to the current interaction.

## Memory Management

The process of storing, retrieving and managing context over multiple interactions (for coherent and relevant responses)

# Why It matters?

*A way to make sure that the AI “remembers” previous context...*

## Coherence

Ensures that the AI's responses are consistent and relevant to the ongoing conversation

## Personalization

AI remembers user preferences, past interactions and other relevant details

## Efficiency

Reduces redundancy by reusing stored context instead of reprocessing it.

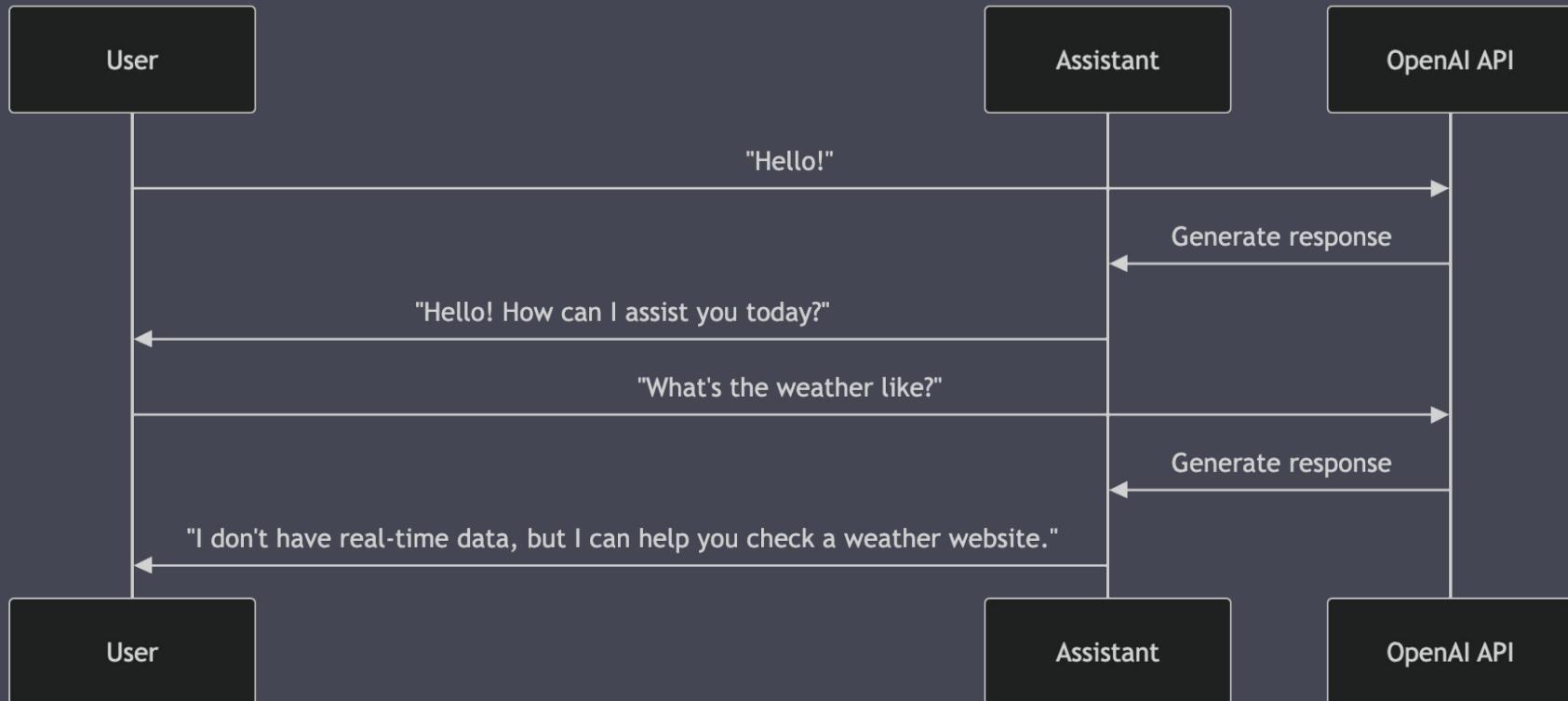
# How Context and Memory Work in OpenAI API

Key concepts:

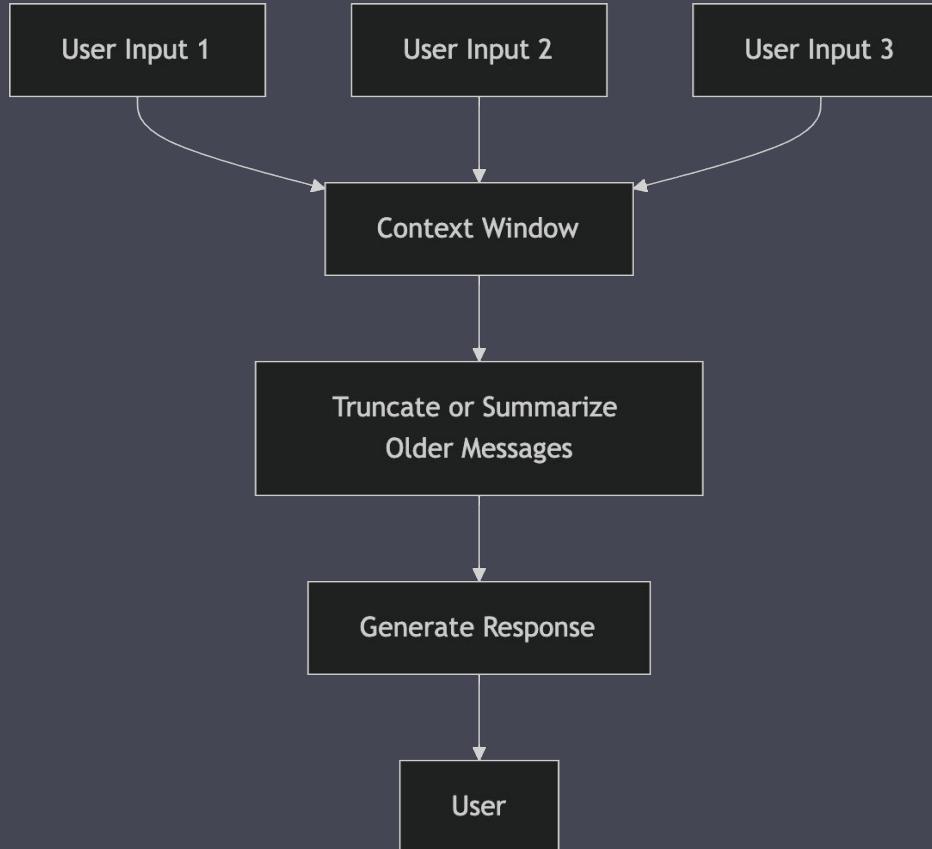
- Message history:
  - The OpenAI API uses a “messages” array to manage context
  - Each message has “role (system, or assistant” and “content - the text of the message

```
# Initialize conversation with a system message
messages = [
    {"role": "system", "content": "You are a helpful assistant."}
]
```

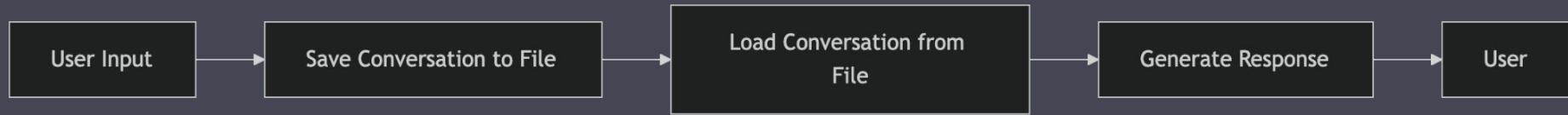
# Basic Conversation Flow



# Context Window



# Persistent Memory Across Sessions



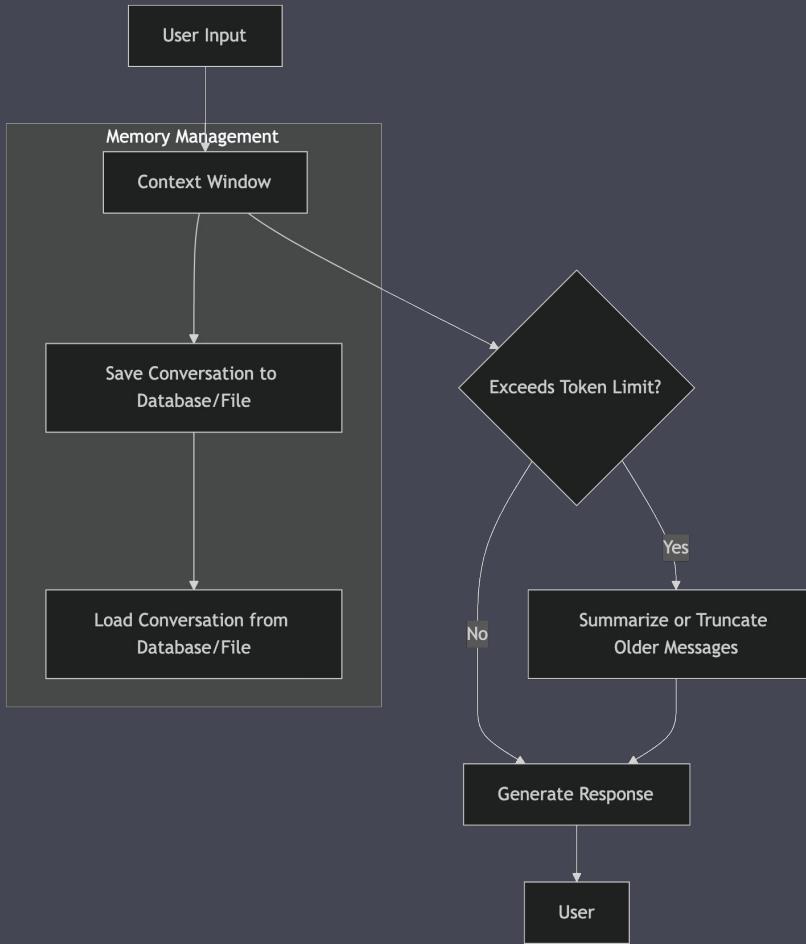
# Use cases (Context & Memory Management)

1. Chatbots and Virtual Assistants:
  - a. Maintain context across multiple turns
    - i. Example: customer support chatbot
2. Personalized recommendations:
  - a. Remember user preferences (past interactions)
    - i. A movie recommendation system
3. Long-term conversations:
  - a. Summarizing or truncating older messages

# Challenges (Context & Memory Management)

1. Context window limitations
  - a. Limited token capacity
2. Summarization accuracy
  - a. Summarizing may lose crucial details
3. Memory persistence:
  - a. Storing and retrieving conversation history is complicated

# Context and Memory Management Workflow



# Conclusion

1. Context and Memory Management are essential for building:
  - a. Coherent
  - b. Personalized
  - c. Efficient systems

# *Logging in LLM Applications*

- What is it?
- Why?
- Implementation

# What is Logging?

Logging

*... the process of recording events, actions, and data during the execution of an application.*

# Why Logging?

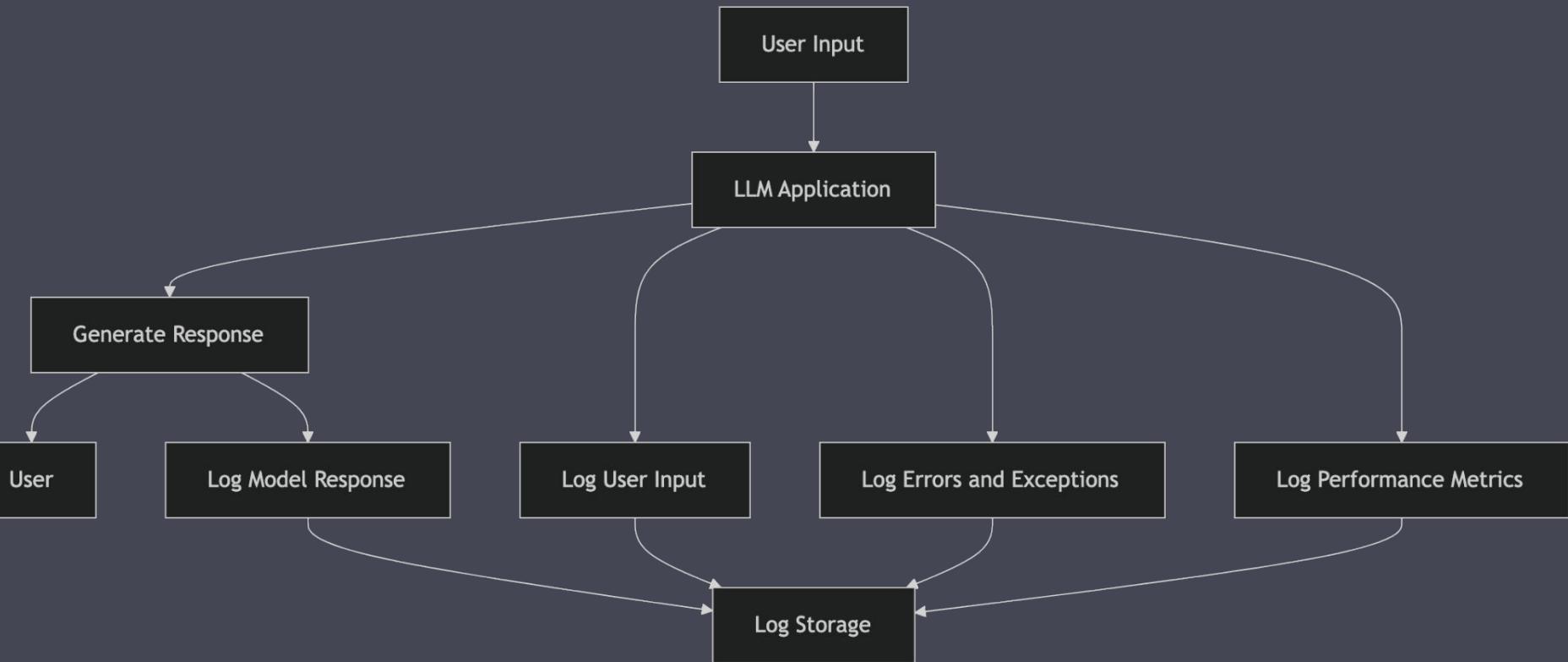
*Helps developers understand what happened in the application*

- *Especially when something goes wrong.*

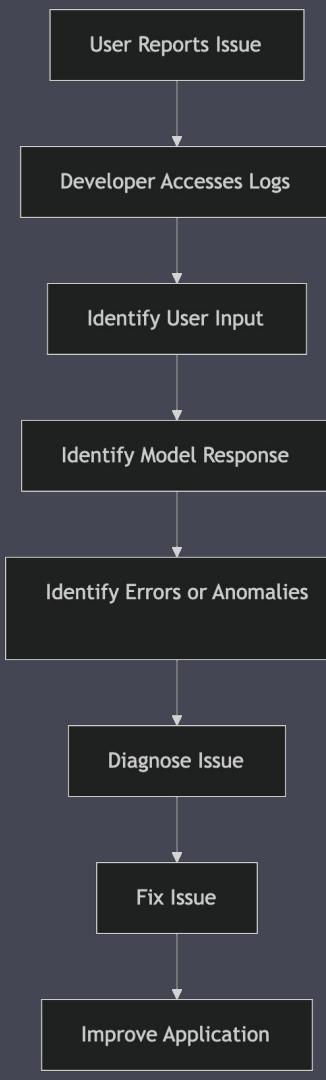
# Why is Logging Important in LLM Applications?

1. Debugging and troubleshooting
  - a. LLMs are complex systems (unpredictable)
  - b. Logs help developers trace the flow of data, identify errors...
2. Monitoring and performance:
  - a. Logs provide insight into the performance of the LLM (response times, token usage, error rates)
3. Compliance and auditing
4. User experience improvement
5. Security

# Logging in LLM Applications

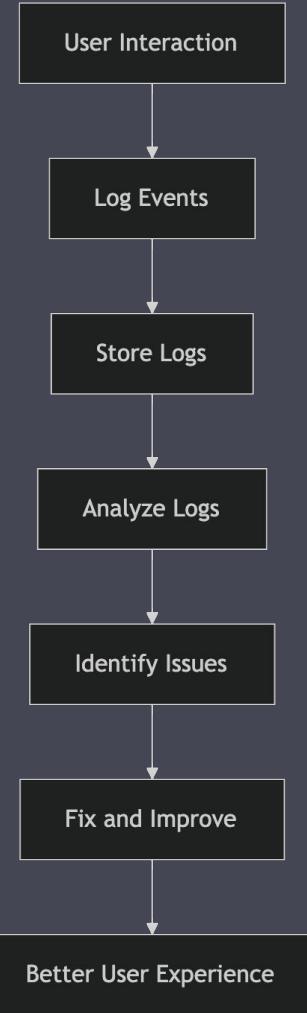


# Logging in LLM Applications

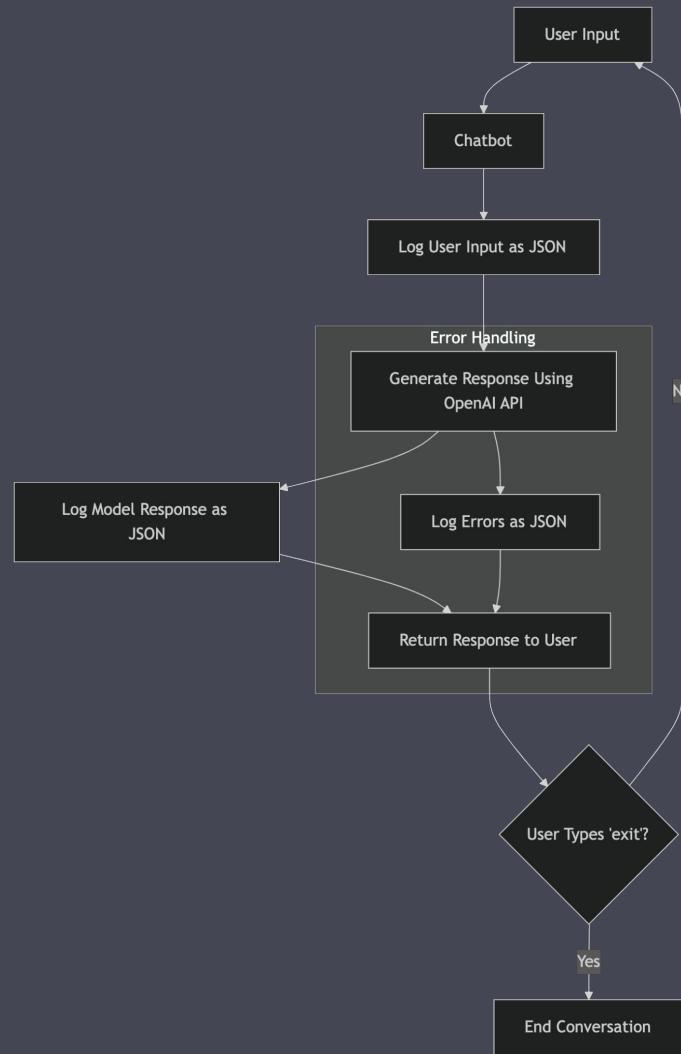


```
{  
  "timestamp": "2023-10-01T12:34:56Z",  
  "session_id": "12345",  
  "user_input": "What's the capital of France?",  
  "model_response": "The capital of France is Paris.",  
  "response_time": 1.2,  
  "tokens_used": 45  
}
```

# Logging Lifecycle



# LLM Logging Hands-on



# Summary

1. Logging
  - a. What is it?
  - b. Why/Motivation
  - c. Logging Lifecycle
  - d. Hands-on: Chatbot with Logging

# **RAG - Retrieval-Augmented Generation**

# **RAG:**

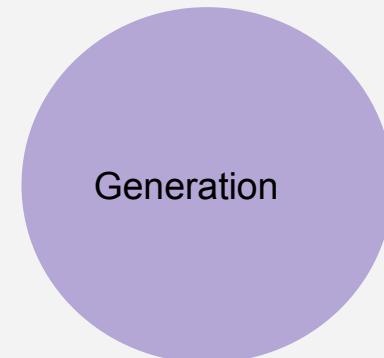
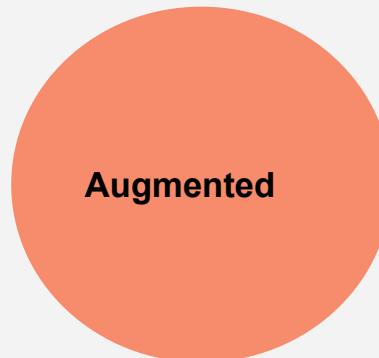
## **Retrieval-Augmented**

## **Generation**

- What is RAG (overview)
- Key concepts
- How it works?
- Advantages

# What is a RAG?

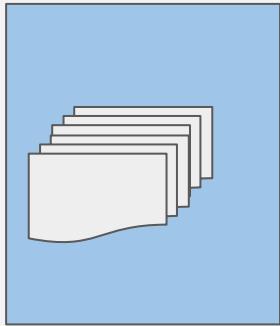
Retrieval-Augmented Generation



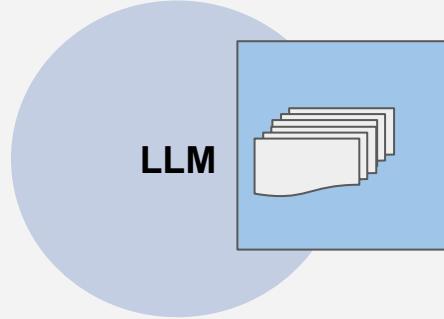
# What is a RAG?

*Definition - a framework that combines the strengths of retrieval-based systems and generation-based models to produce more accurate and contextual relevant response*

**Translation** - ... efficient way to customize an LLM (model) with your own data.



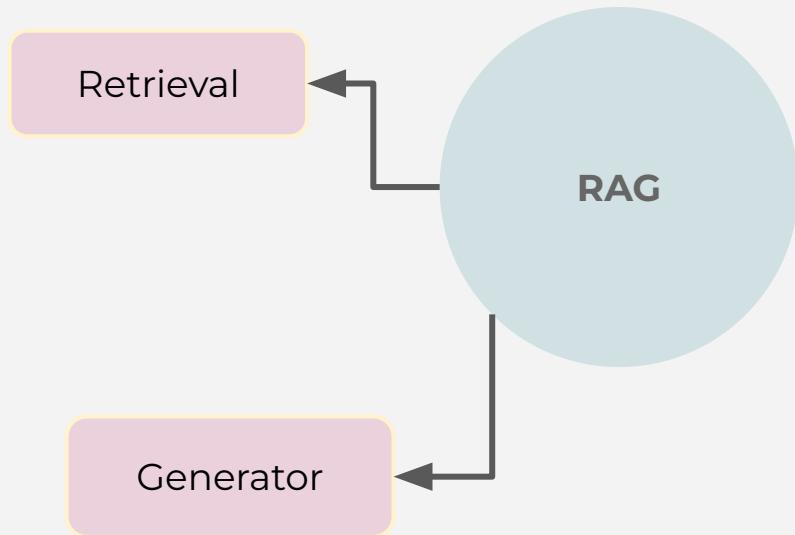
Inject own  
data



*I only know what I was  
trained on..*

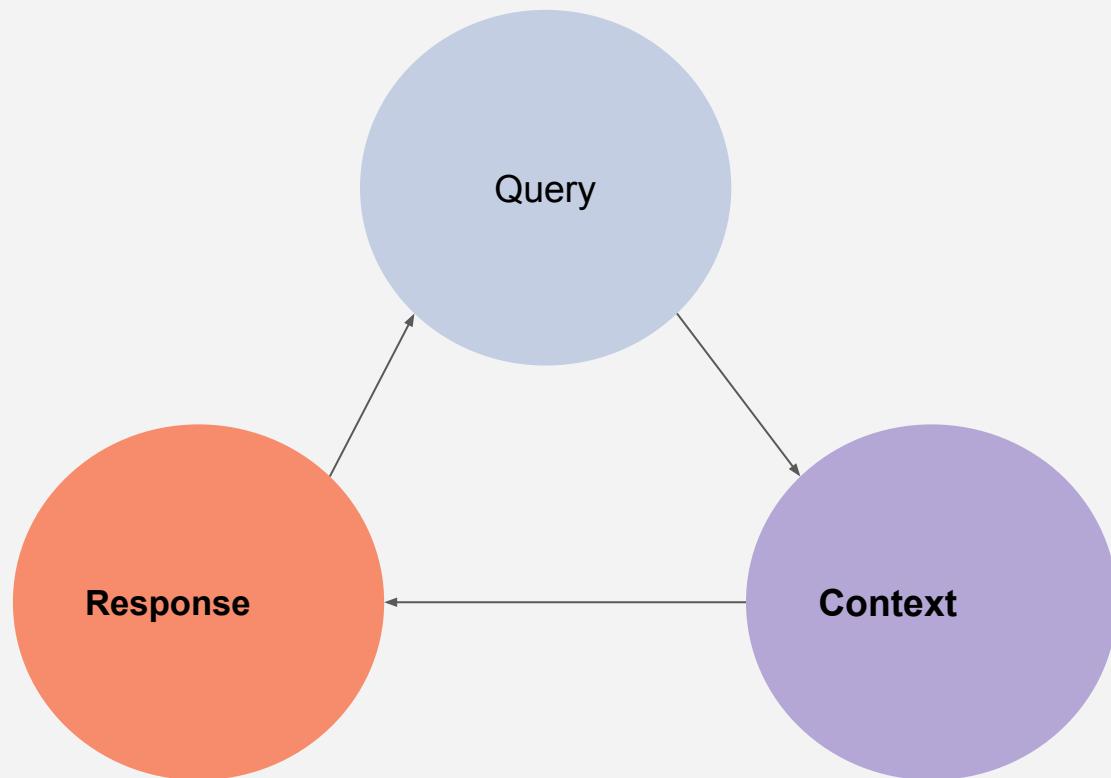
*Now I also know about  
specific contextual data :)*

# Key components



# RAG Triad

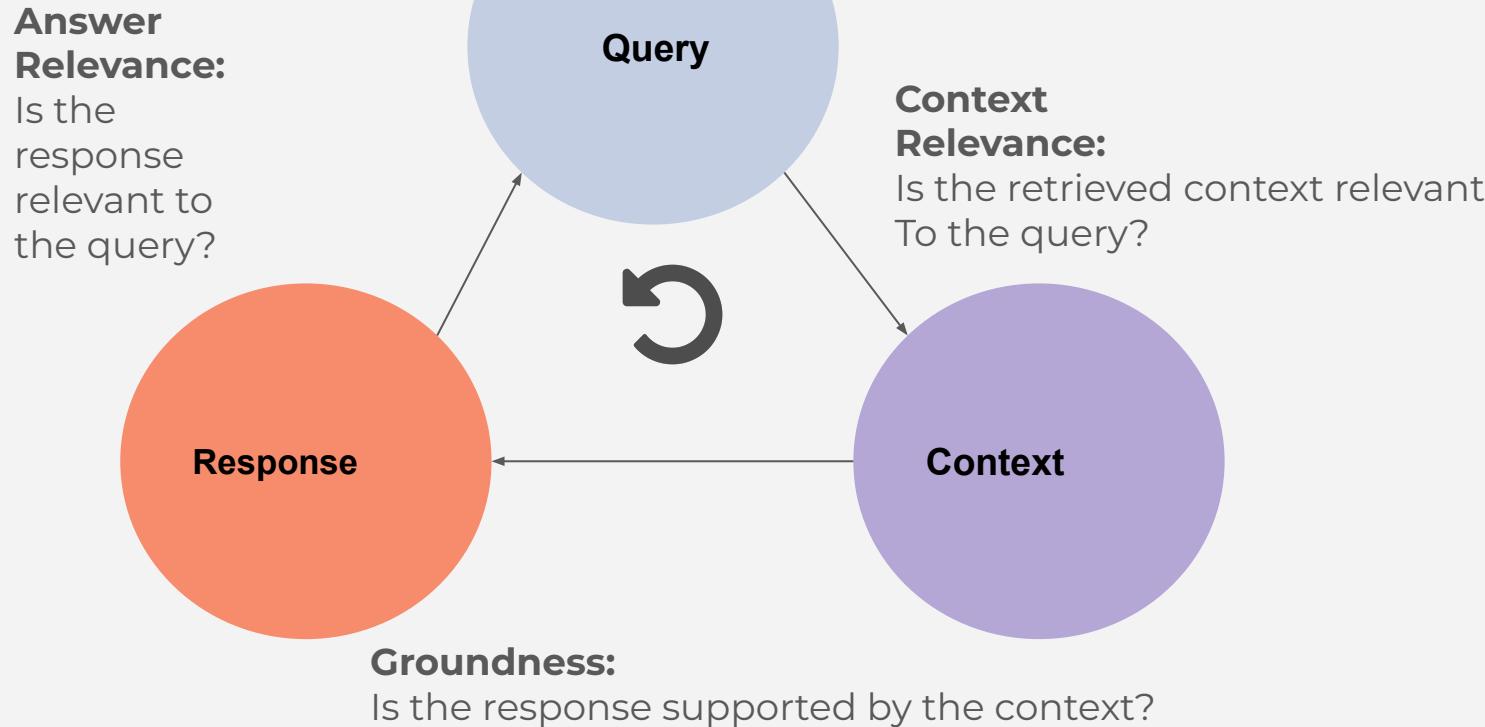
Retrieval-Augmented Generation



The RAG Triad

- Context Relevance
- Groundness
- Answer Relevance

# The RAG Triad

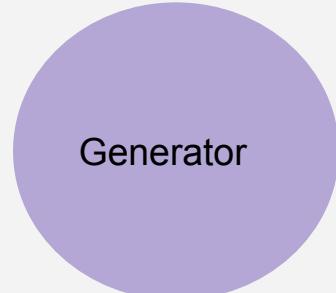


# Components of RAG



Retriever

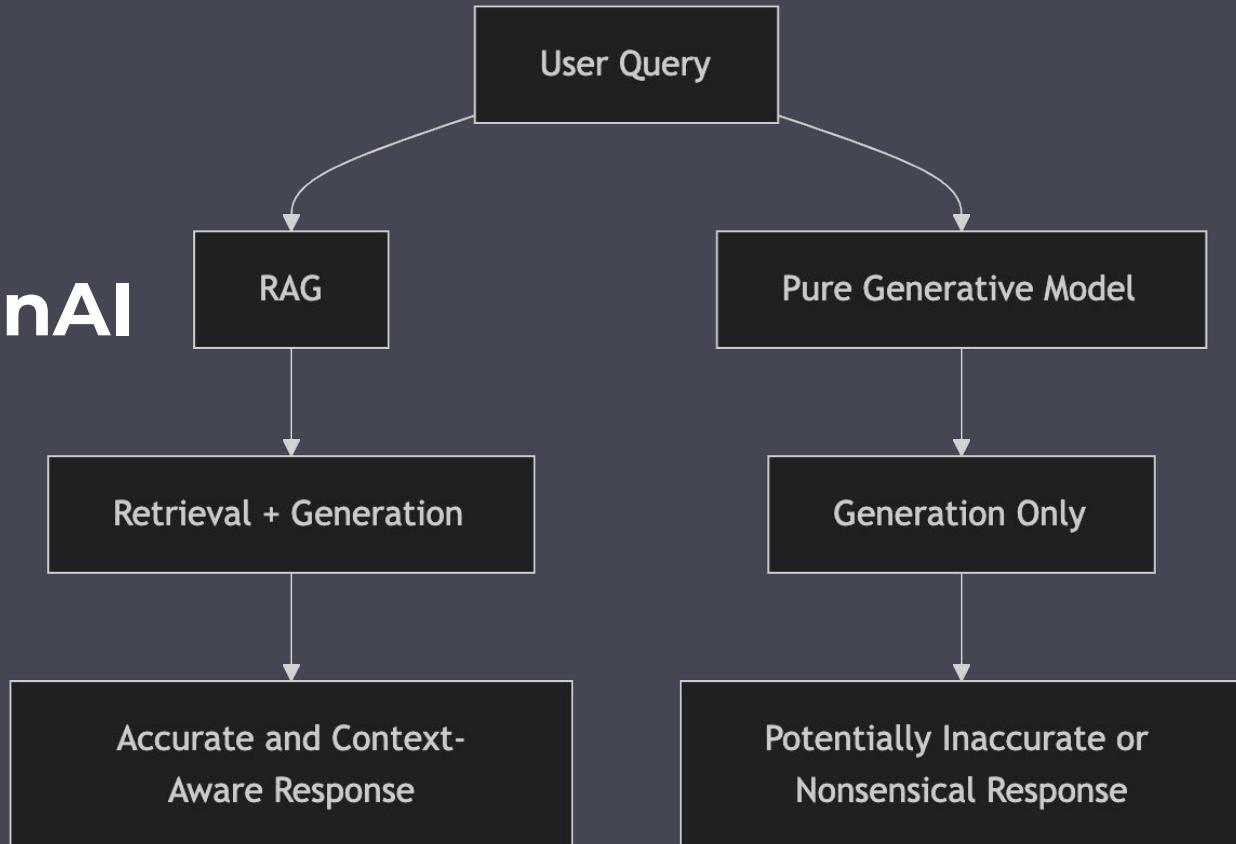
Identifies and retrieves relevant documents



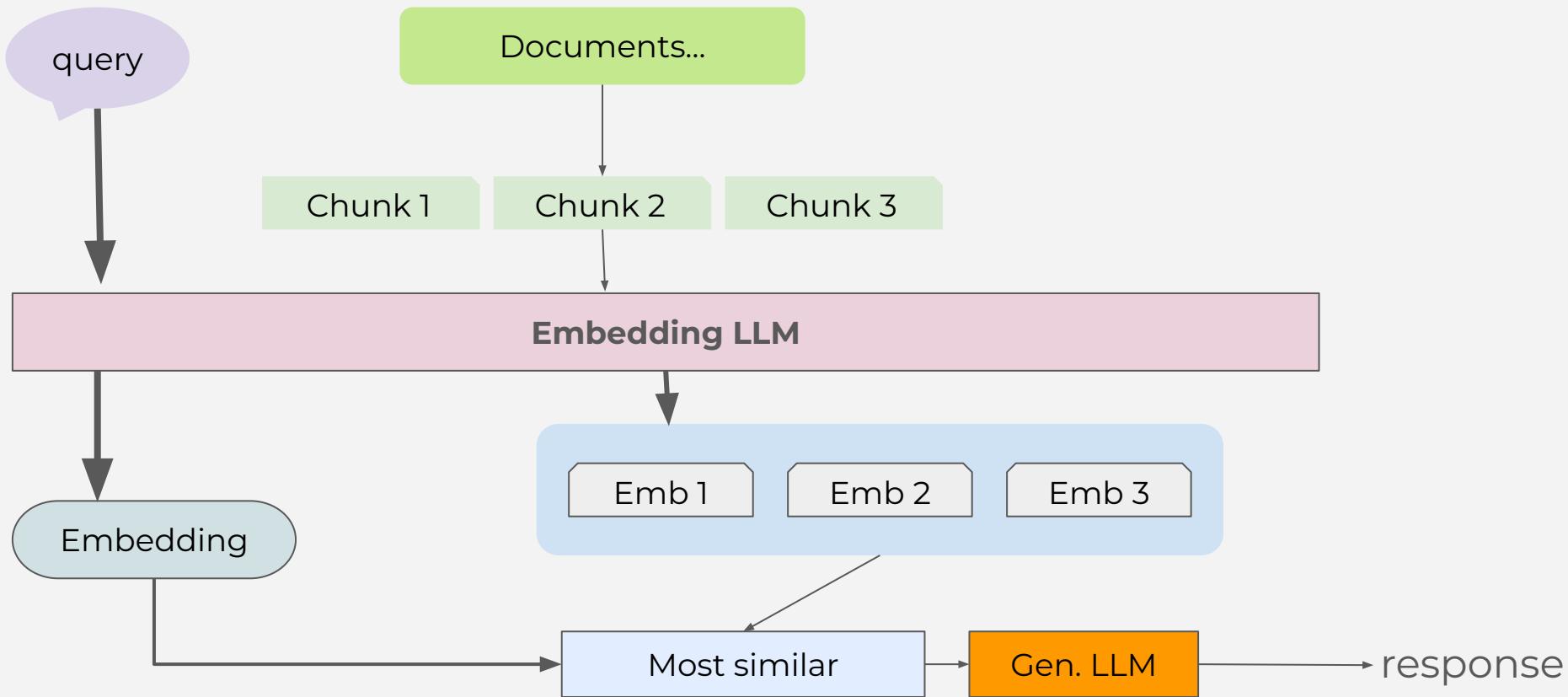
Generator

Takes retrieved docs and the input query to generate coherent and contextually relevant response.

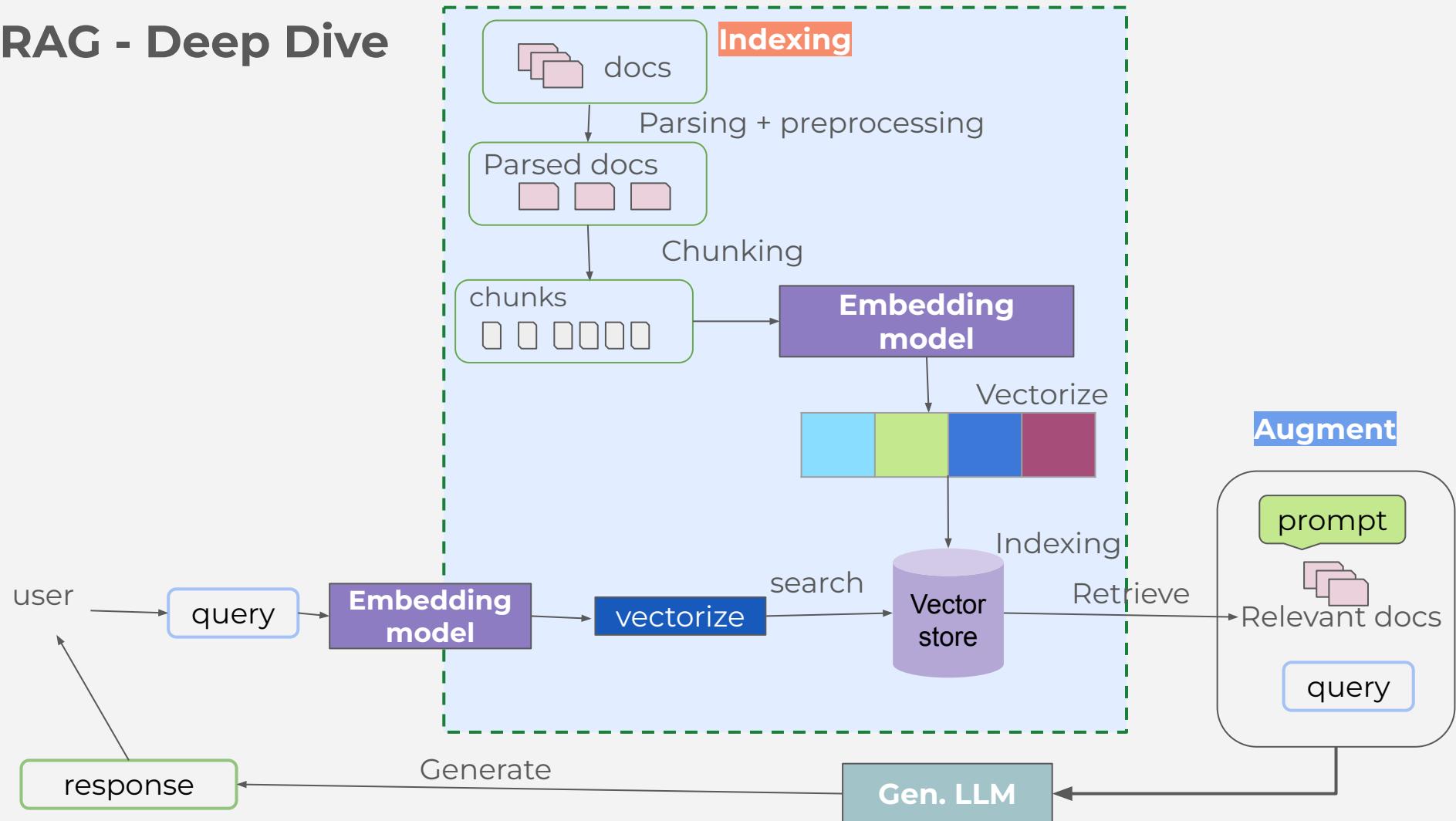
# RAG vs Pure GenAI Models



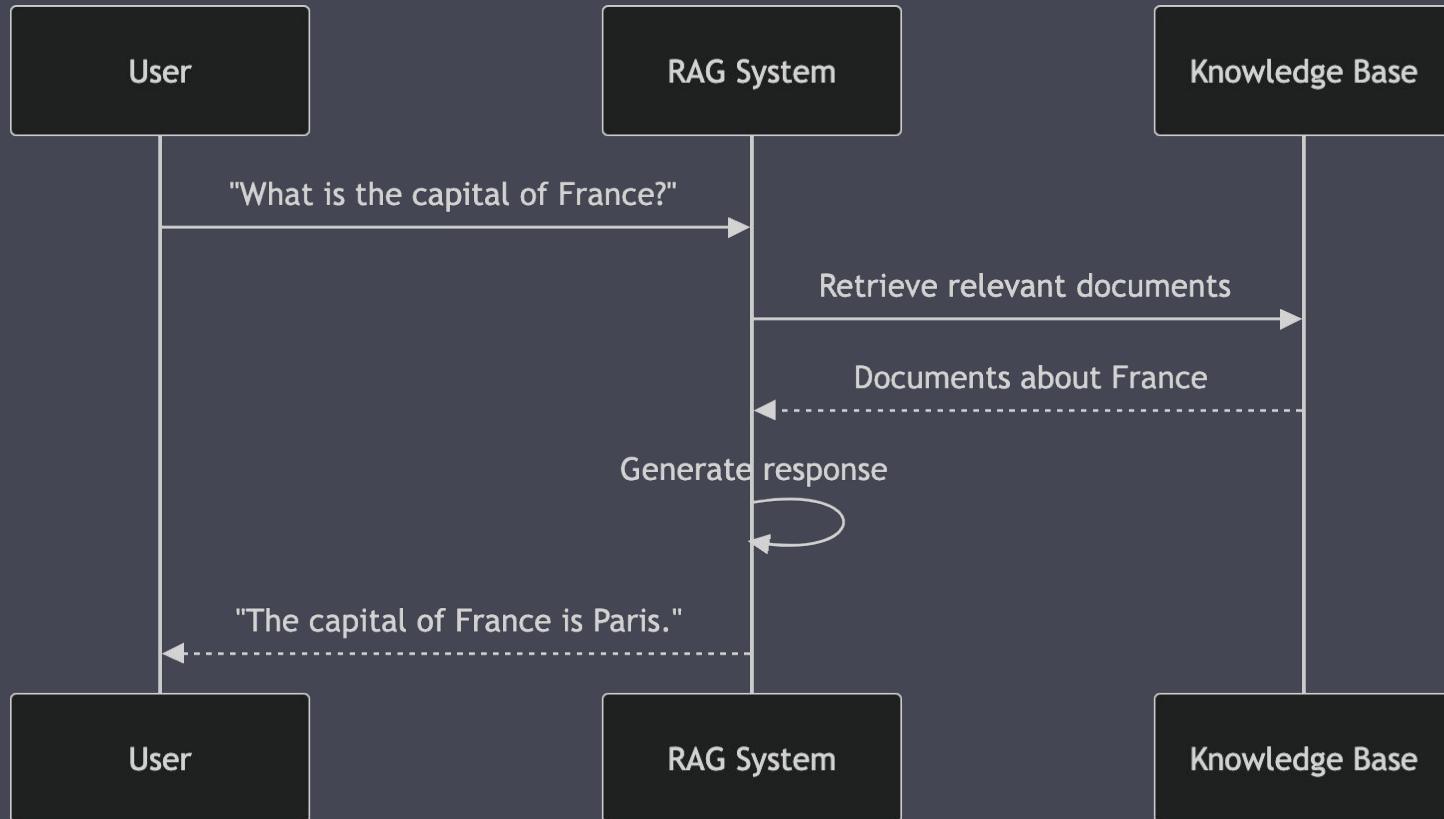
# RAG Overview



# RAG - Deep Dive

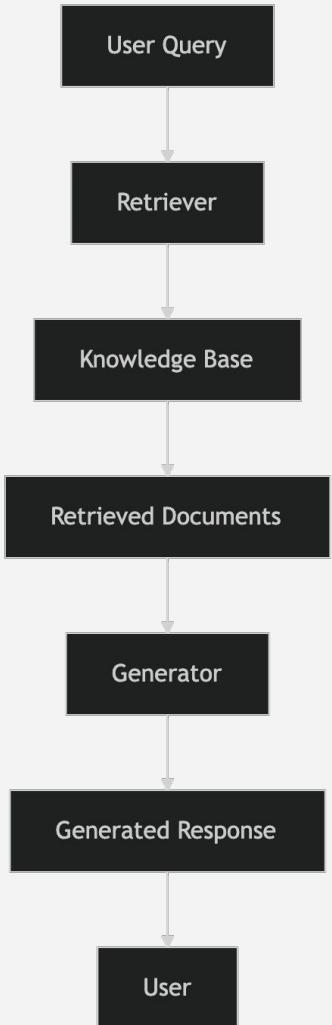


# RAG in Conversational AI



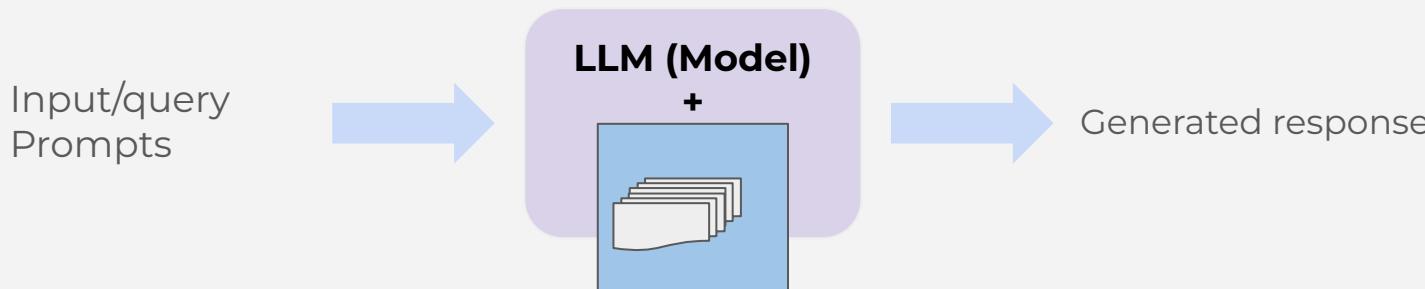
# Why RAG?

- **Enhanced Accuracy:** By retrieving factual information, RAG reduces the risk of generating incorrect or hallucinated responses.
- **Dynamic Knowledge:** Unlike static generative models, RAG can access up-to-date or domain-specific knowledge.
- **Scalability:** Can handle large knowledge bases efficiently.



# Practical Applications of RAG

- **Question answering systems:** Q&A systems that retrieve factual information from a knowledge base.
- **Search engine enhancement:** users get direct, well-formed answers derived from multiple sources
- **Document summarization:** synthesizes summaries on sections from specific document
- **Education and learning tools:** build systems to help students with questions related to relevant educational materials.



# Benefits of Retrieval-Augmented Generation

- **Enhanced factuality:** now we have up-to-date and factual information from external sources (no more solely relying on the pre-trained knowledge of the model).
- **Improved accuracy and specificity:** retrieves relevant context and grounds model's response in factual data
- **Reduced hallucination**
- **Adaptability:** no need for retraining the model anymore

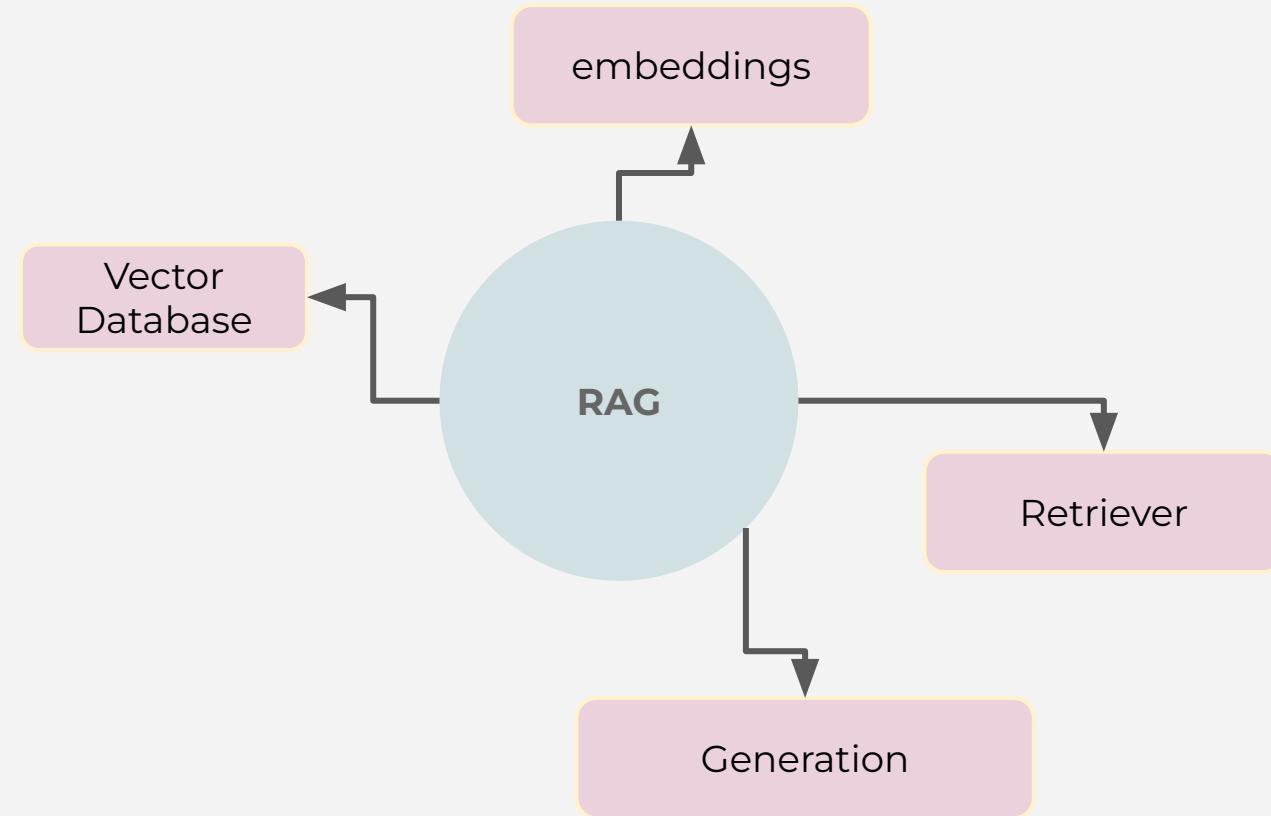
# Challenges of RAG

- **Retrieval Quality:** info quality depends on the relevance of retrieved docs.
- **Computational Cost:** retrieving and processing large corpus can be resource-intensive
- **Integration Complexity:** retrieval and generation requires careful design and tuning
- **Bias in Knowledge Base:** retrieved info may contain biases present in the knowledge base.

# Conclusion: takeaways

- RAG (Retrieval-Augmented Generation) is a powerful framework that:
  - Combines the strengths of retrieval-based methods and generative models.
  - Delivers accurate, context-aware, and dynamic responses.
- By leveraging:
  - Large knowledge bases (e.g., Wikipedia, custom corpora).
  - Advanced language models (e.g., GPT, BERT).
- RAG enables applications such as:
  - Question answering.
  - Document summarization.
  - Conversational AI (e.g., chatbots, virtual assistants).
- Challenges include:
  - Retrieval quality: Ensuring the retrieved information is relevant and accurate.
  - Computational cost: Managing resource-intensive retrieval and generation processes.
- Ongoing advancements in AI are:
  - Making RAG more efficient.
  - Increasing its accessibility for various use cases.

# Vector Database and Embeddings



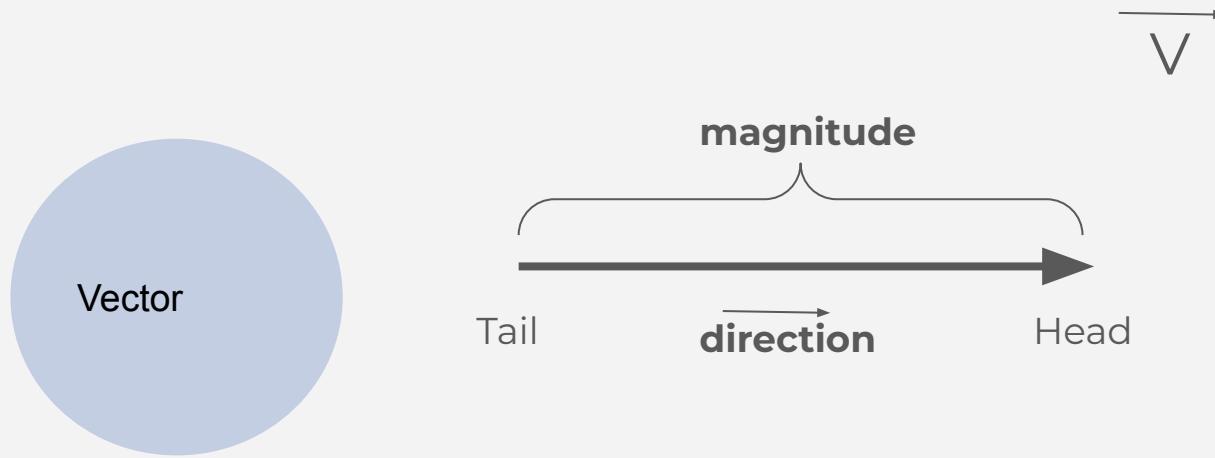
# What is a Vector Database?

A vector database encodes information as *vectors* in a multi-dimensional space to perform high-efficient queries based on similarity.

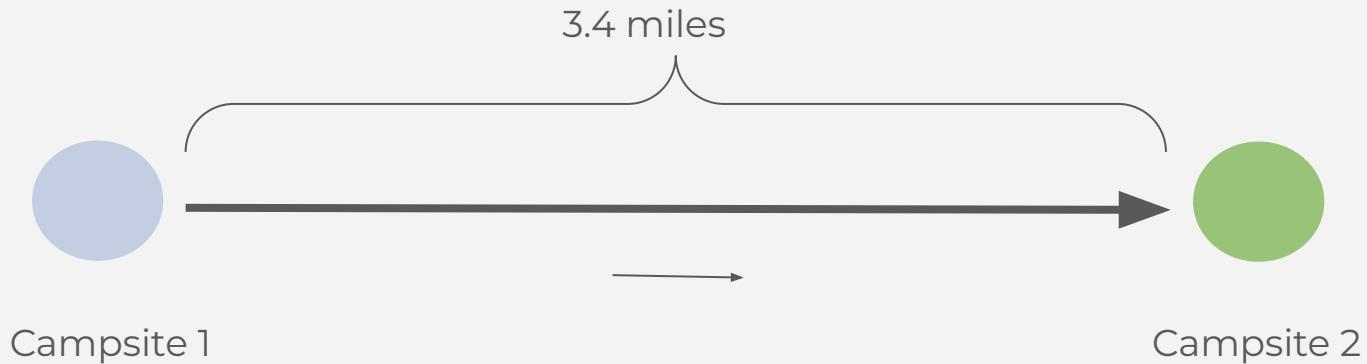


# What is a Vector Database?

A *vector* - what is it?



# Vectors



**Direction** - toward camp 2  
**Magnitude** - 3.4 miles

# Why Vector Databases?

How data “shows up” in the world?

80% or more of data is unstructured



**Vector databases** are specifically excellent for working with these types of data (actually, *the only type of databases that can work with unstructured data*)

Why?

# Why Are Vectors Used in a Vector Database?

Because...

Vector databases turn....



into

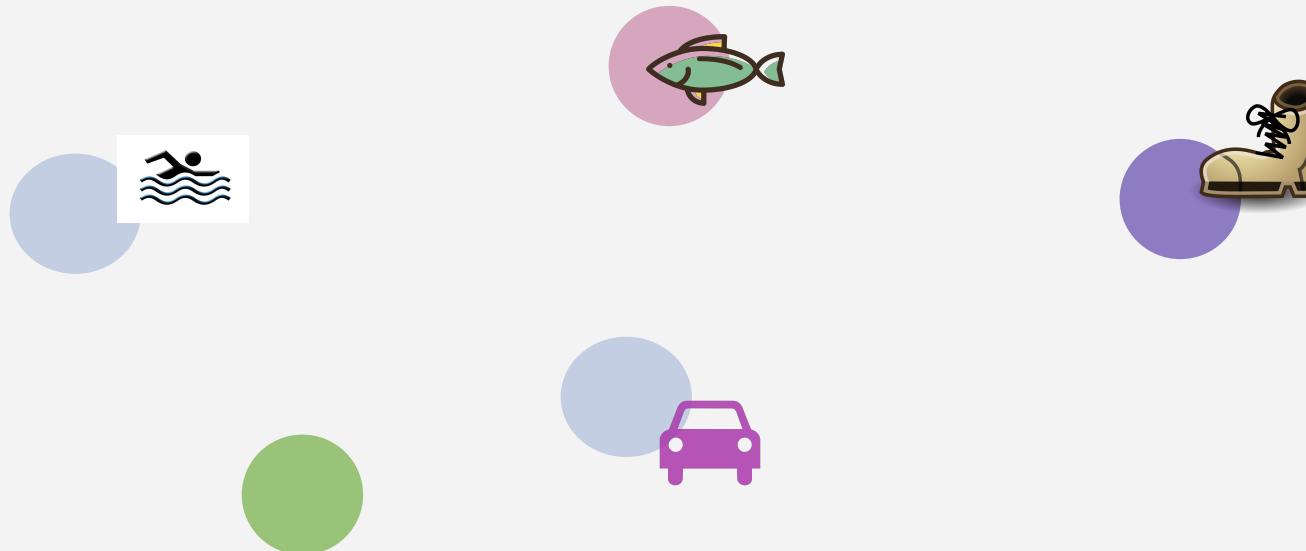
**Numbers (vectors)**

-0.05, -0.0955, ..., 0.0722

-0.053, -0.885, 0.1622, ...

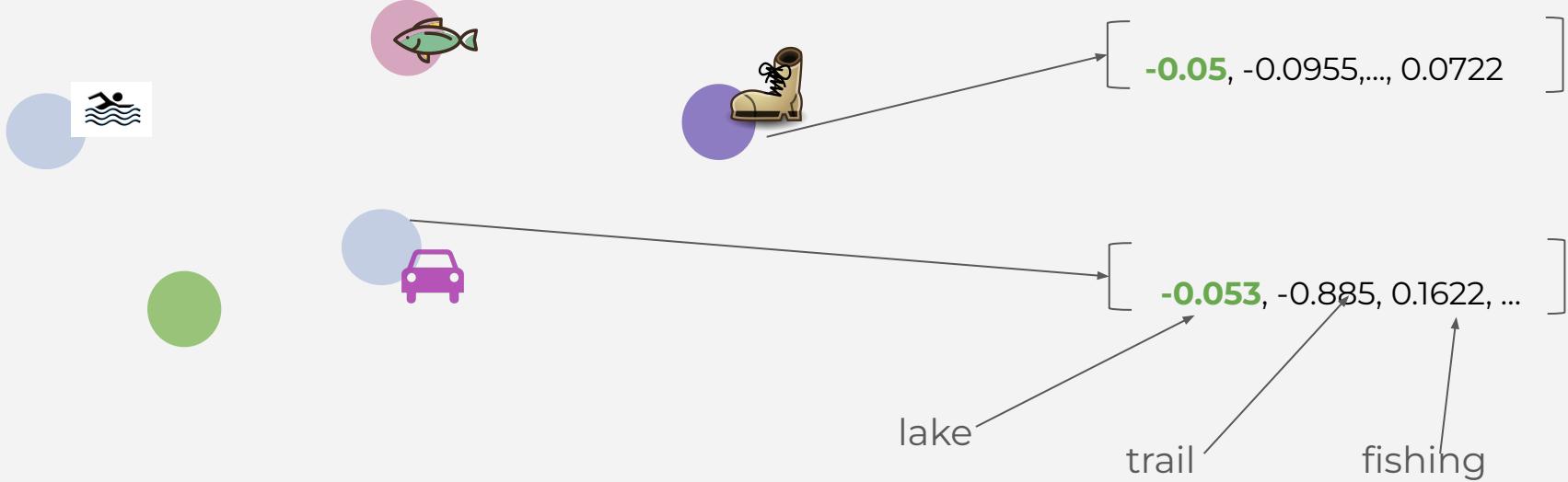
# Why Are Vectors Used in a Vector Database?

We are back to the first question... and the campground

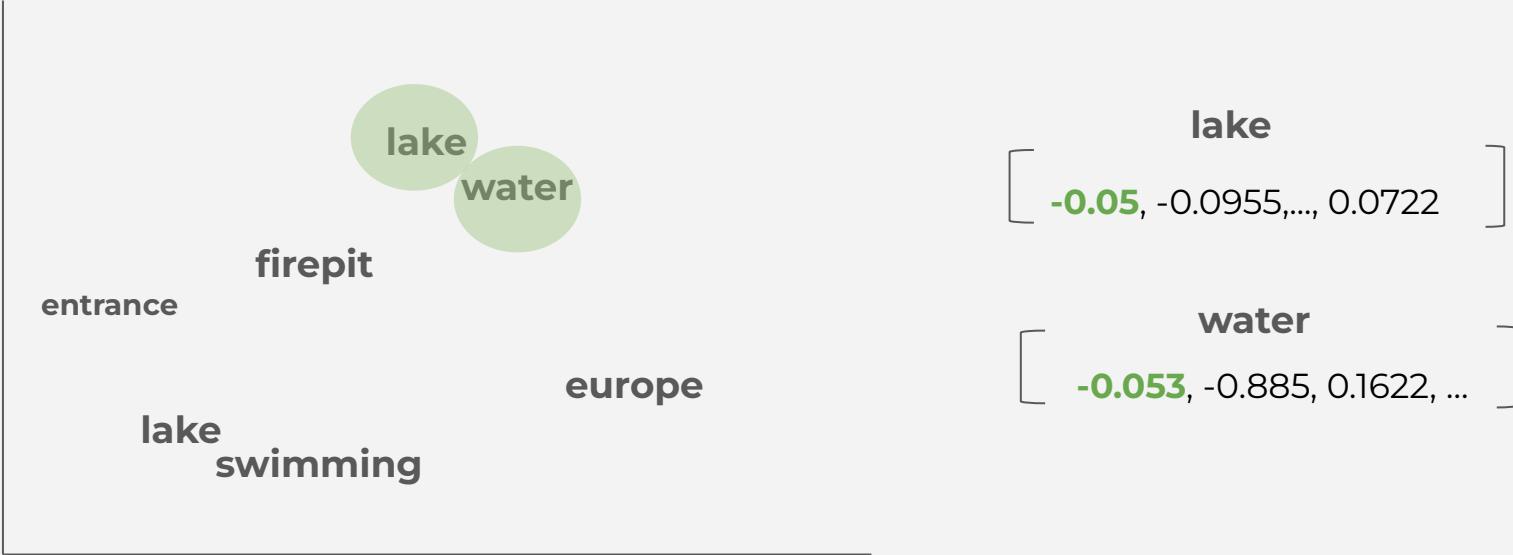


# Vectors in Action

Campsite as vectors



# Benefits - Quick Search for the Best Campsite



Rapid discoverability  
Efficient organization

# Why are Vectors Used in a Vector Database?

## 1. Efficient Representation of Complex Data

- a. Dimensionality - representing data in high-dimensional space
- b. Uniformity - data can be converted into a uniform format (numerical vectors)

## 2. Enabling Similarity Search

## 3. Leveraging Machine Learning Models

## 4. Optimizing Performance and Scalability

## 5. Improving User Experience

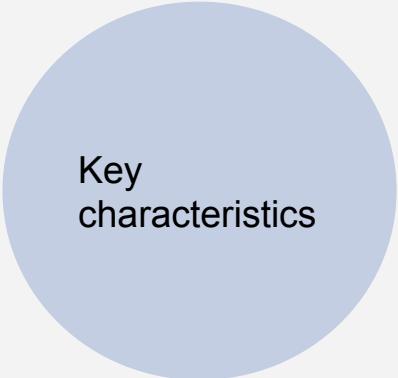
- a. Real-time interaction (recommendations, search results or data analysis outputs)

lake  
[ -0.05, -0.0955,..., 0.0722 ]

water  
[ -0.053, -0.885, 0.1622, ... ]

# Traditional Databases

## In contrast to Vector Databases - RDBMS

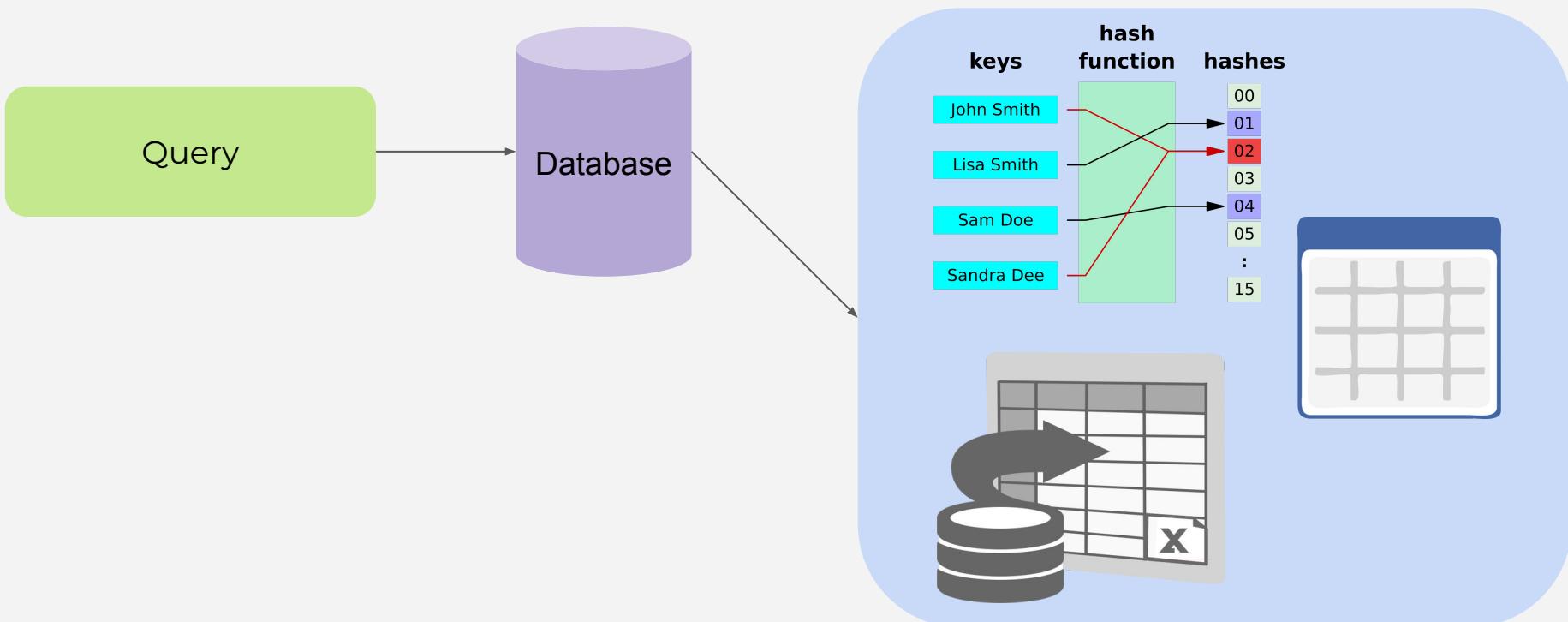


Key  
characteristics

- **Structured** data: predefined columns and rows
- **Schema-based**: database structure must be defined before hand.
- **Data manipulation and querying**: manipulation through SQL
- **ACID Compliant**: Atomic, Consistency, Isolation, Durability
- **Indexing**: to speed up data retrieval

# Traditional Databases

How data search works in traditional databases:

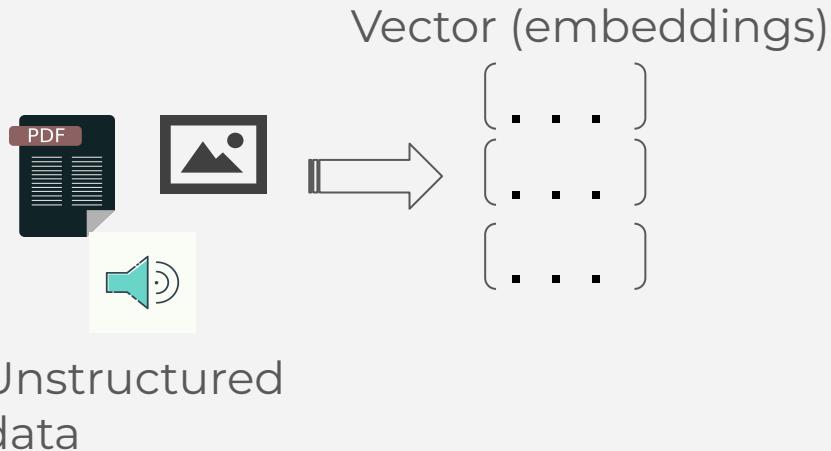


# Traditional Databases

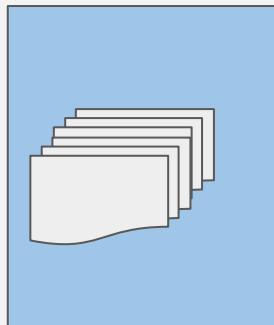
## Limitations

- **Scalability:** hard to deal with complex queries across large tables
- **Flexibility:** changing DB schema can be disruptive
- **Handling Unstructured Data:** not well-suited for handling unstructured data (images, text, audio, video)

# Transforming Unstructured Data into Vectors - Deep dive



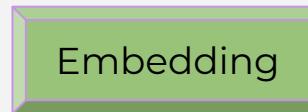
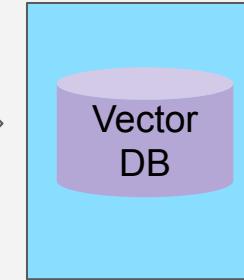
## Splitting



Embedding



## Storage



$$[-0.00543, -0.0984055, \dots, 0.075622]$$

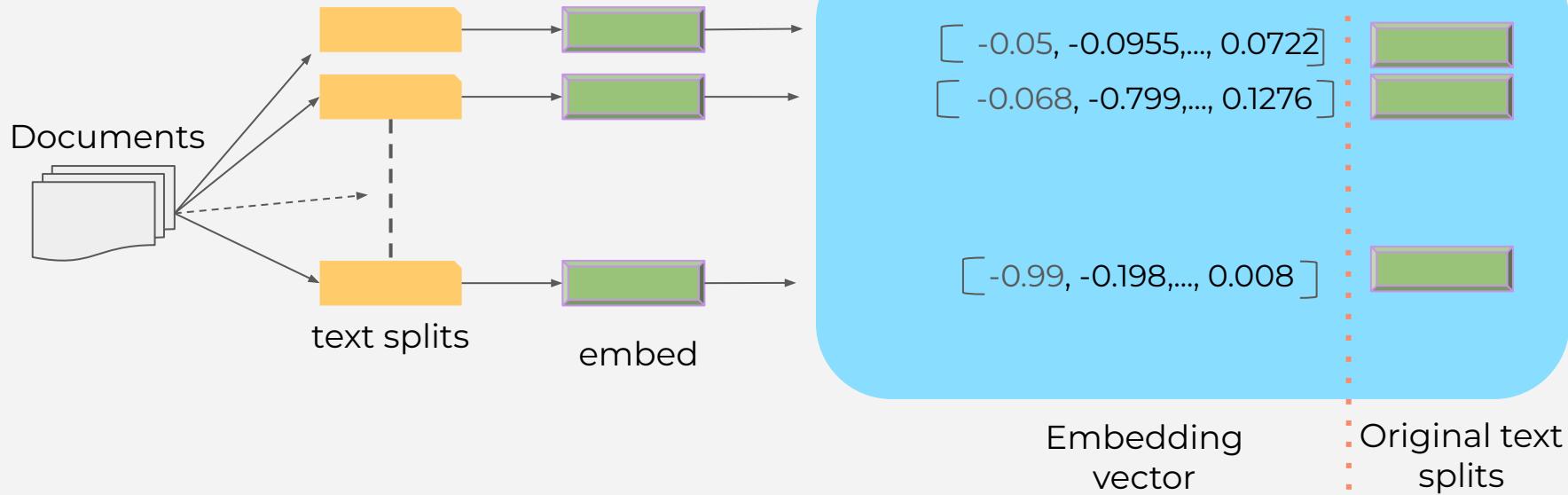
Embedding vector keeps the content and the meaning of the text  
Text with similar content and meaning will have similar vectors

**Text with similar content and meaning will have similar vectors**

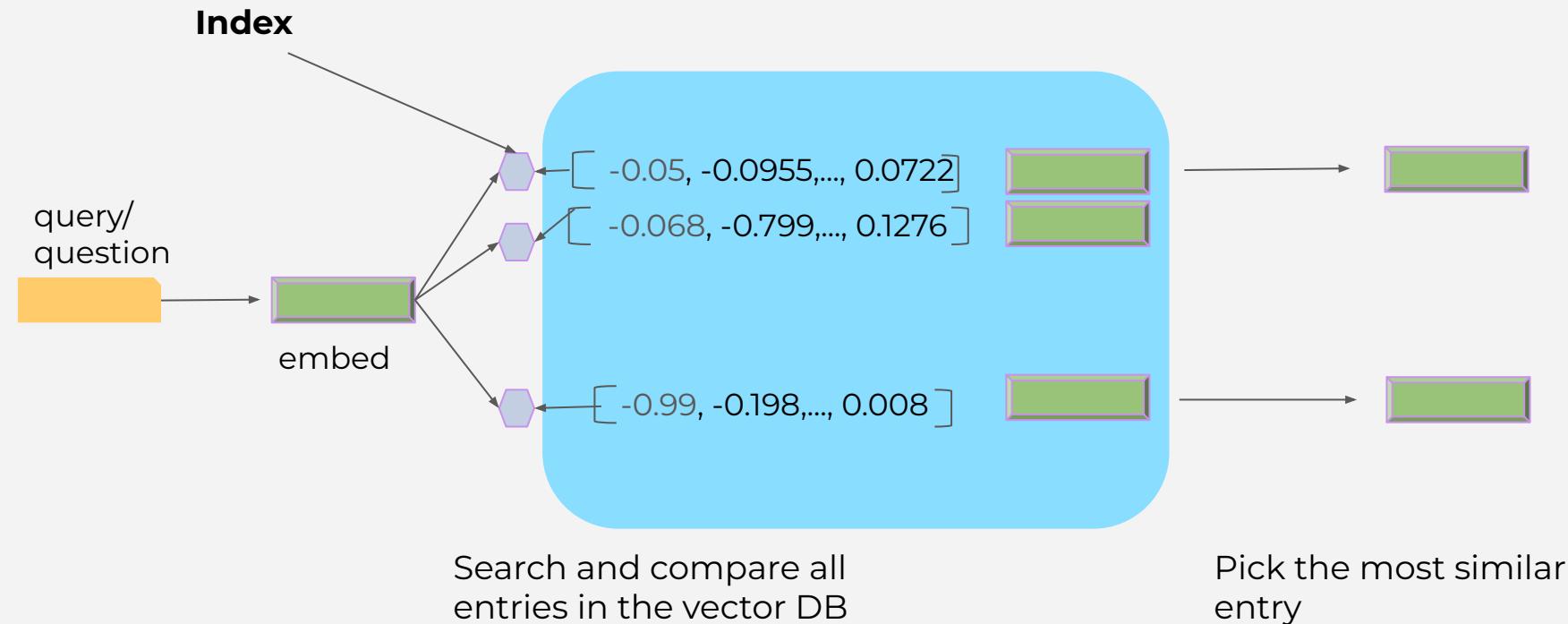


# Vector database (vectorstore) - full overview

## Embedding creation and storage



# Vector Database - querying the vector database



# Vector Store - Processing with LLM



Pick the most similar entry

Take the most similar entry found and pass it on to the LLM along with the question/query...

# Embeddings vs vectors

They essentially refer to the same thing... BUT they have distinct definitions and roles.



Mathematical representation of data in an n-dimensional space ( each dimension == a feature of the data)

## Embeddings

A specific type of vector used in Machine Learning and Artificial Intelligence

# Embeddings vs vectors

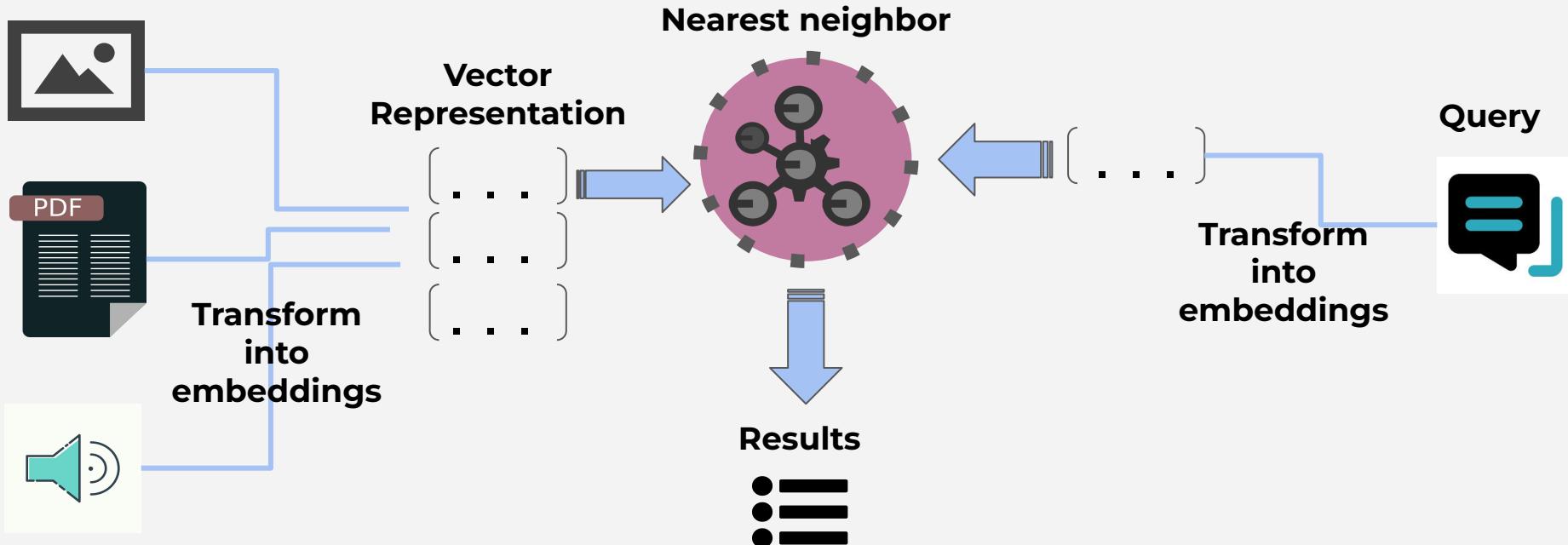
## Key differences and relation

**Vectors** are generic and used for a wide array of applications (handle numerical computations...)

**Embeddings** map raw data into a vector space (**preserves** semantic relationships - **meaning**)

**Bottomline:** embeddings are **vectors**, but not all vectors are embeddings. Embeddings are vectors that encodes semantic similarities between the items they represent (great for AI applications)

# Vector databases - how they work



# Vector Databases

## Advantages

- **Data Representation as Vectors:** vector is vectorized which brings lots of benefits for searching
- **Similarity Search:** finding data points closest to a given query vector.
- **Efficiency in High-Dimensional Searches:** use of specialized indexing structures that are highly optimized
- **Handling Unstructured Data:** vector database are made to deal with unstructured data!
- **Schema-less Design:** don't require schema - allowing more flexibility in handling various data types and structures

# Vector Databases Use cases



Powerful for handling complex queries

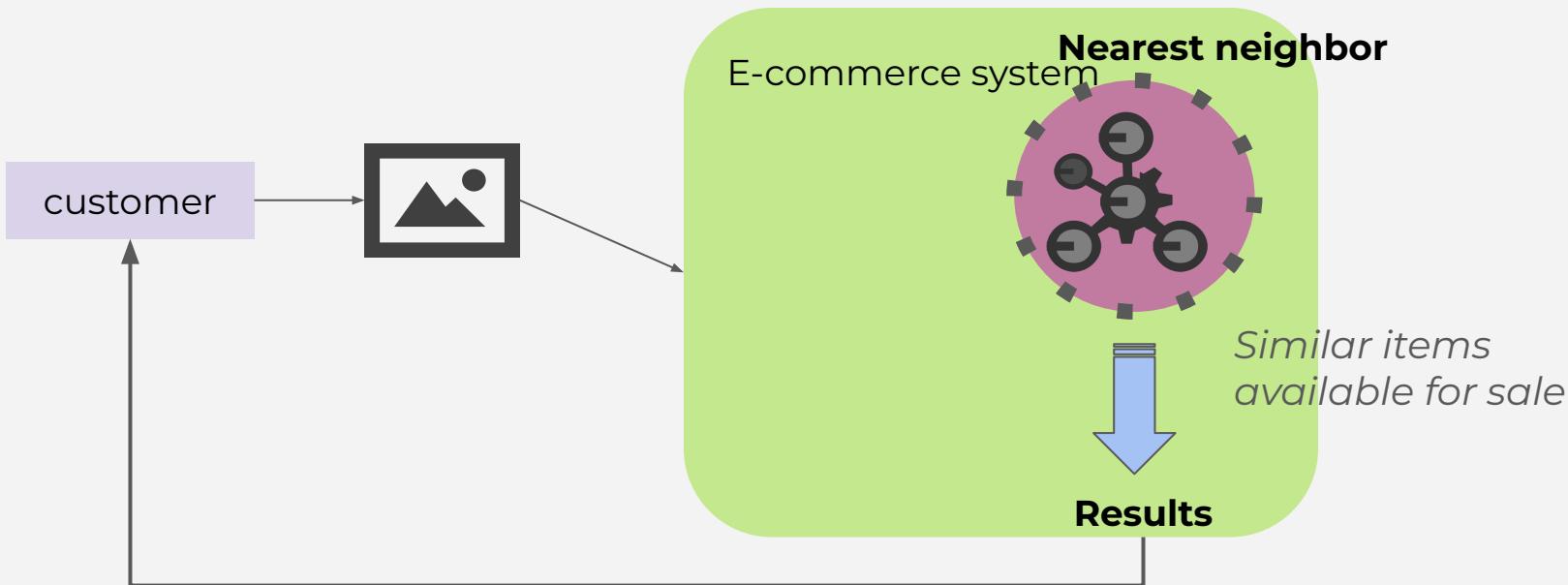
Here a 5 use cases:

1. Image Retrieval & Similarity Search
2. Recommendation Systems
3. Natural Language Processing (NLP)
4. Fraud Detection
5. Bioinformatics

# Image Retrieval & Similarity Search

For example: ***an e-commerce platform***

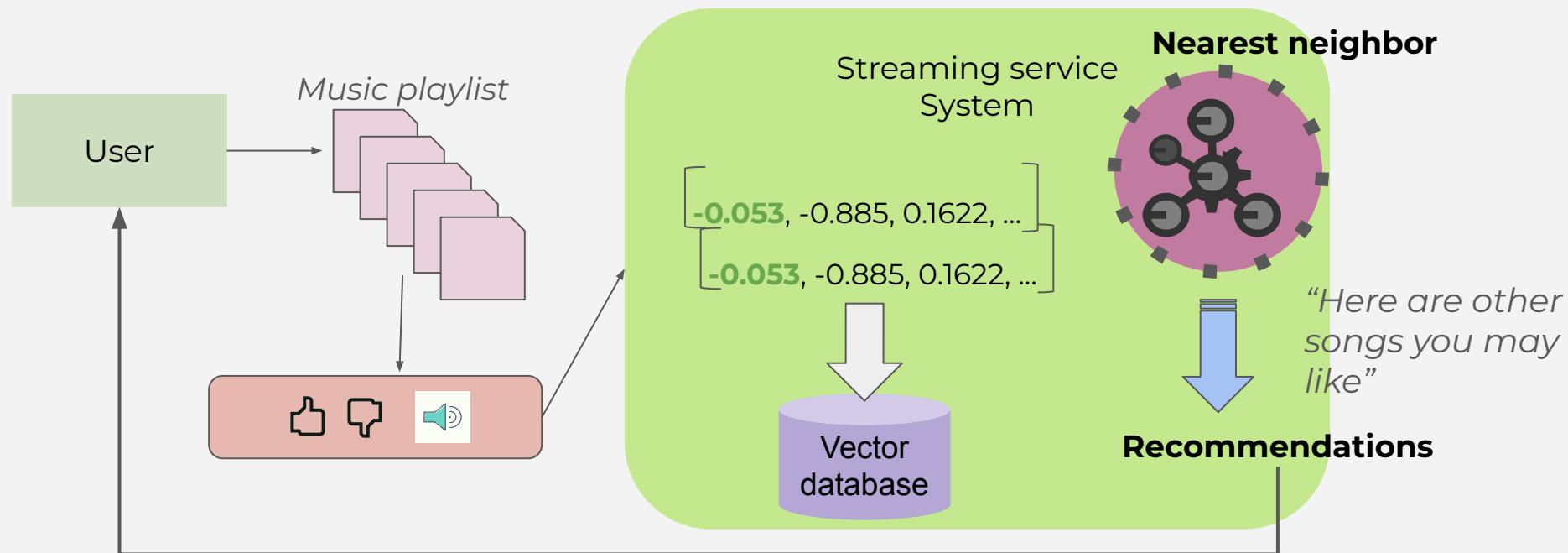
Can use a vector database for visual search feature.



# Recommendation Systems

For example: ***music streaming service***

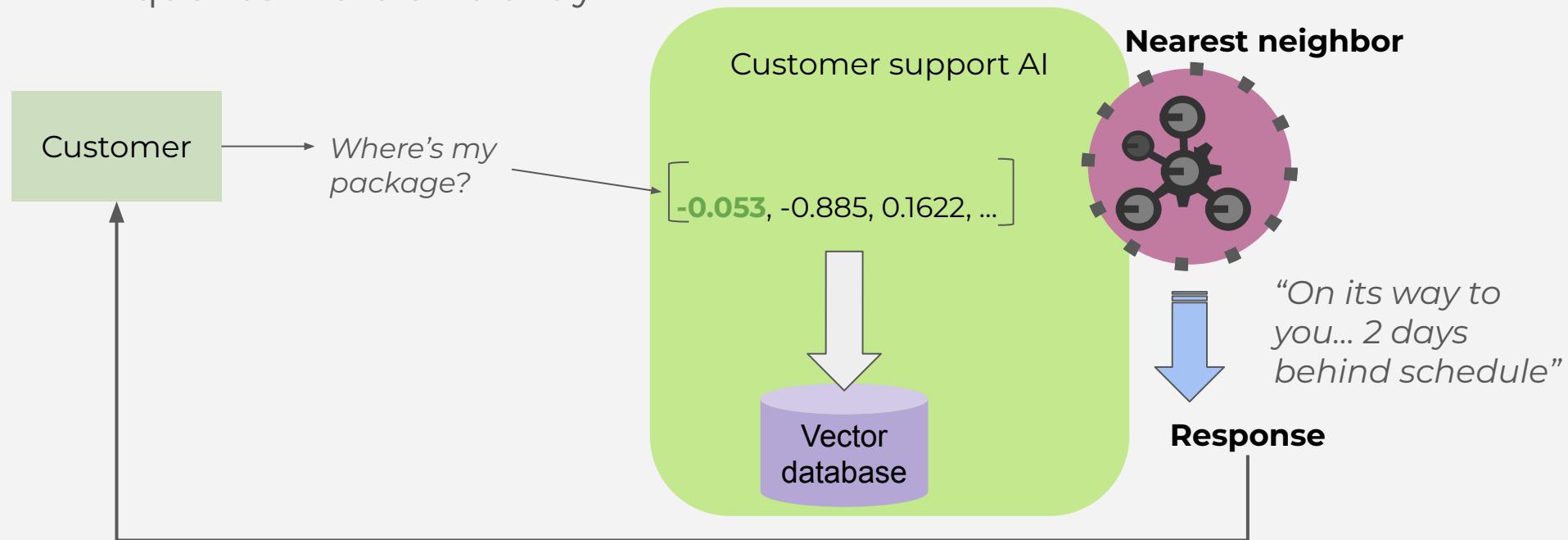
Uses a vector database to recommend songs



# NLP - Natural Language Processing

For example: ***customer support AI***

Uses a vector database to understand and respond to user queries more efficiently



# Fraud Detection & Bioinformatics

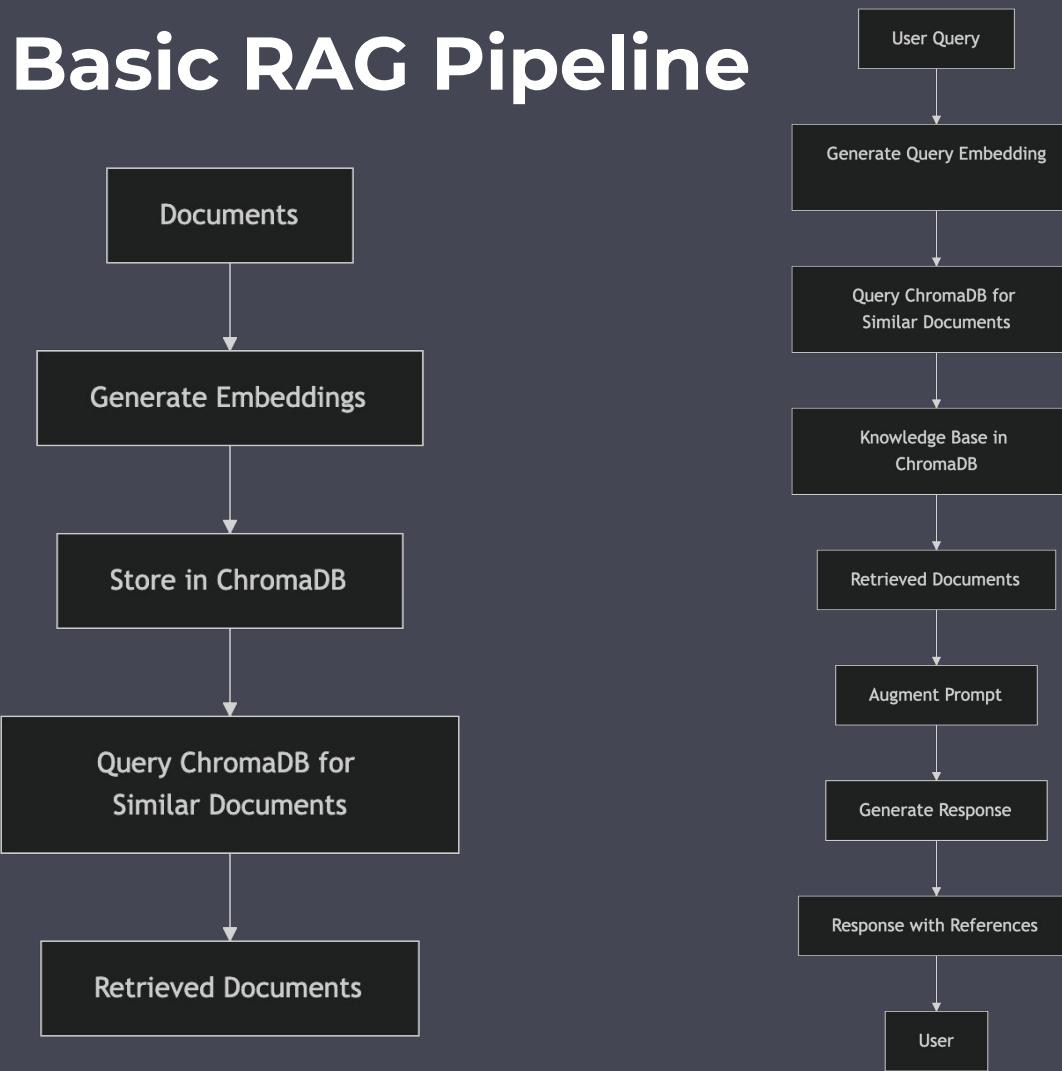
For example: ***detect fraudulent activities***

Uses a vector database to quickly compare user behavior patterns and flag anomalies...

For example: ***compare genetic sequences***

Uses a vector database to compare gene expression profiles from different patient samples.

# Hands-on Basic RAG Pipeline



# **LangChain - Building LLM-powered Applications**

# LLM Frameworks

## Why Use LLM Frameworks?

LLMs are **challenging** especially when it come to integrating them into applications without the right tools.

LangChain

LlamaIndex

Haystack



Simplify it all by providing:

- Abstractions
- Tools
- workflows

# LLM Frameworks

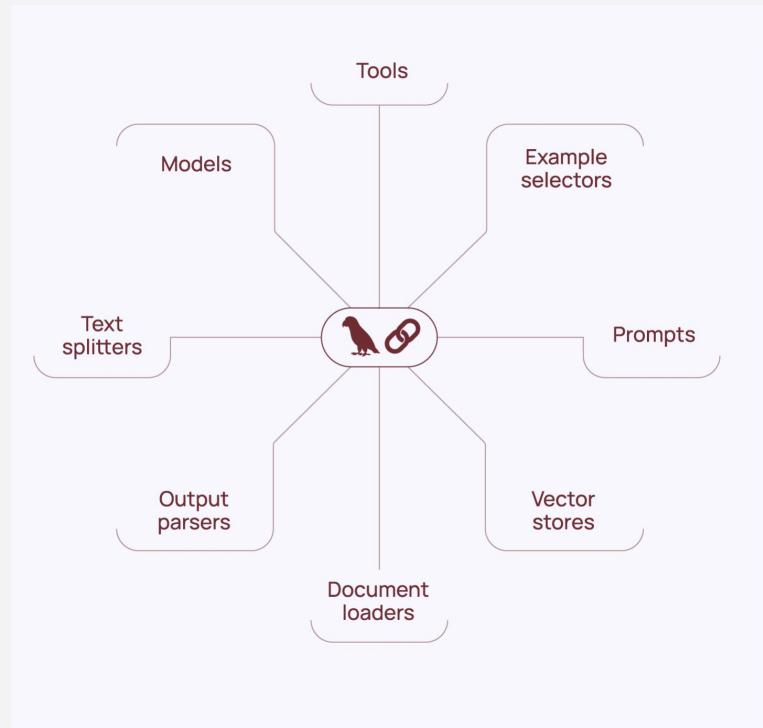
## Key motivation for using LLM Frameworks

- Simplify integration
  - Developers only focus on building applications rather than managing intricacies.
- Enable modularity
  - Application is broken down into reusable components (easier to iterate and scale)
- Support context management
  - LLMs require context to generate meaningful responses
    - Frameworks provide memory system to store context and metadata
- Facilitate data integration
  - Connect to APIs, datasources...
- Enable autonomous agents, reduce development time...

# LangChain

A framework (*open source*) for building applications that leverage various LLMs (Large Language Models).

Additionally - you can also use **external sources of data** combined with various LLMs!



# Key Components (LangChain)

Models

Prompts

Chains

Memory

Tools

Agents

# Key Components (LangChain)

Models



Wrapper for various models (LLMs)

- OpenAI's GPT4 and many more
- Embedding models
- Chat models

# Key Components (LangChain)

Prompts



Wrapper for various prompts:

- Reusable prompt templates (for generating input for LLMs)
- Few-shot learning: providing examples in the prompt to guide the model's behavior

# Key Components (LangChain)

## Chains



Wrapper for various chains:

- **Sequential chains** - combine multiple components in a sequence
- **Custom chains**: building custom workflows

# Key Components (LangChain)

## Memory



Wrapper for various memory modules:

- **Short-term memory** - storing context within a single interaction
- **Long-term memory**: persisting context across multiple interactions

# Key Components (LangChain)



- **External APIs** - integration
- **Databases**: querying databases for structured data

# Key Components (LangChain)



- **Autonomous systems:** building agents that use LLMs for decision-making and task execution

# **Key Components (LangChain)**

Hands - on: Langchain website walkthrough and how to install it locally and start to use it...

**Hands-on: Building an LLM-Powered Application  
with LangChain**

# Other LLM frameworks

There are many LLM frameworks (and more being released):

- **LangChain** - good for general-purpose LLM applications  
(memory, agents, modular workflows)
- **LlamaIndex** - retrieval-augmented generation (RAG), and  
workflows that require querying structured and  
unstructured data.
- **Haystack** - good for search and question-answer systems  
(advanced document retrieval and preprocessing)

# **Building Vector Databases**

Hands-on - Building vector databases from scratch

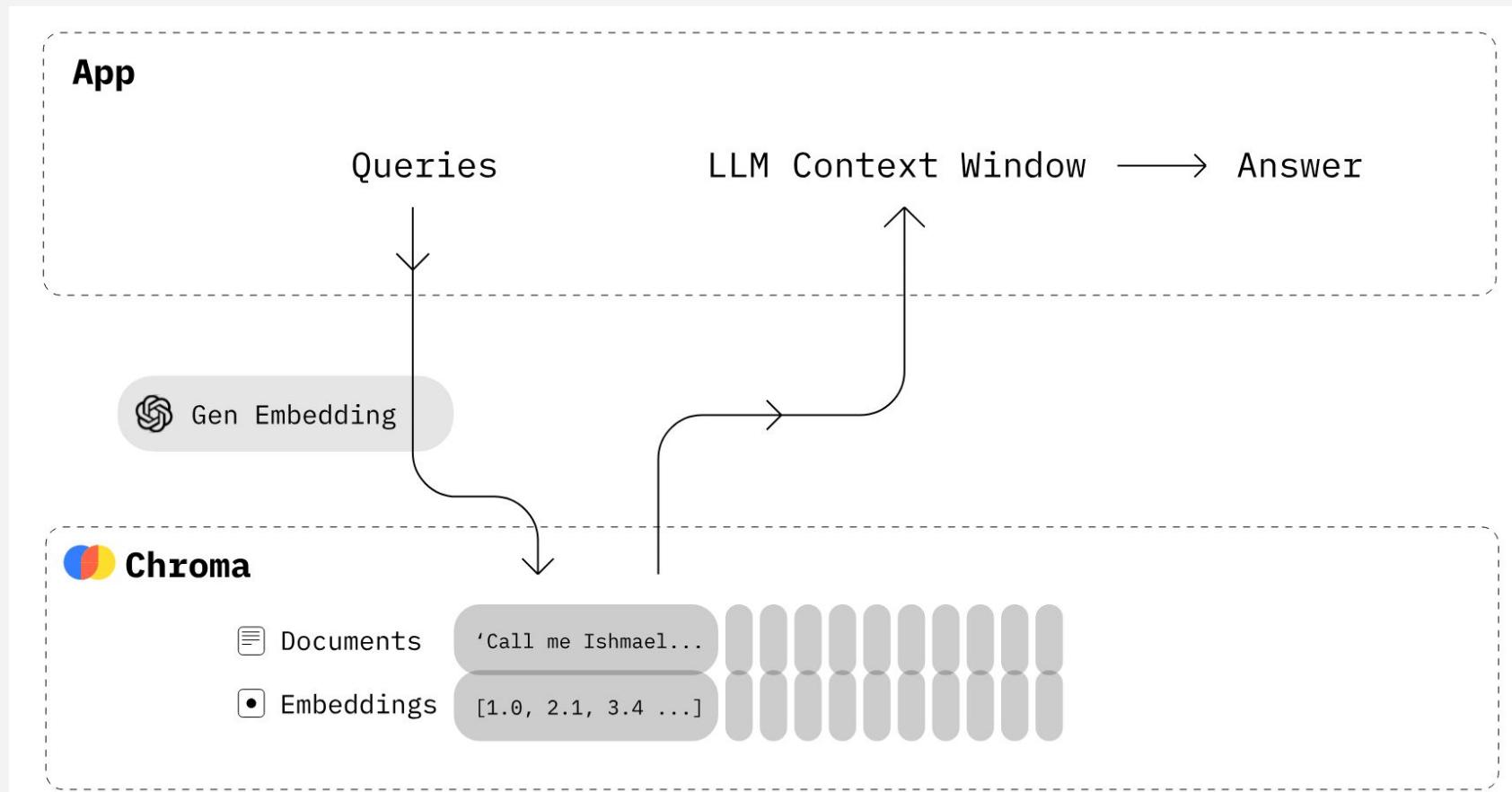
# Development Environment Set up

- VS Code
- Python
- OpenAI account and API Key

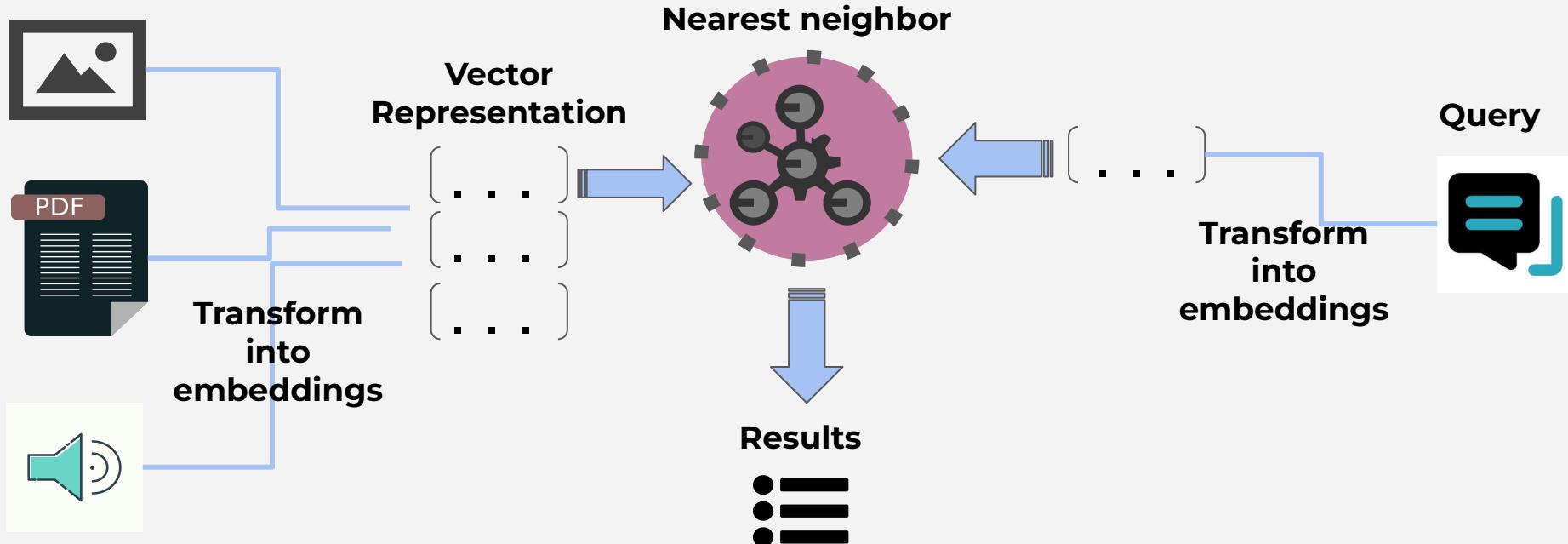
# **Hands-on - Create a Vector DB using Chroma**

**Vector database with Chroma**

# Chroma database workflow



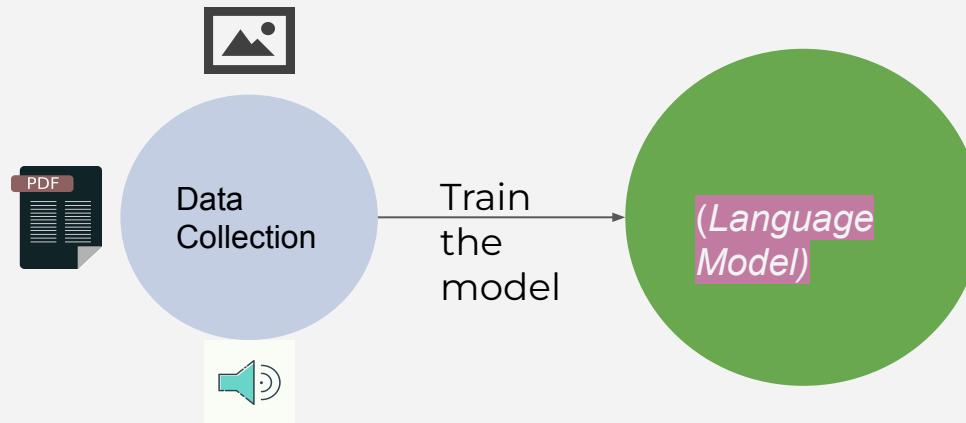
# Vector Search - How Does It Work?



# Chroma database - OpenAI Integration

## Understanding Large Language Models (LLMs)

### Data Collection and Training...



Collection of lots of data & train the model: learn patterns in human language (*articles, studies, news...*)

# How Does it Work?

VectorStore holds **embeddings - Vector** representation of the text



But why **embeddings**?

Because we can easily do search where we look for pieces of text that are most similar in the vector space

- Capture semantic meaning
- Enabling similarity measures
- Handling high-dimension data

## Vector Stores

### 1. Load Source Data



Load, Transform, Embed

### Vector Store

0.5, 0.2...0.1, 0.9  
:  
2.1, 0.1...-1.7, 0.9

### Embed

5.5, -0.3...  
2.1, 0.1

### 2. Query Vector Store

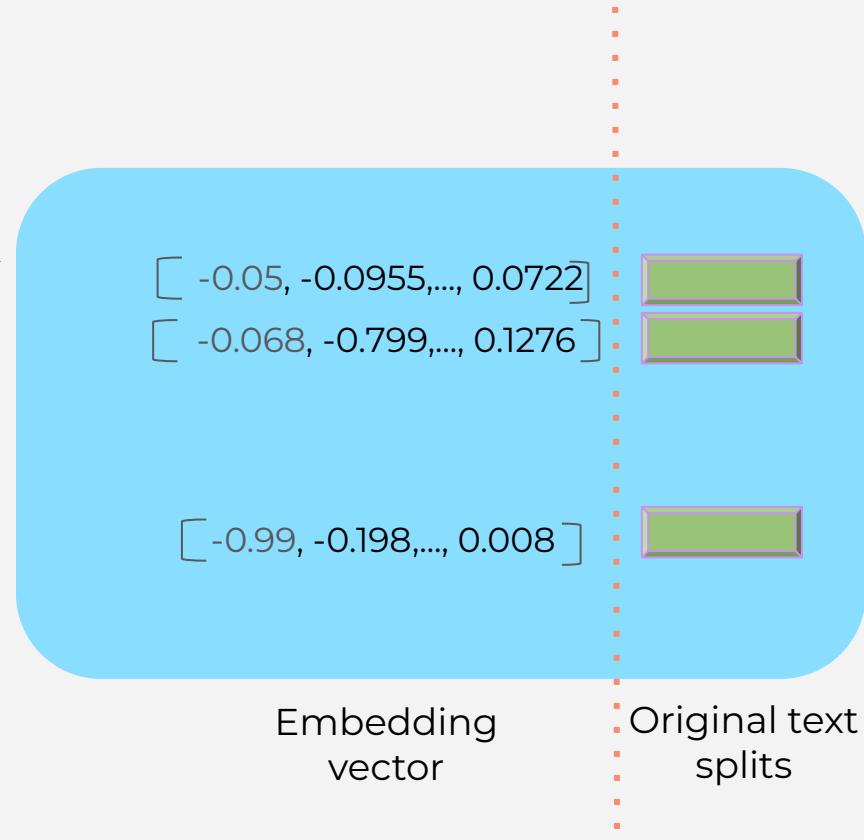
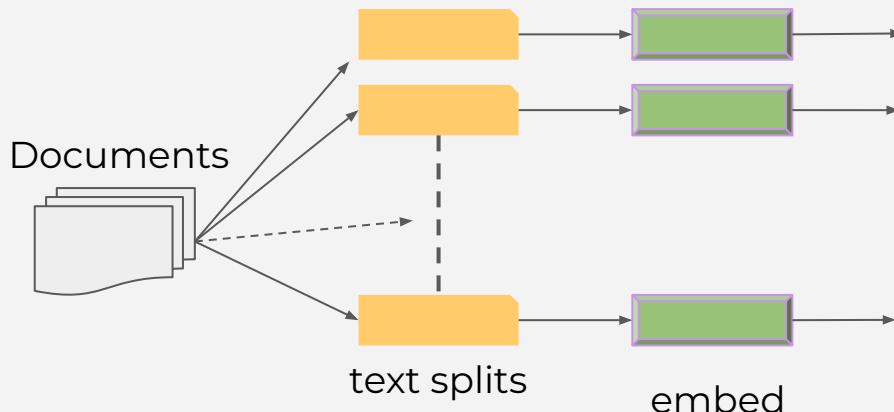
XXXXXXXXXXXXXX  
XXXXXXXXXXXXXX

XXXXXXXXXXXXXX  
XXXXXXXXXXXXXX

### 3. Retrieve 'most similar'

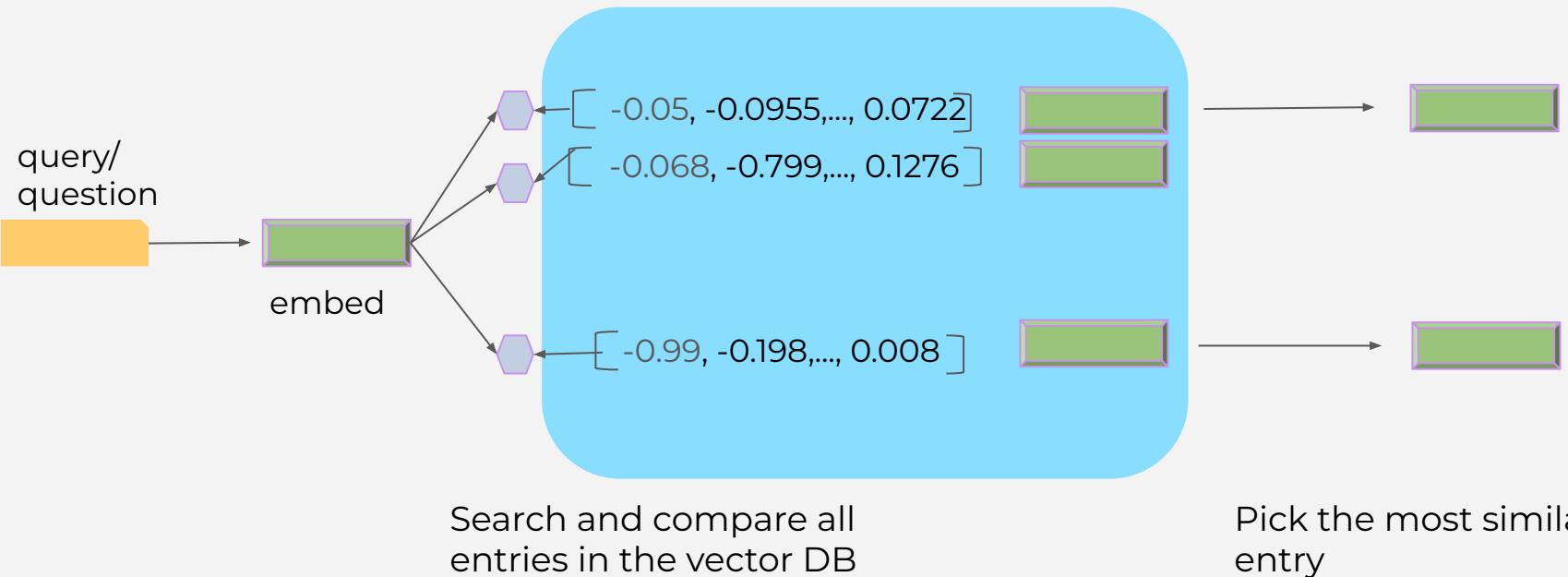
# Vector database - full overview

## Embedding creation and storage

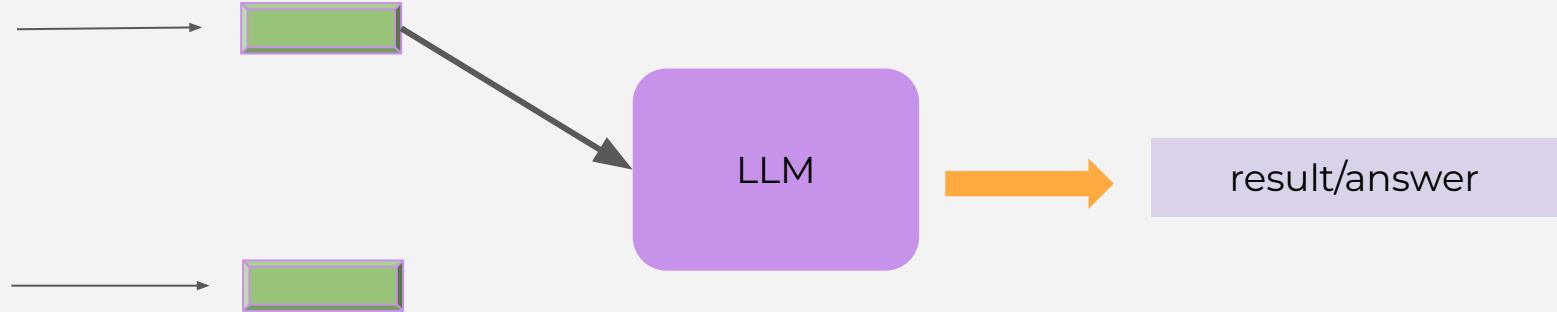


# Vector Database - querying the vector database

## Index



# Vector Store - Processing with LLM



Pick the most similar entry

Take the most similar entry found and pass it on to the LLM along with the question/query...

# Indexes, Retrievers & Data Preparation

Data Preparation

Loading, preprocessing and structuring.

Indexes

Data structures that organize documents for efficient retrieval.

Retrievers

Finders of relevant documents based on query

# Data Preparation

Data Preparation

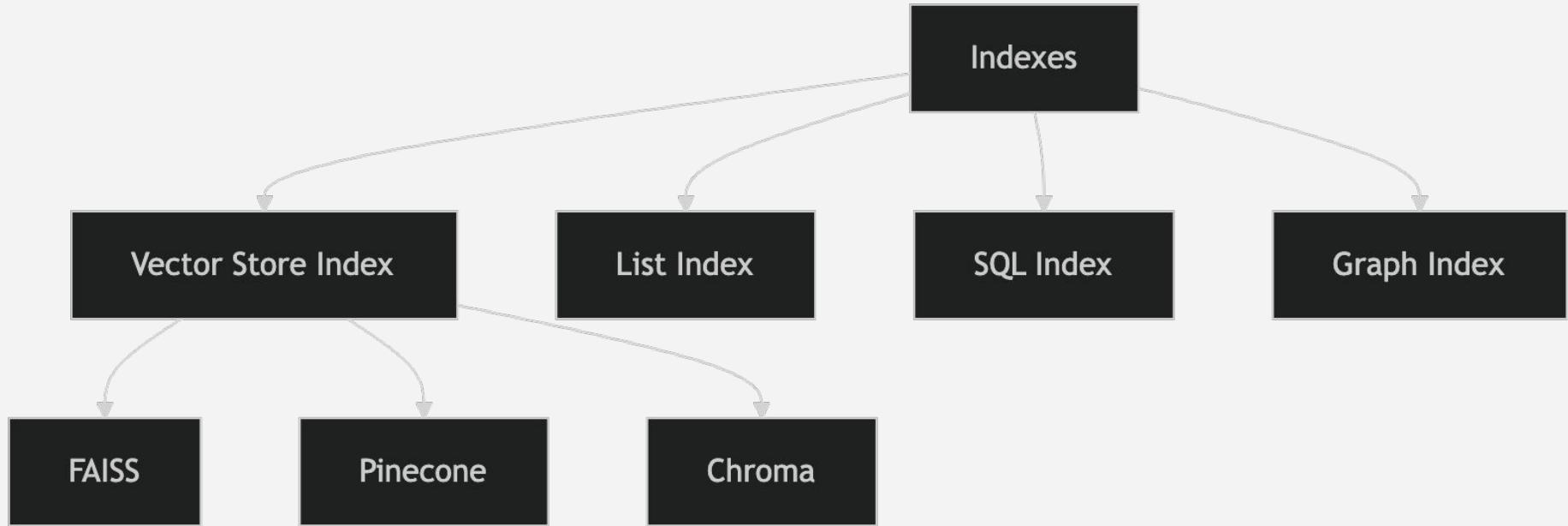
Loading, preprocessing and structuring.



# Indexes

Indexes

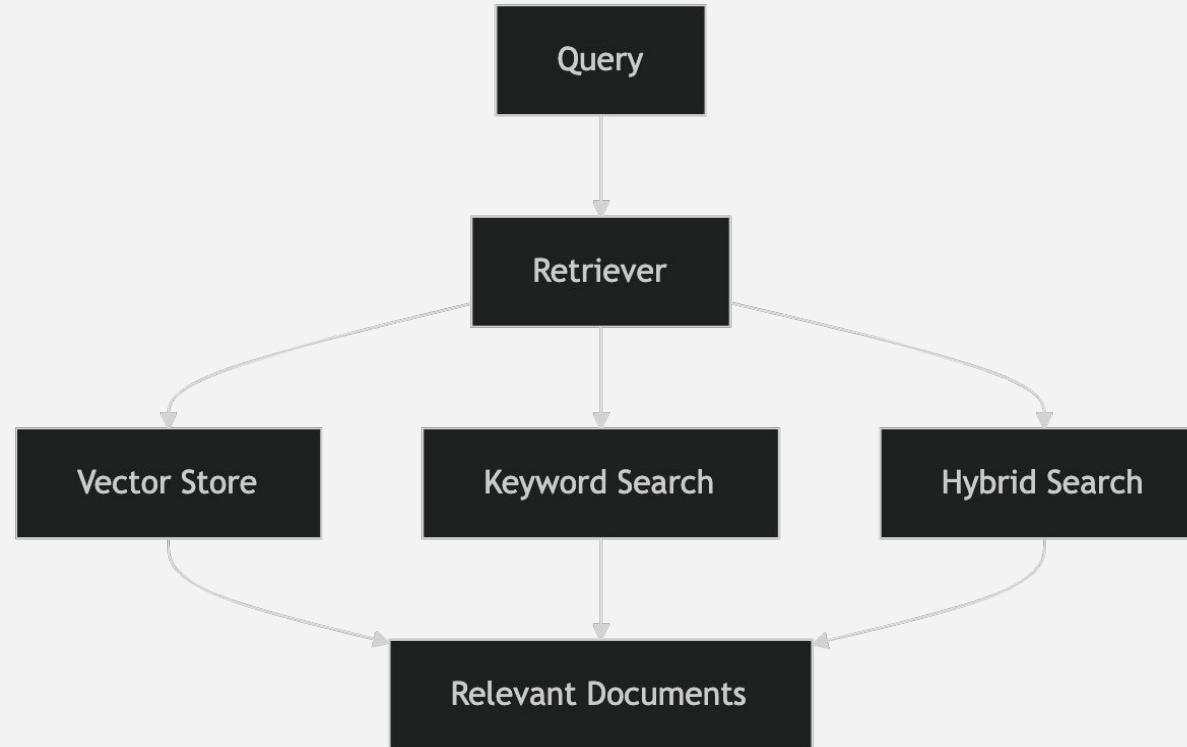
Data structures that organize documents for efficient retrieval.



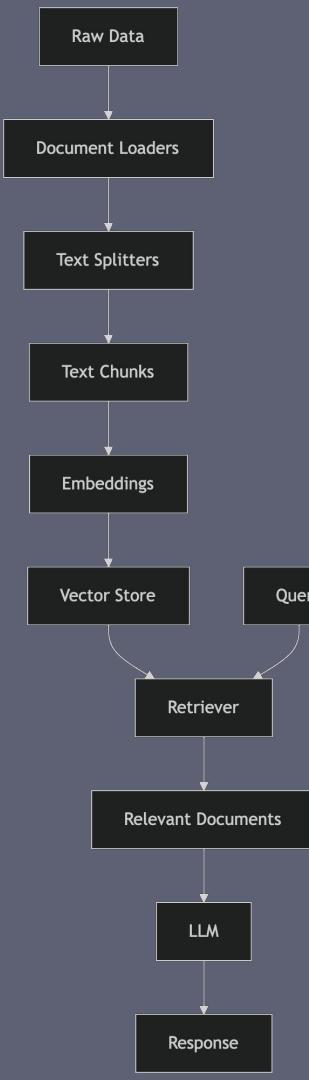
# Retrievers

Retrievers

Finders of relevant documents based on query



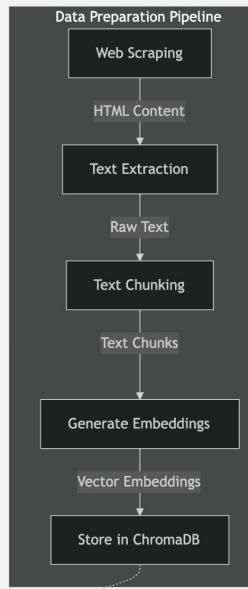
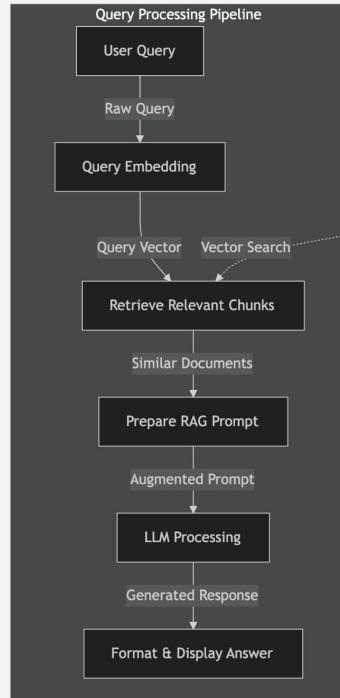
# Full workflow



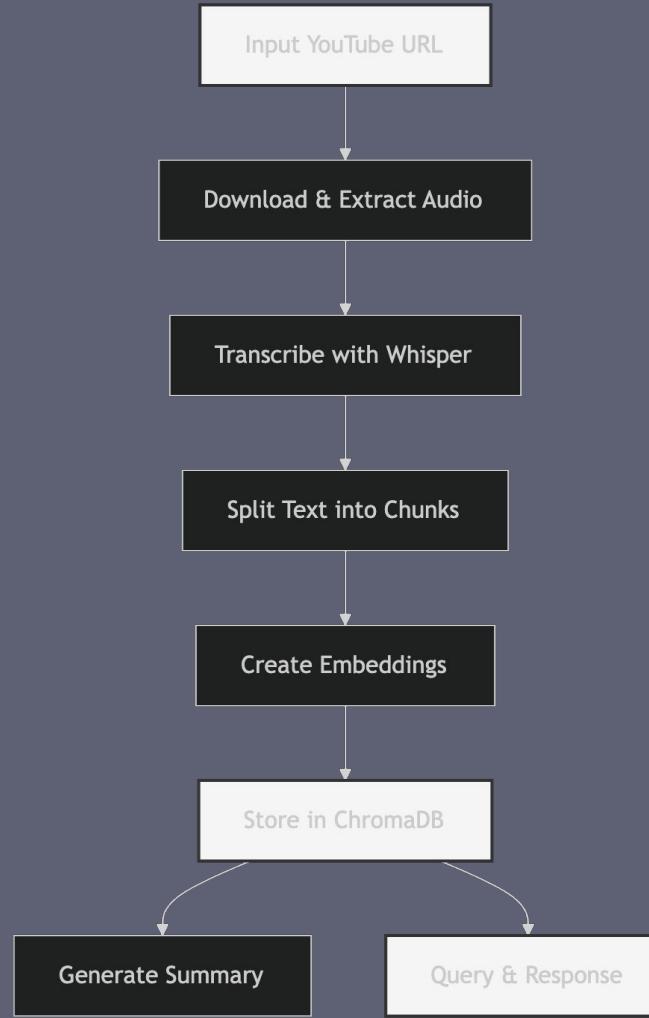
# Complete integration Example

Document search system

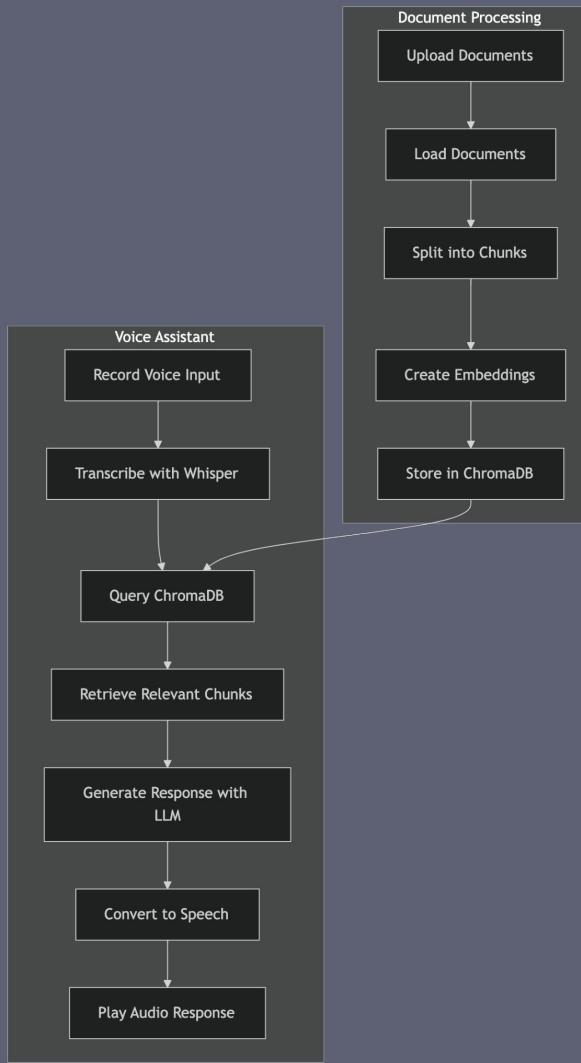
# Hands-on - Q&A Bot (RAG workflow)



# Hands-on - Youtube Video Summarizer



# Hands-on - Voice assistant



# Naive RAG

## Indexing

Cleaning,  
extracting data  
from docs...

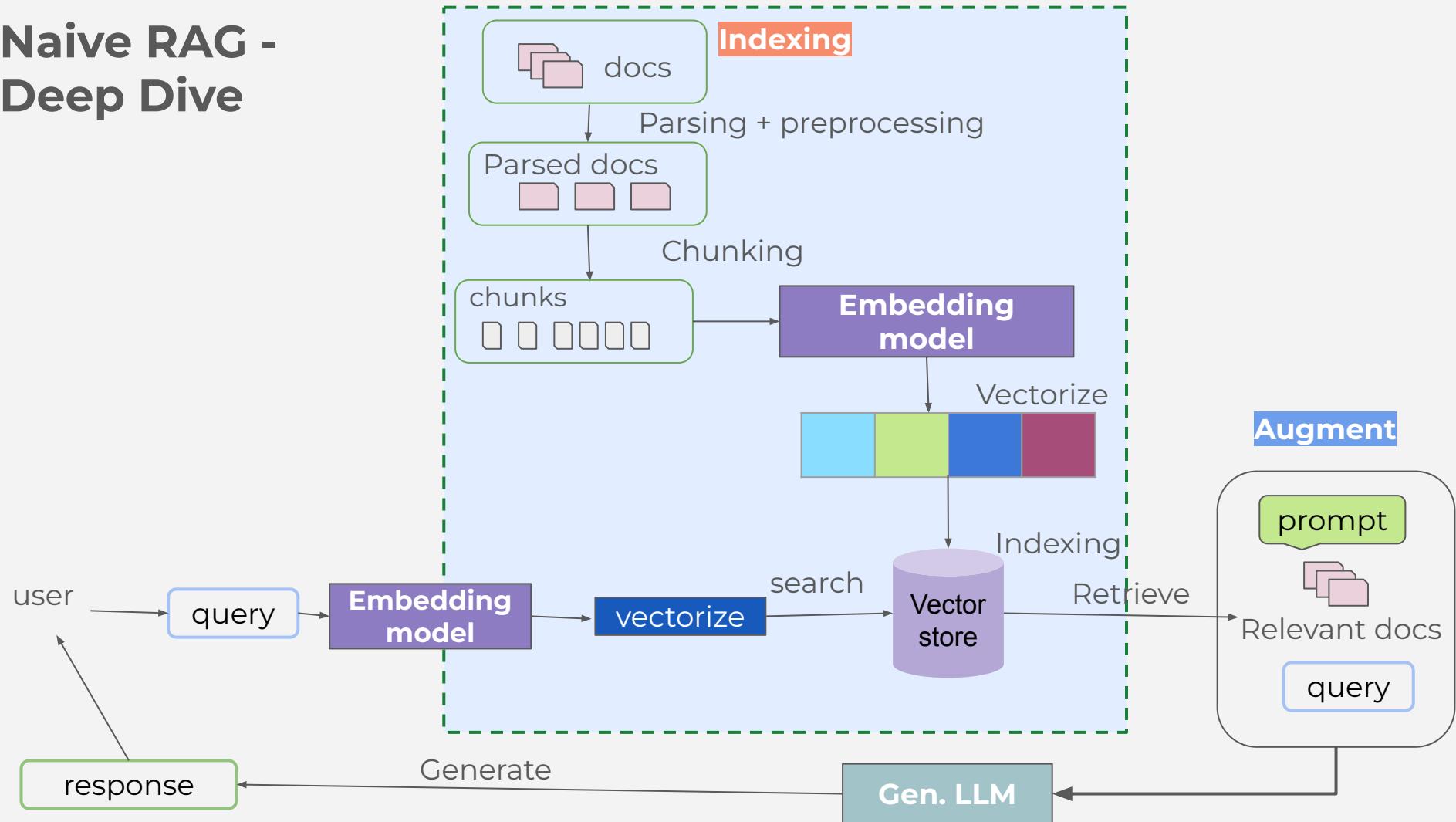
## Retrieval

Turn question  
into a vector.  
Vector  
comparison...  
Retrieves closely  
Related chunks...

## Generation

The query,  
choose docs  
combined into a  
prompt.  
The model  
generates an  
answer...

# Naive RAG - Deep Dive



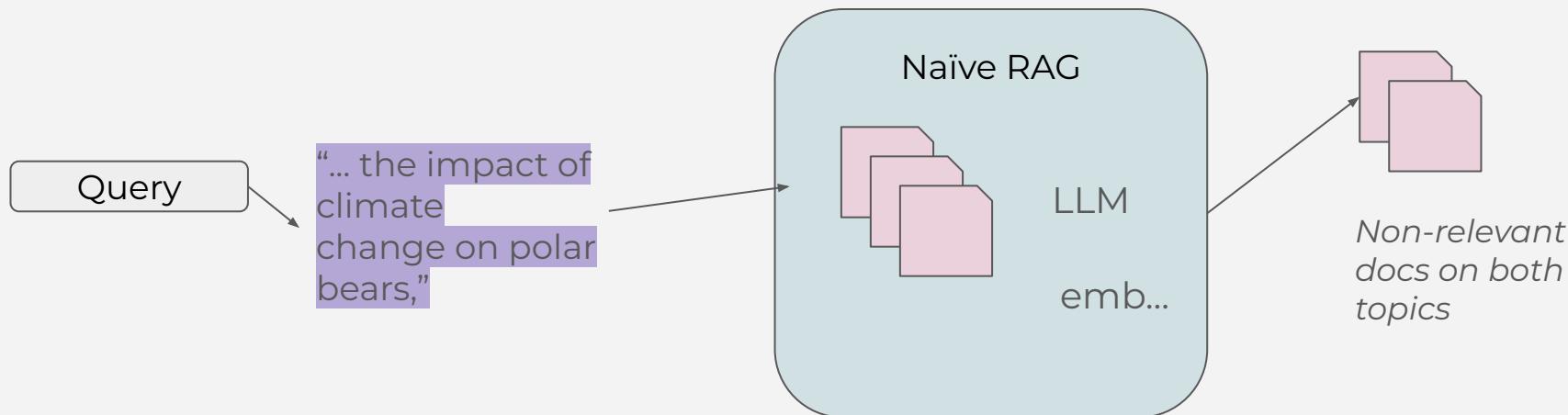
# Naive RAG Drawbacks/Challenges/Pitfalls

1. Limited contextual understanding
2. Inconsistent relevance and quality of retrieved documents
3. Poor integration between retrieval and generation
4. Inefficient handling of Large-Scale data
5. Lack of robustness and adaptability

# Naïve RAG Drawbacks/Challenges/Pitfalls

## Limited contextual understanding

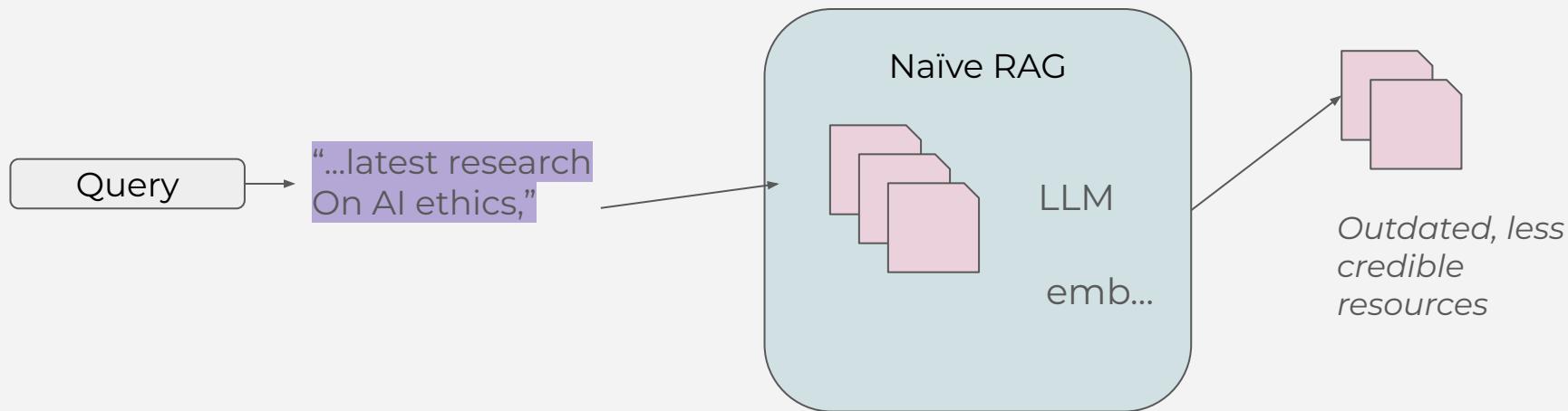
- a. Focus on keyword matching or basic semantic search (*retrieving irrelevant or partially relevant documents*)



# Naïve RAG Drawbacks/Challenges/Pitfalls

## Inconsistent relevance and quality of retrieved documents

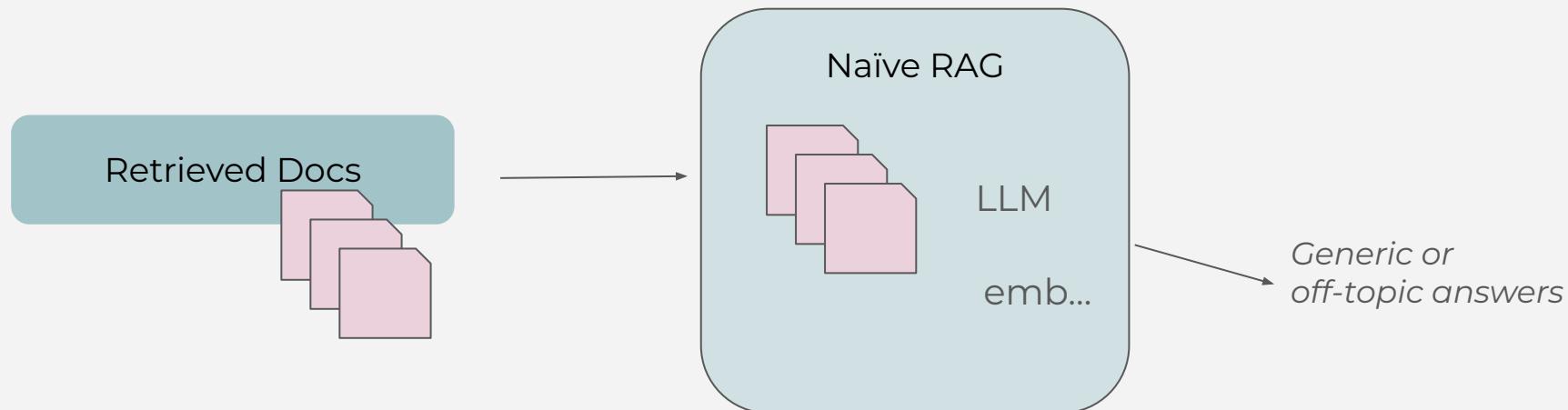
- a. Varying in quality and relevance documents  
*(poor-quality inputs for the gen model)*



# Naïve RAG Drawbacks/Challenges/Pitfalls

## Poor integration between retrieval and generation

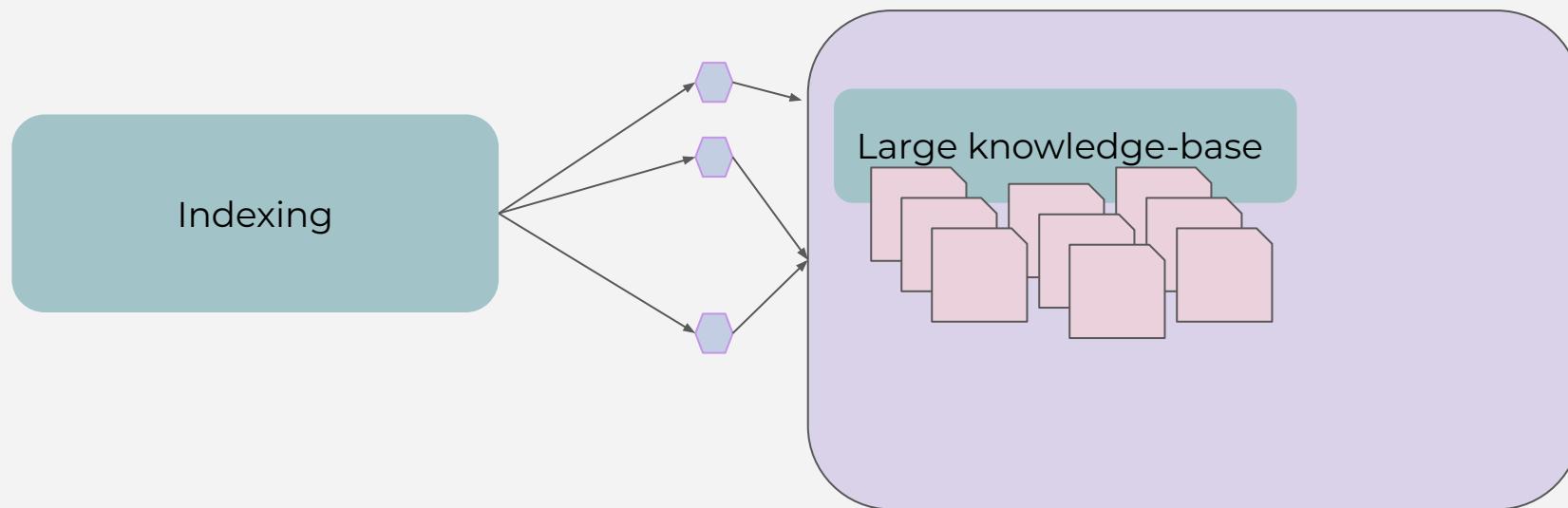
- a. Retriever and generator working in sync  
(unoptimized information)



# Naive RAG Drawbacks/Challenges/Pitfalls

## Inefficient handling of Large-Scale data

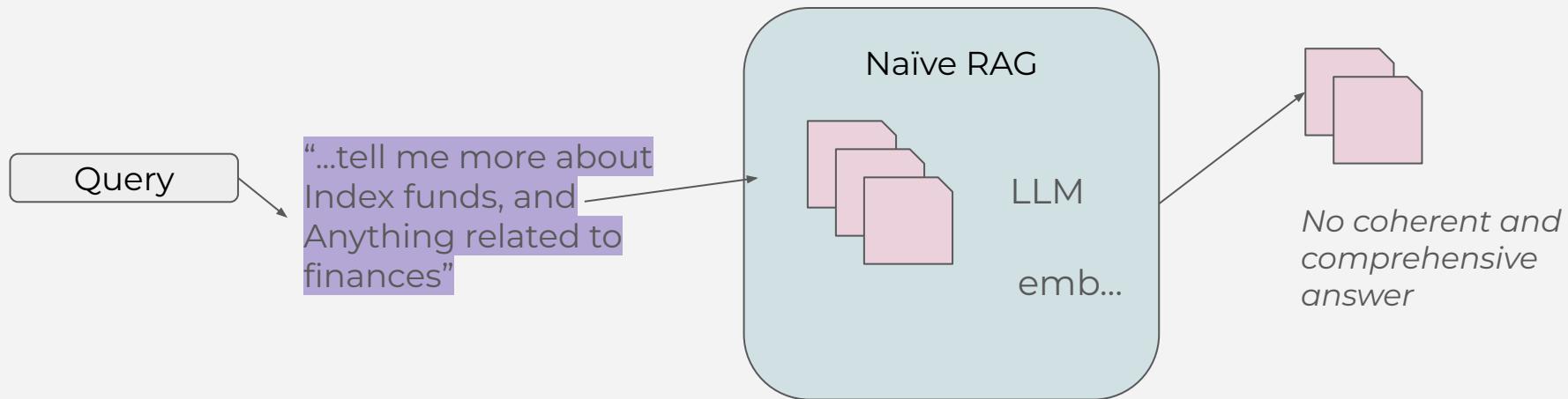
- a. Scaling issues, take too long to find relevant docs, or miss critical info due to bad indexing...



# Naïve RAG Drawbacks/Challenges/Pitfalls

## Lack of Robustness and Adaptability

- a. Not adaptable to changing contexts or user needs without significant manual intervention



# Naive RAG Drawbacks/Challenges/Pitfalls

## In summary

We have:

### 1. Retrieval Challenges

- a. Lead to the selection of misaligned or irrelevant chunks, therefore missing of crucial information

### 2. Generative Challenges

- a. The model might struggle with hallucination and have issues with relevance, toxicity or bias in its outputs

# **Advanced RAG Techniques**

And their solutions

# Advanced RAG Benefits

**Advanced RAG** - introduces specific improvements to overcome the limitations of Naive RAG. Focus on enhancing retrieval quality.

Advanced RAG employs the following strategies:

## 1. Pre-retrieval

- a. Improvement of the indexing structure and user's query
- b. Improves data details, organizing indexes better, adding extra information, aligning things correctly...

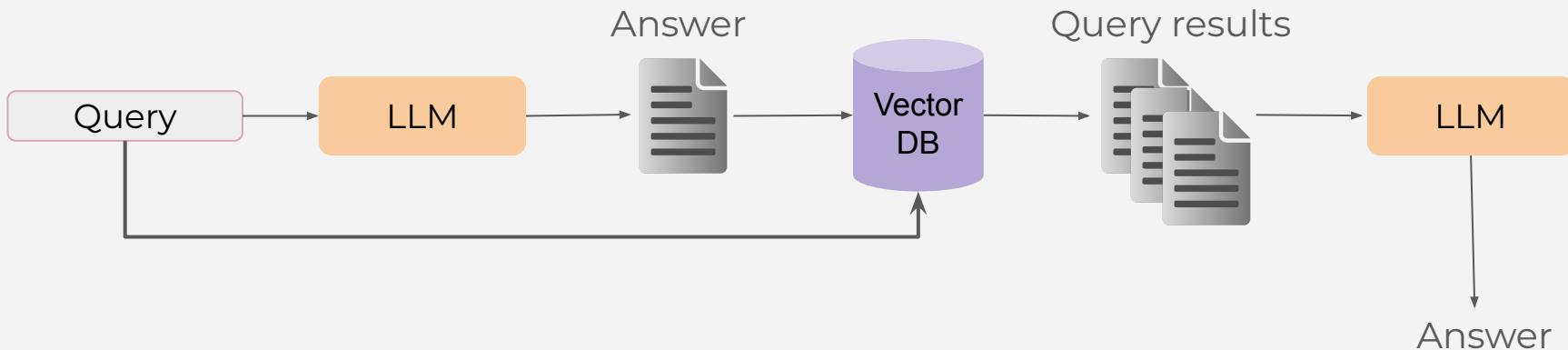
## 2. Post-retrieval

- a. Combine pre-retrieval data with the original query
  - i. Re-ranking to highlight the most important content...

# Advanced RAG Techniques

## Query Expansion (*with generated answers*)

Generate potential answers to the query  
[using an *LLM*] and to get relevant context



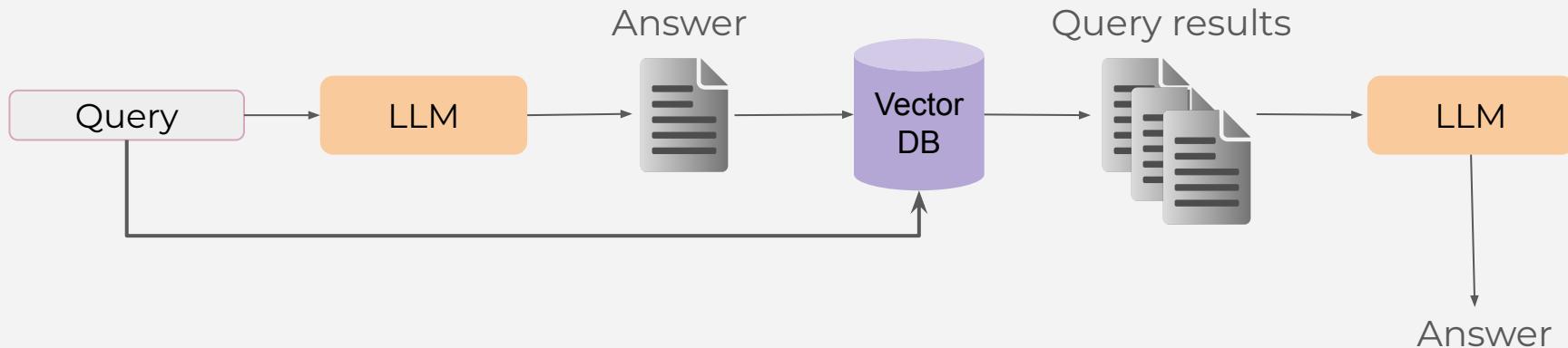
# Advanced RAG Techniques

## Query Expansion (*with generated answers*)

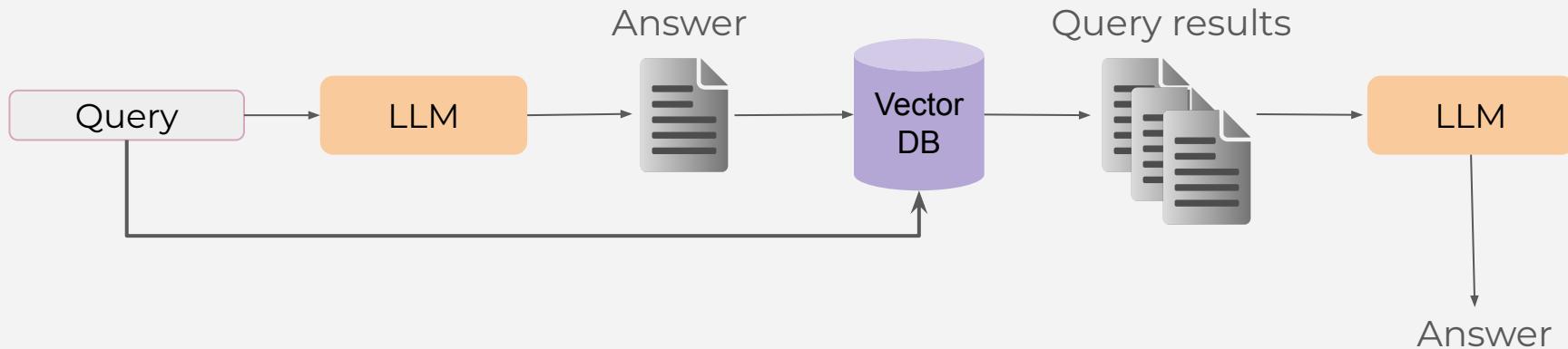
Use cases:

- Information Retrieval
- Question Answering Systems
- E-commerce Search
- Academic Research

# Hands-on - Query Expansion



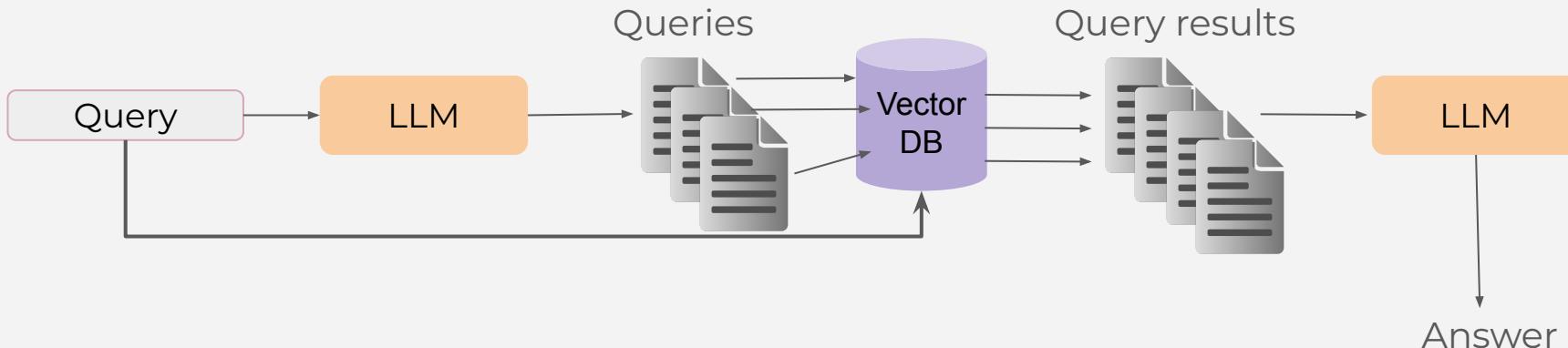
# STOP here



# Advanced RAG Techniques

## Query Expansion (with multiple queries)

Use the LLM to generate additional queries that might help getting the most relevant answer.



# Advanced RAG Techniques

## Query Expansion (*with multiple queries*)

Use cases:

- Exploring Data Analysis
- Academic Research
- Customer Support
- Healthcare Information Systems

# Advanced RAG Techniques

Query Expansion (with multiple queries)

# Advanced RAG Techniques

Downsides:

- Lots of results
  - queries might not always be relevant or useful
- Results not always relevant and or useful

Hence, we need another technique to find relevant results...

**Relevance Feedback or Re-ranking**

# Advanced RAG Techniques - Expansion with Multi queries

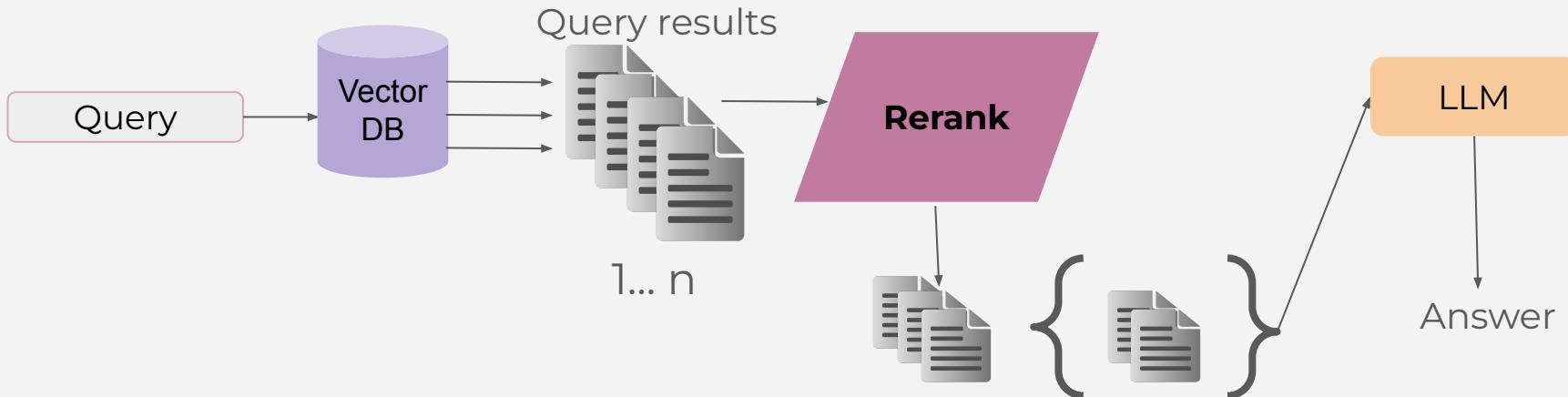
Your turn:

- Play with different prompts and queries
  - See what results you get each time
  - Keep refining the prompt and see the results

# Advanced RAG Techniques

## Re-ranking(Cross-encoder)

Applying a sophisticated process to re-evaluate and reorder the initial retrieved documents



# Advanced RAG Techniques

## Re-Ranking

Use cases:

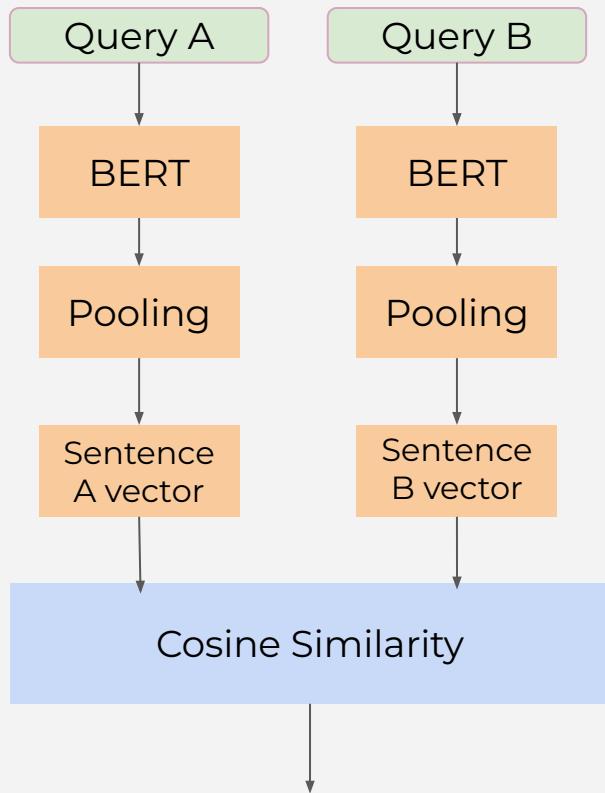
- Search engines
- Question Answering systems
- Recommendation systems
- Legal document search

# Hands on

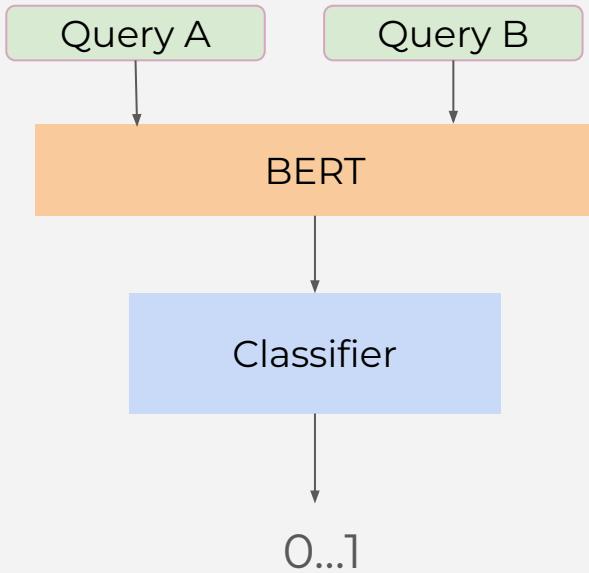
Re-ranking

# Cross-Encoder Model

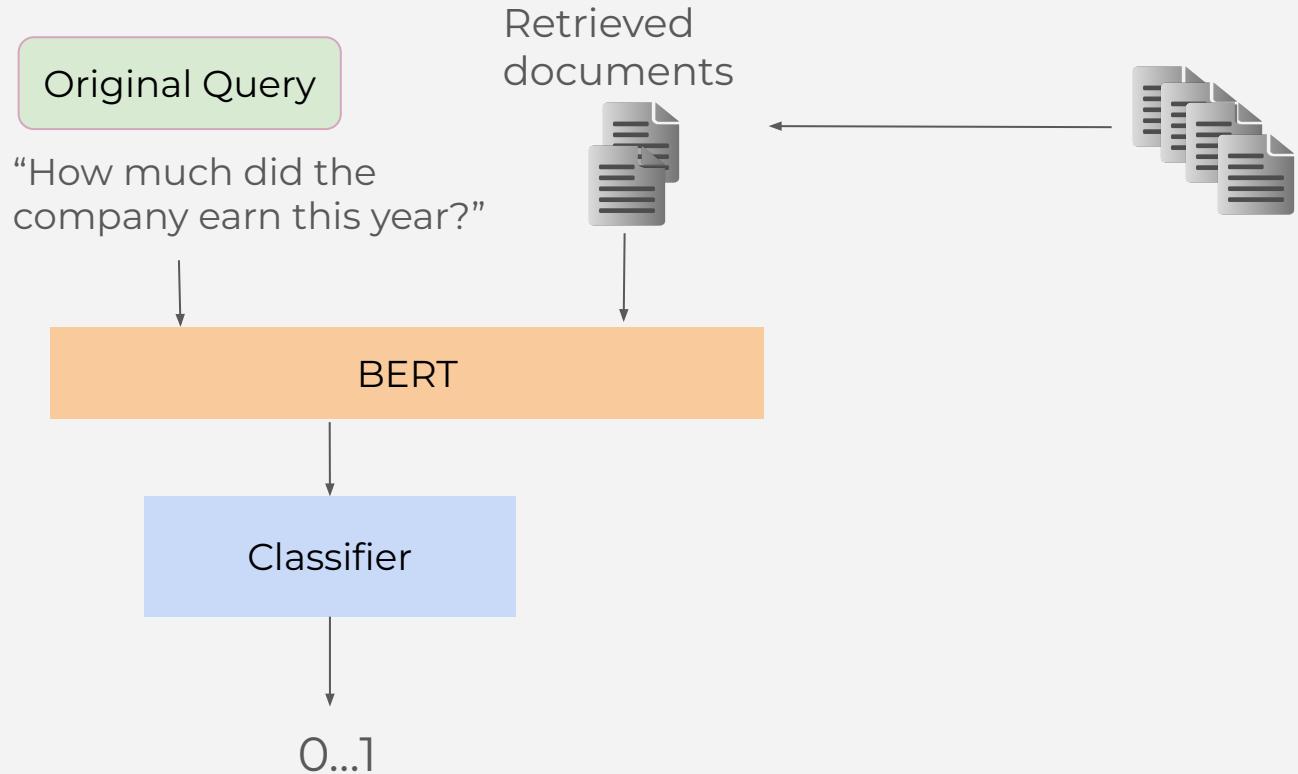
Bi-encoder



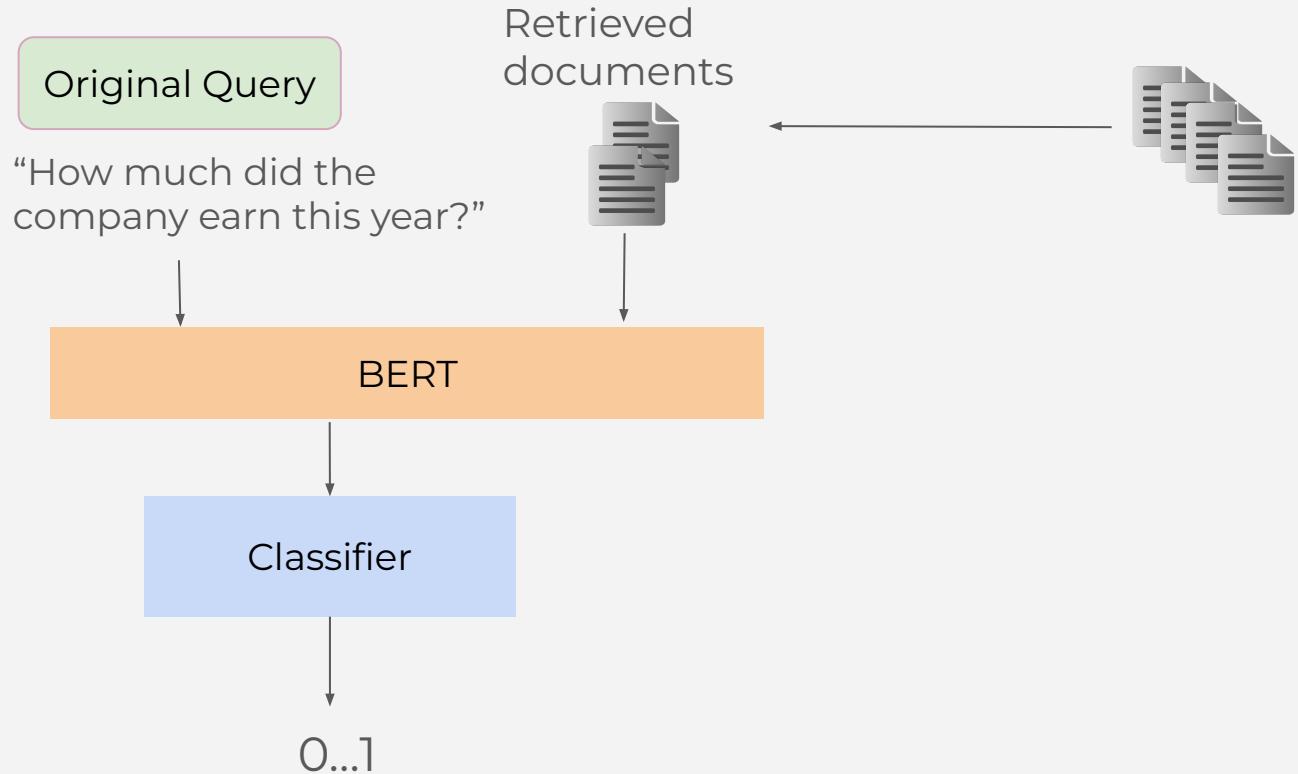
Cross-encoder



# Cross-Encoder in Re-ranking

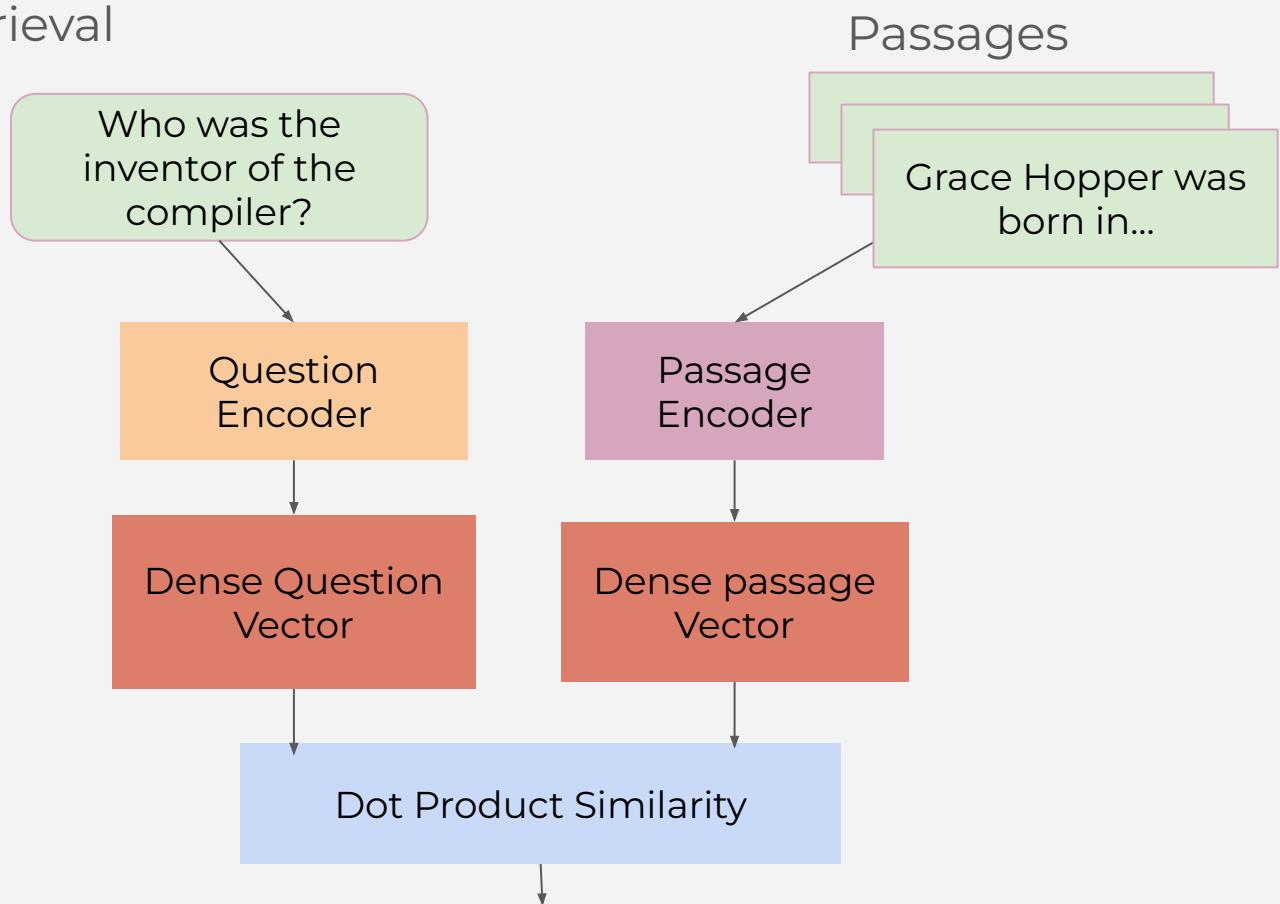


# Cross-Encoder in Re-ranking



# Other techniques...

## Dense Passage Retrieval



# Advanced RAG Techniques

## DPR

Use cases:

- Open-Domain question answering
- Document retrieval
- Customer support

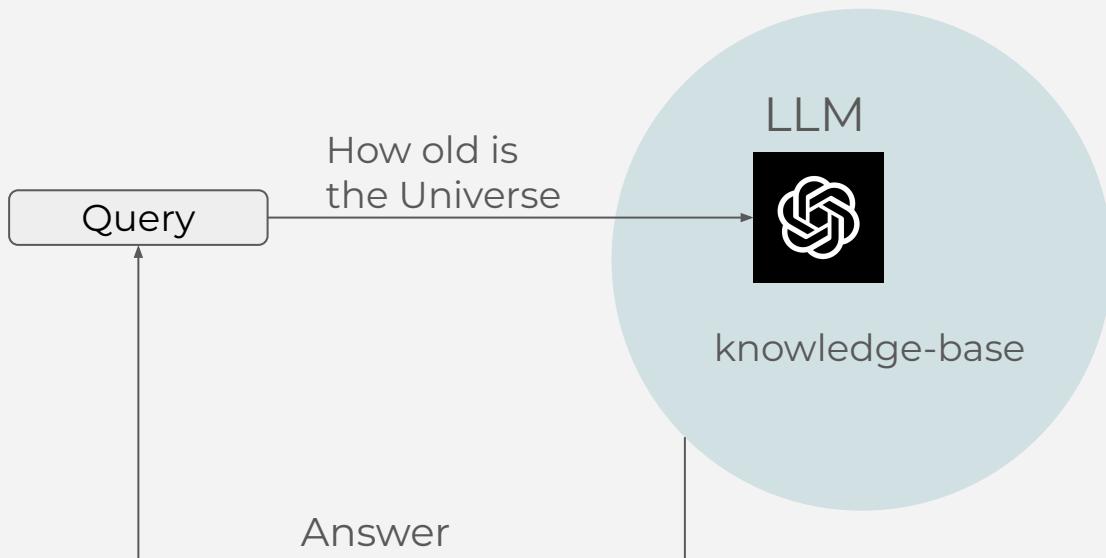
# AI Agents

Fundamentals

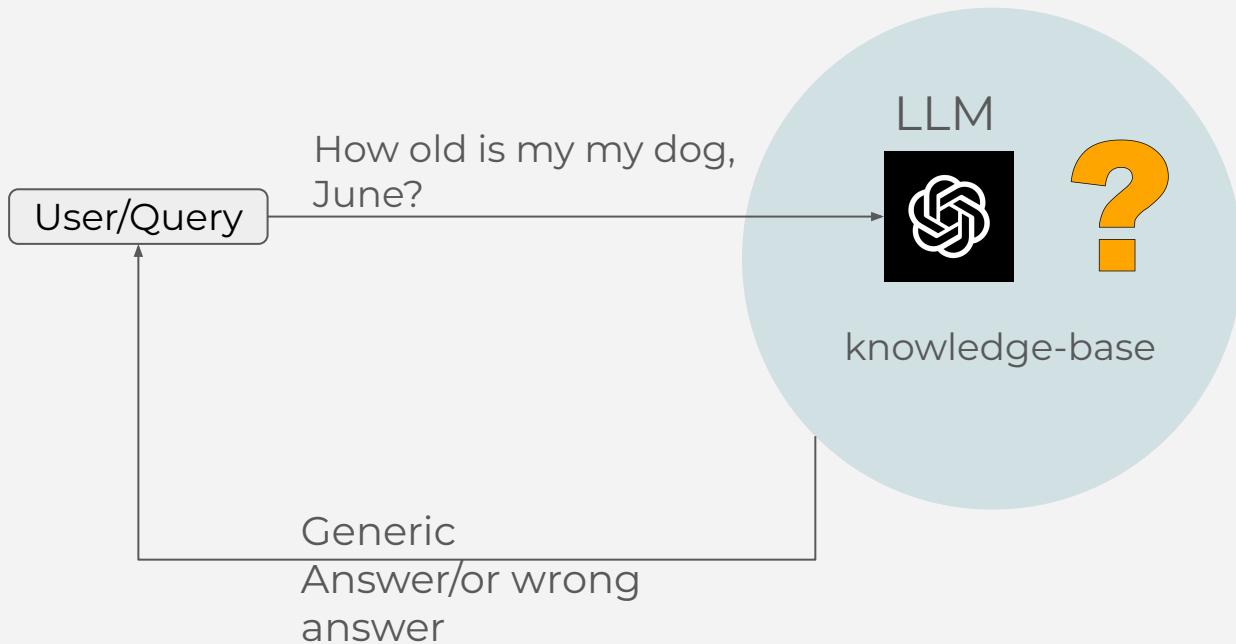
# *AI Agent Deep Dive*

- What is it?
- Why (motivation)?
- Advantages

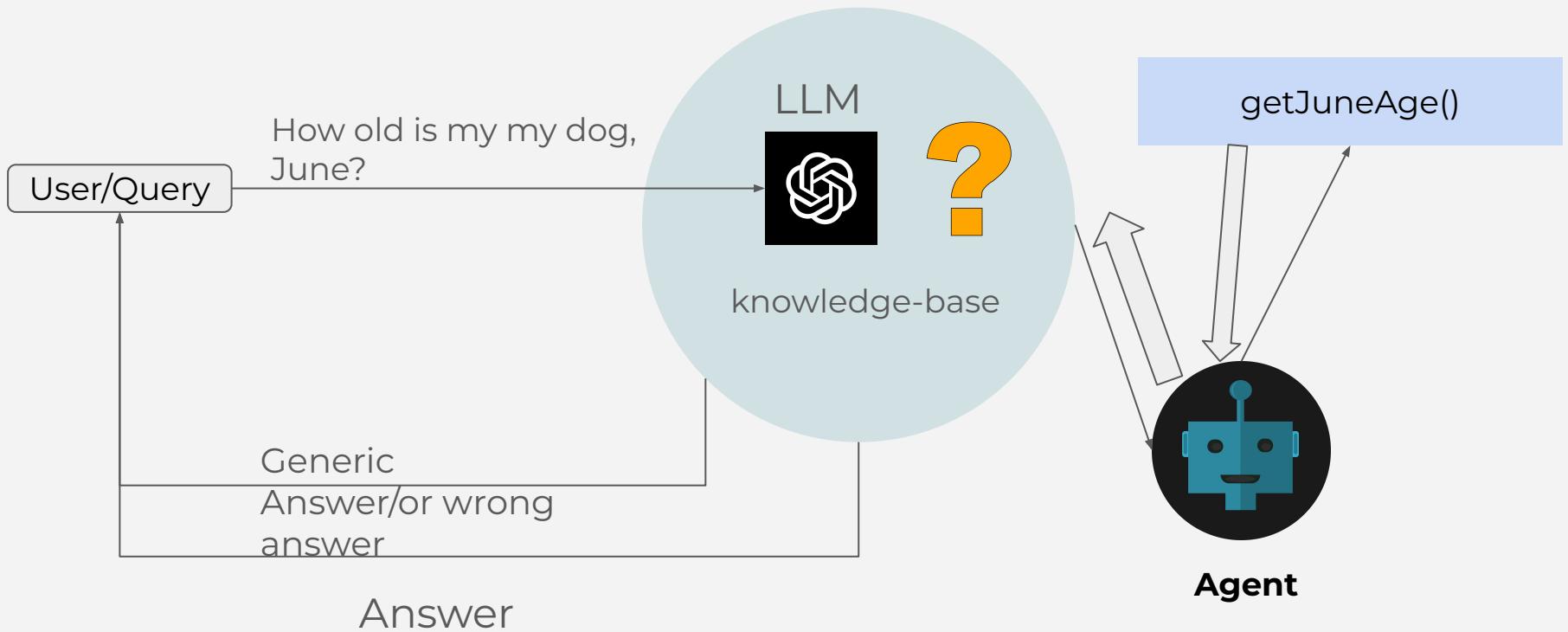
# What is an (AI) agent (Motivation)?



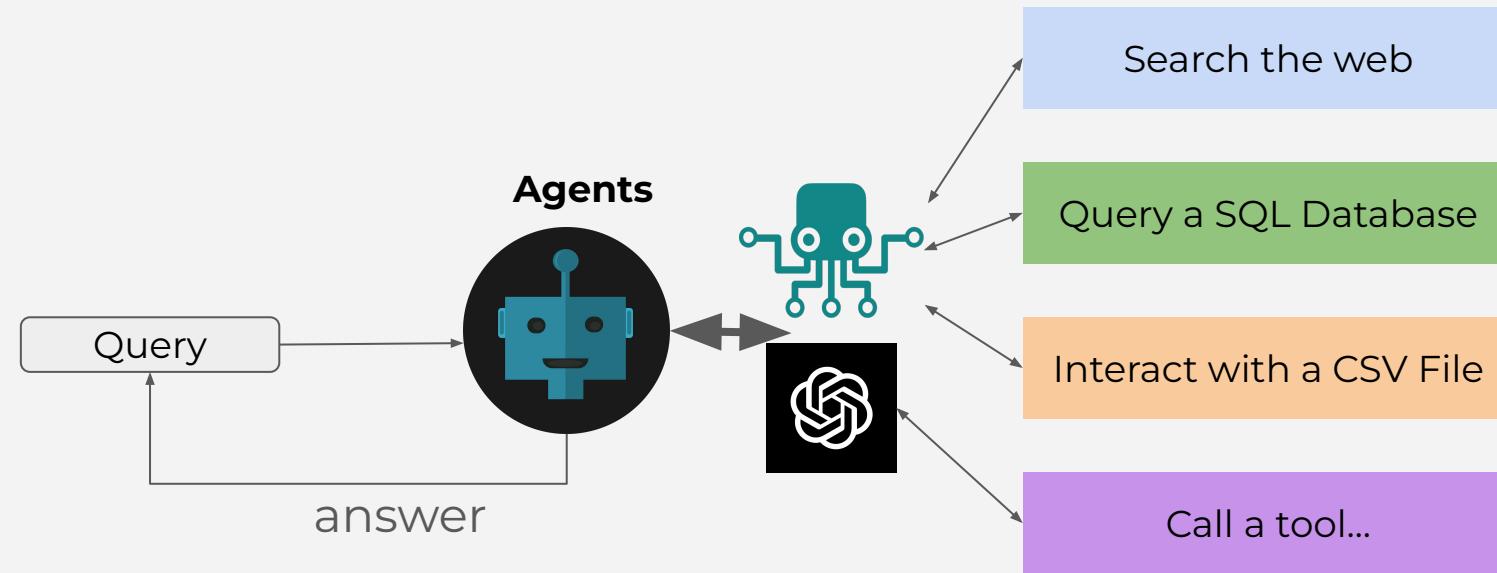
# What is an (AI) agent (Motivation)?



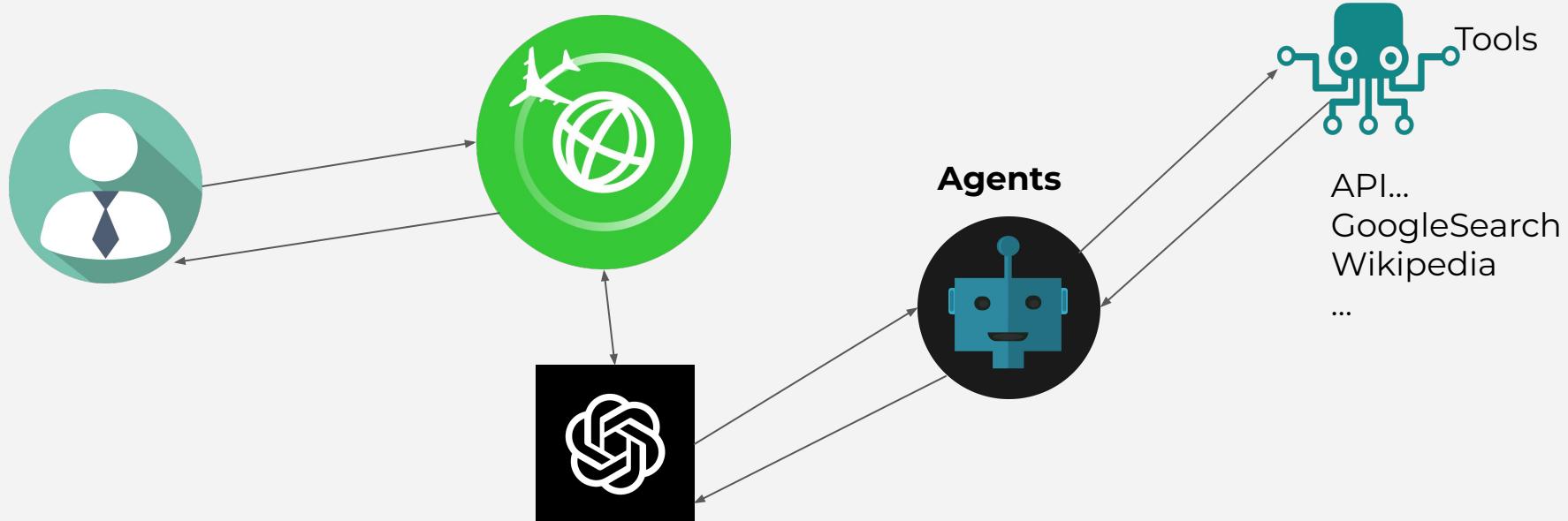
# What is an (AI) agent (Motivation)?



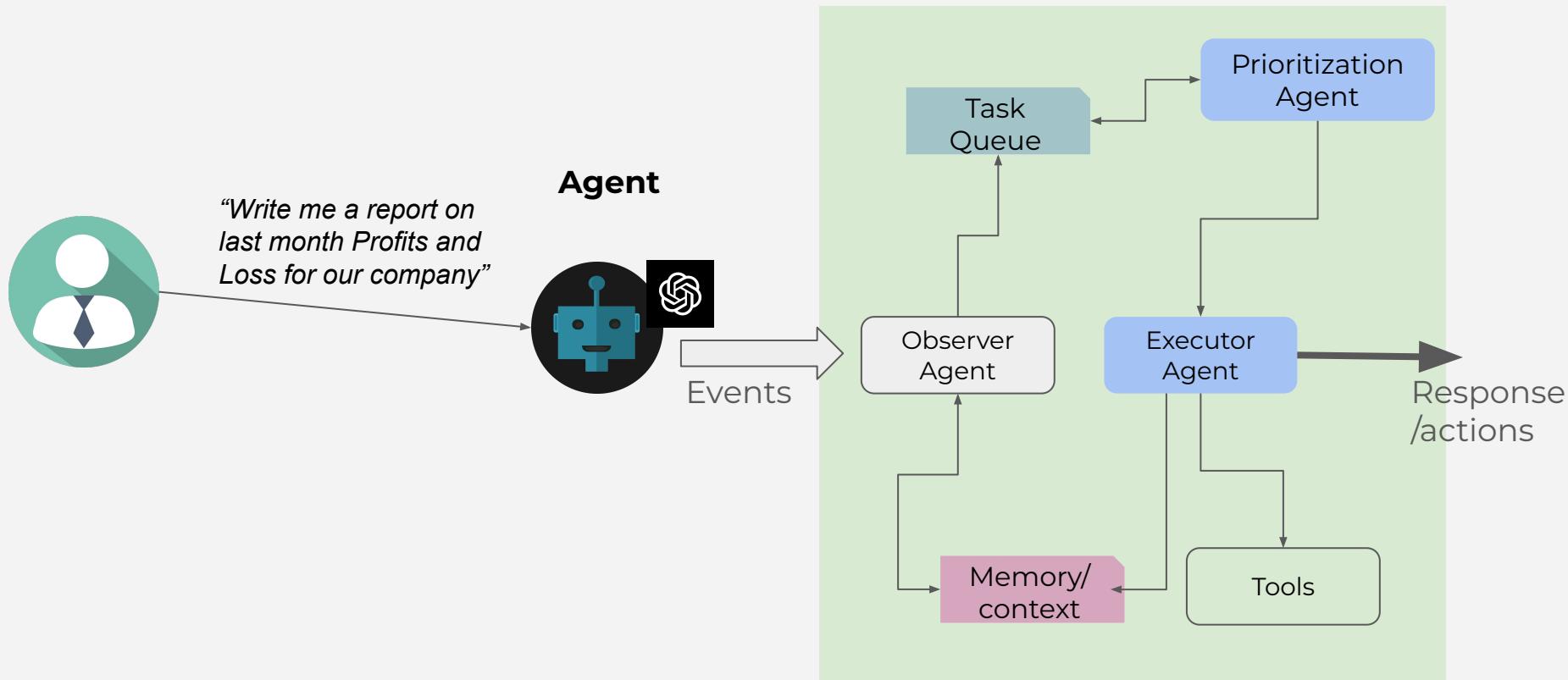
# What is an (AI) agent?



# What is an agent & what they can do?



# Motivation Behind AI Agents: Solving Real-World Problems



# **Key characteristics**

## **Autonomy**

Learning and adaptation

Interaction

Goal-driven

# Use cases

Customer service chatbots

Personal assistants

Data analysis

Smart home systems...

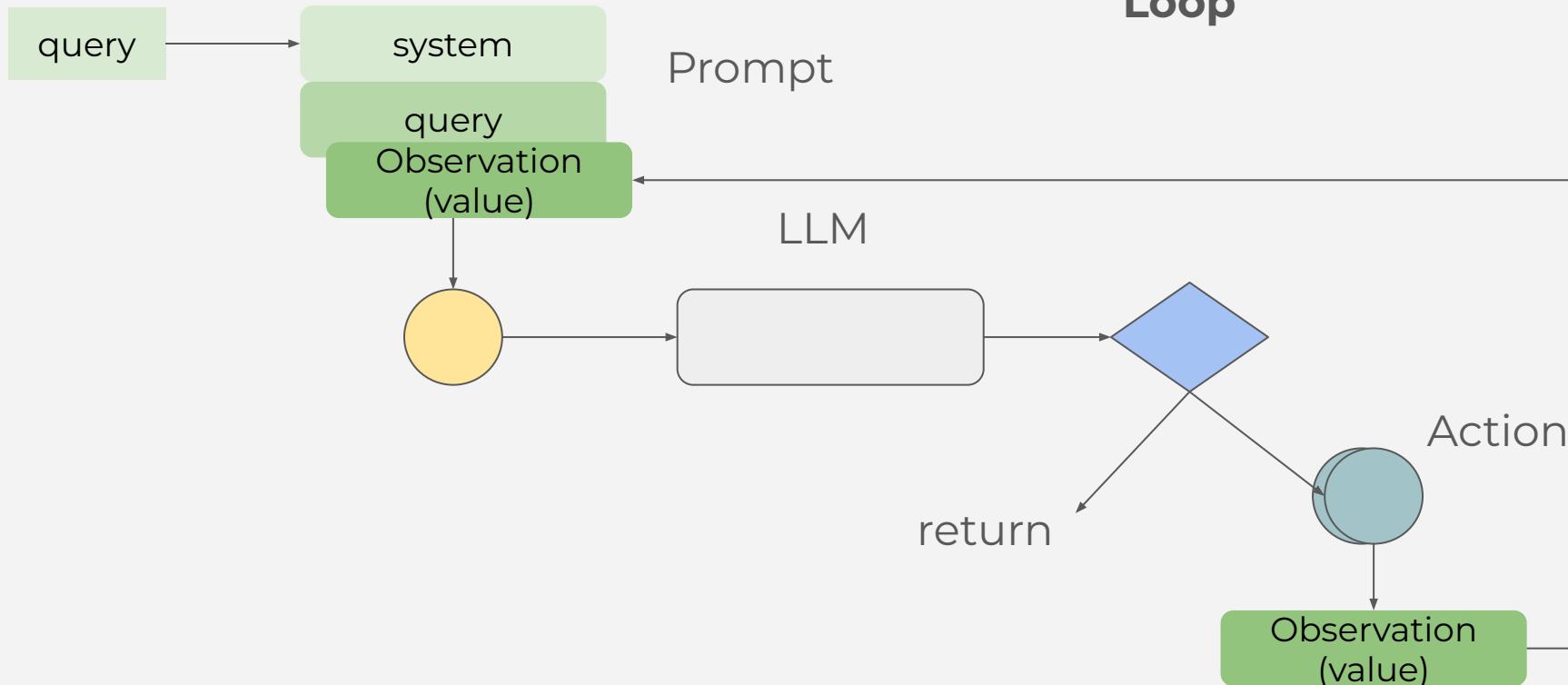
# First AI Agent

Build your very first Agent

- OpenAI model
- Python

# First Agent

User



# LangGraph

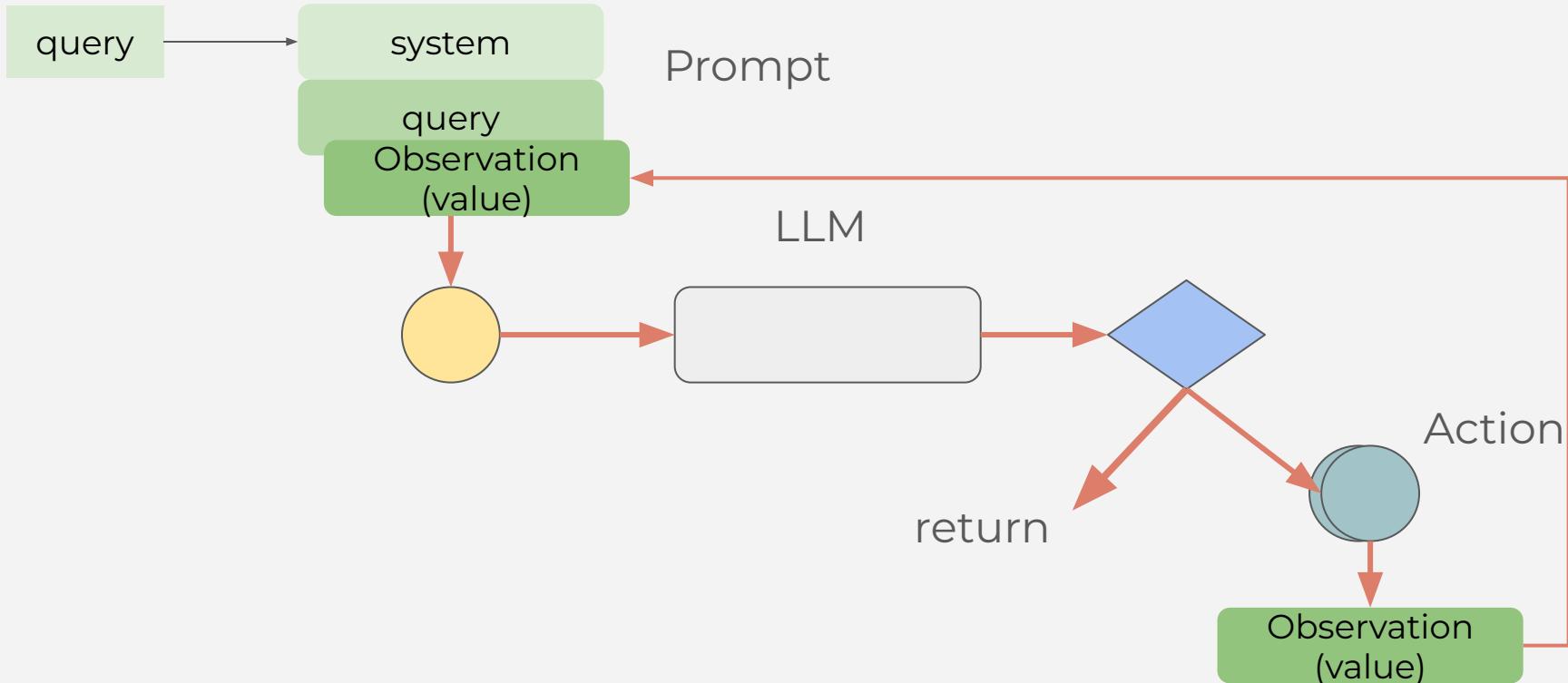
## *Deep Dive*

Understanding LangGraph

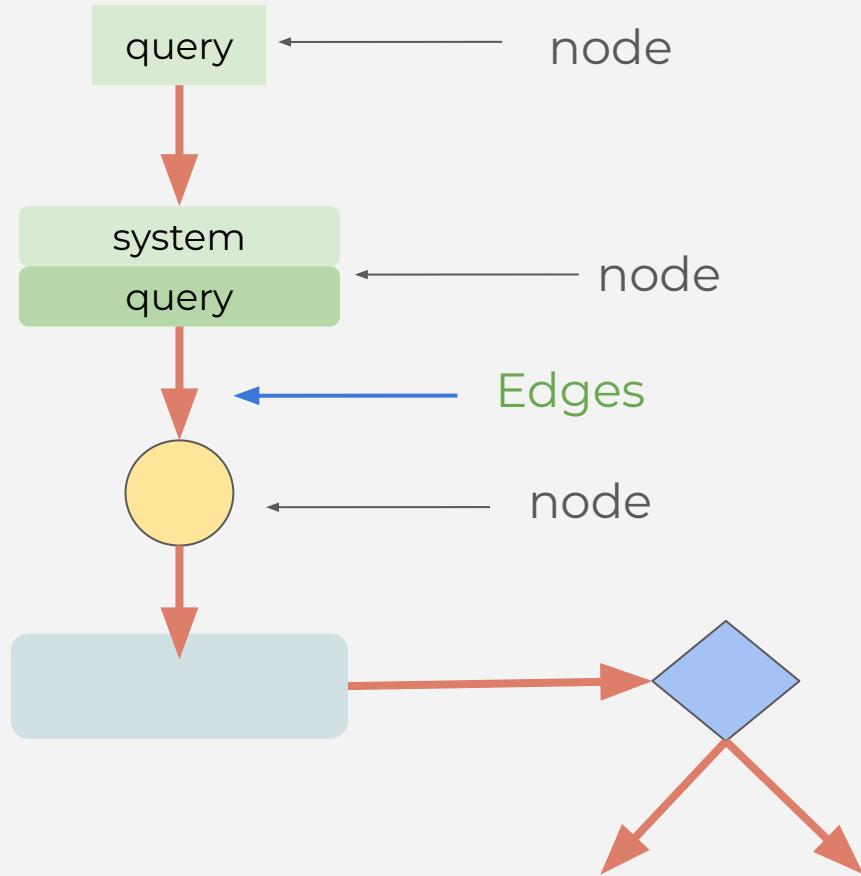
- What is it?
- How does it work?
- How to use it to build AI Agents

# What is LangGraph?

User



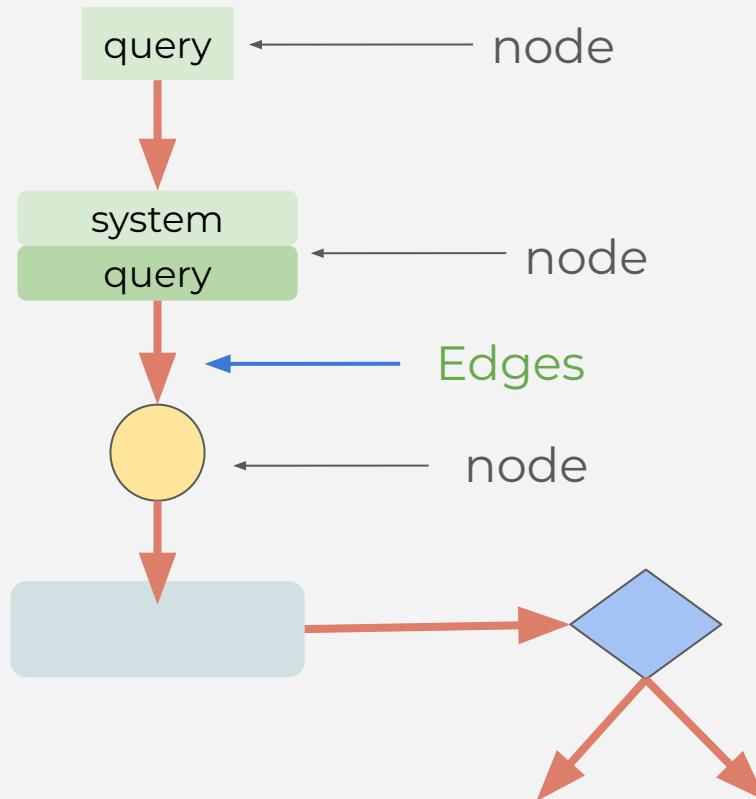
# What is LangGraph?



**Nodes** - entities, concepts

**Edges** - relationships between nodes.

# What is LangGraph - Key Components

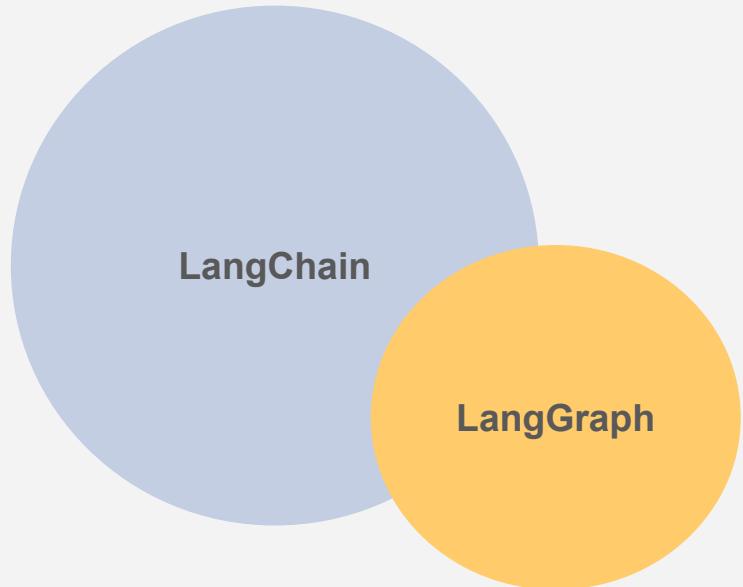


1. **Graph-Based Knowledge Representation**
  - a. Uses graphs to represent knowledge
2. **Natural Language Processing**
  - a. Uses NLP to parse and understand user input
3. **Reasoning and Inference**
  - a. Logical reasoning and inference to make decisions
4. **Interaction and Dialogue Management**
  - a. Maintain context

# How LangGraph Helps in Building AI Agents

1. **Enhanced Knowledge Management**
  - a. Graphs are amazing for organizing and querying knowledge
2. **Improved Natural Language Understanding**
  - a. NLP + graph-based representation is a great combination for agents
3. **Efficient Reasoning and Decision Making**
  - a. Graph-based reasoning enables agent to become more efficient
4. **Scalability and Flexibility**
  - a. It's easy to scale graphs to handle large datasets and complex relationships

# LangGraph and LangChain



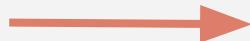
**LangGraph** is an extension of **LangChain** that support Graphs

- Allows to create **cyclic graphs**
- **Persistence** - *remembering previous conversations or context*
  - **Human-in-the-loop** - *human interaction and feedback (thanks to persistent feature)*

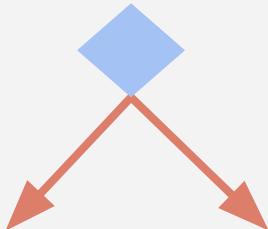
# LangGraph Core Concepts



**Nodes:** Agents (entities) or functions

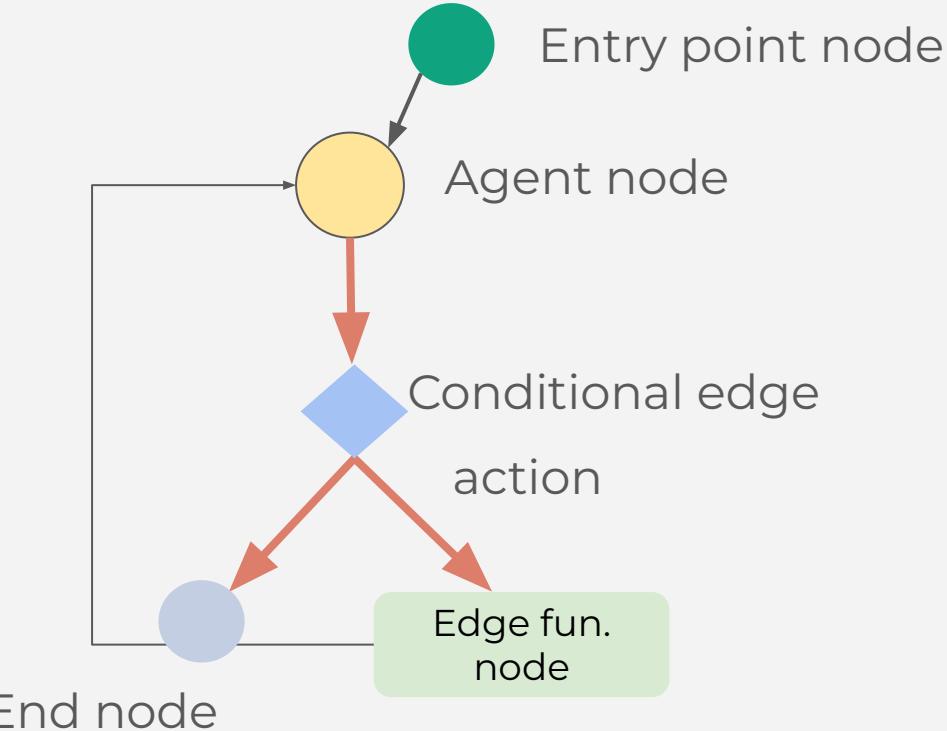


**Edges:** connect nodes (relationships)

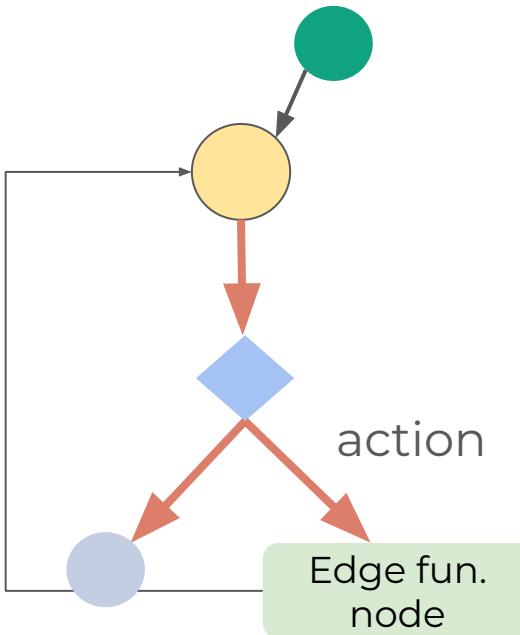


**Conditional Edges:** Decision

# LangGraph Flow Example



# LangGraph Data & State



- State is accessible to all parts of the graph
- Local to the graph
- Can be stored (DB, etc)

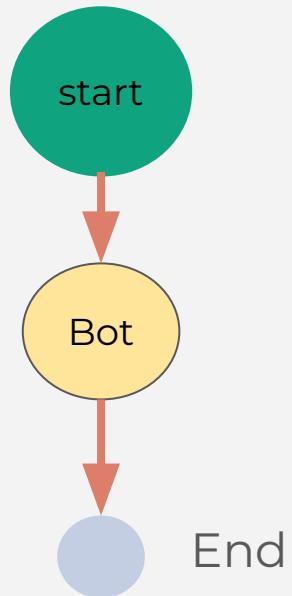
# LangGraph

## *Hands on*

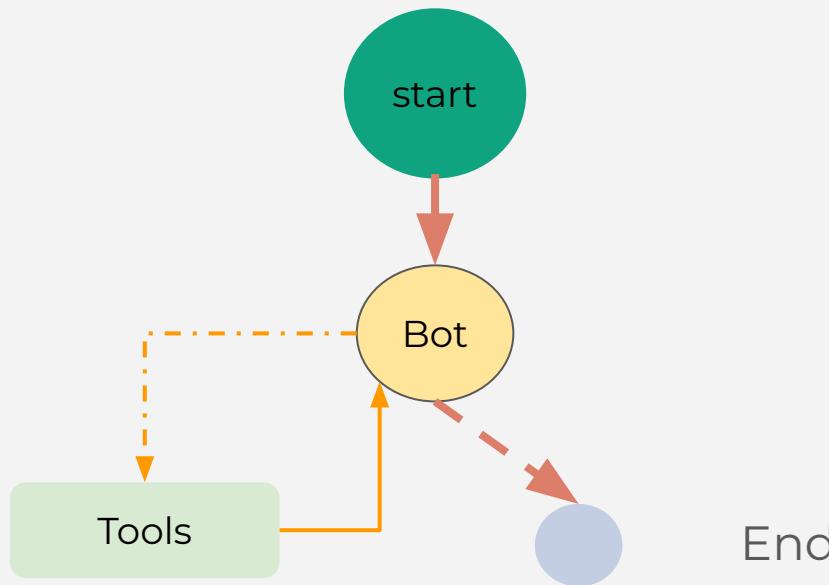
LangGraph Hands on

- Build a simple Agent with...
  - LangGraph
    - Basics of LangGraph

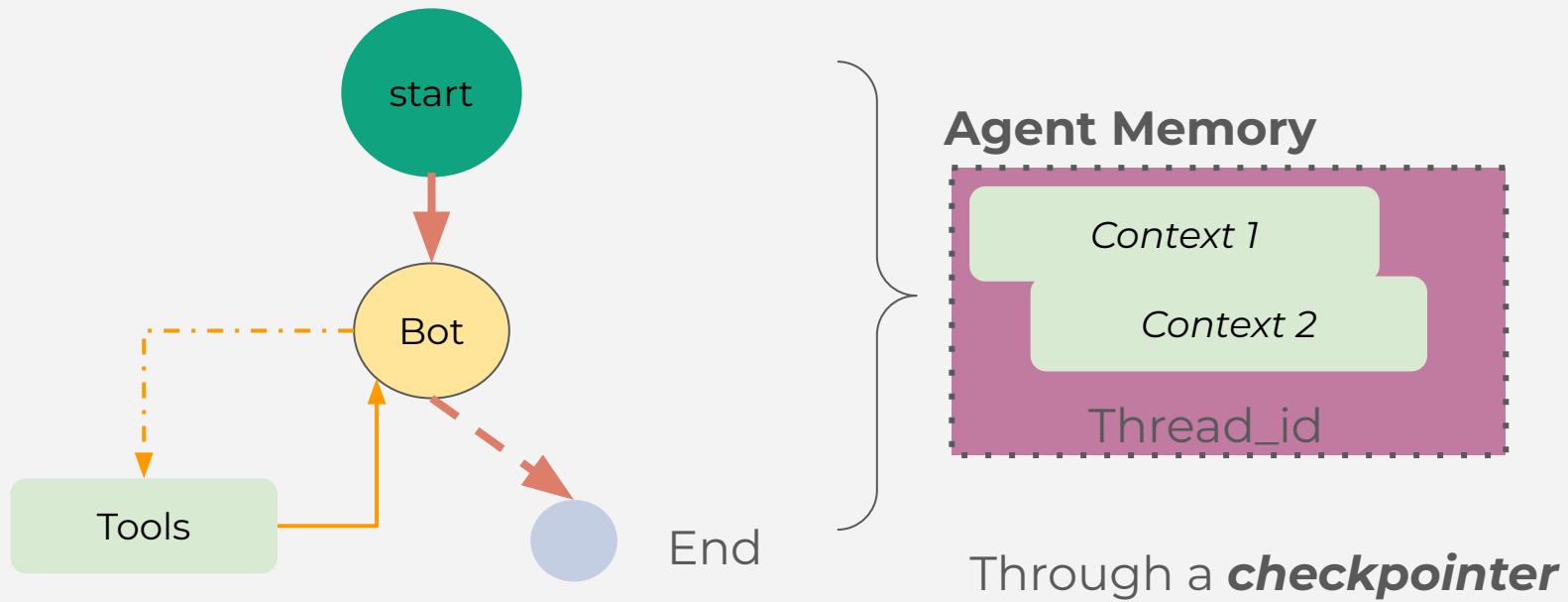
# LangGraph - Basic Agent



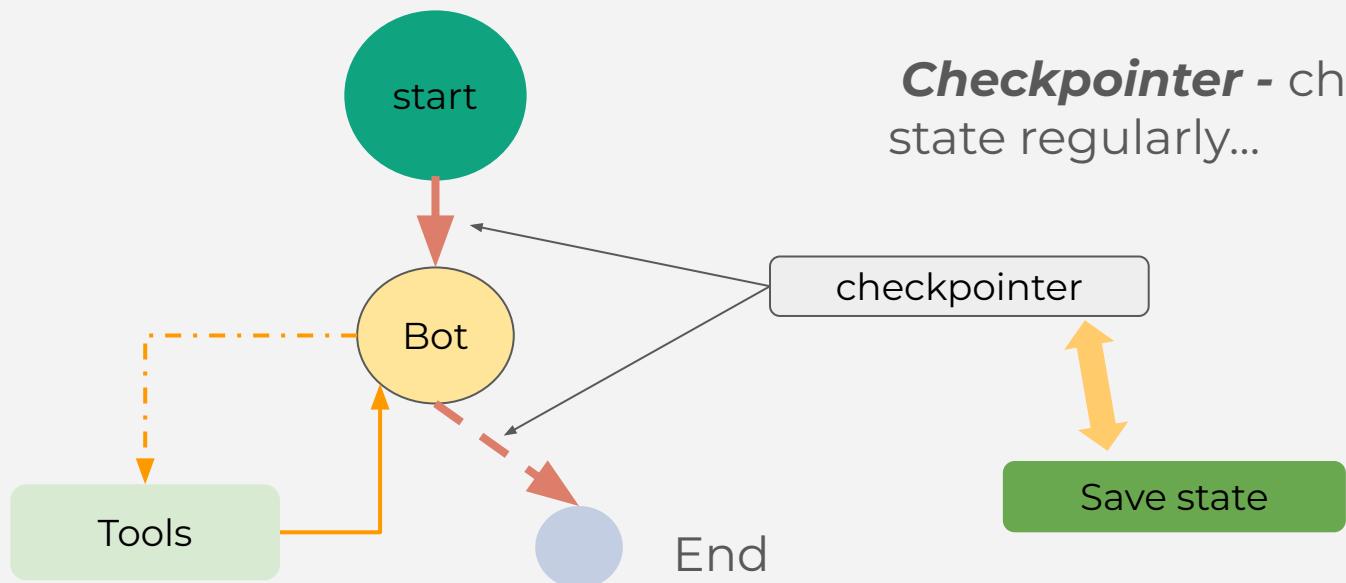
# LangGraph - Basic Agent - Add Tools



# LangGraph - Basic Agent - Add Memory

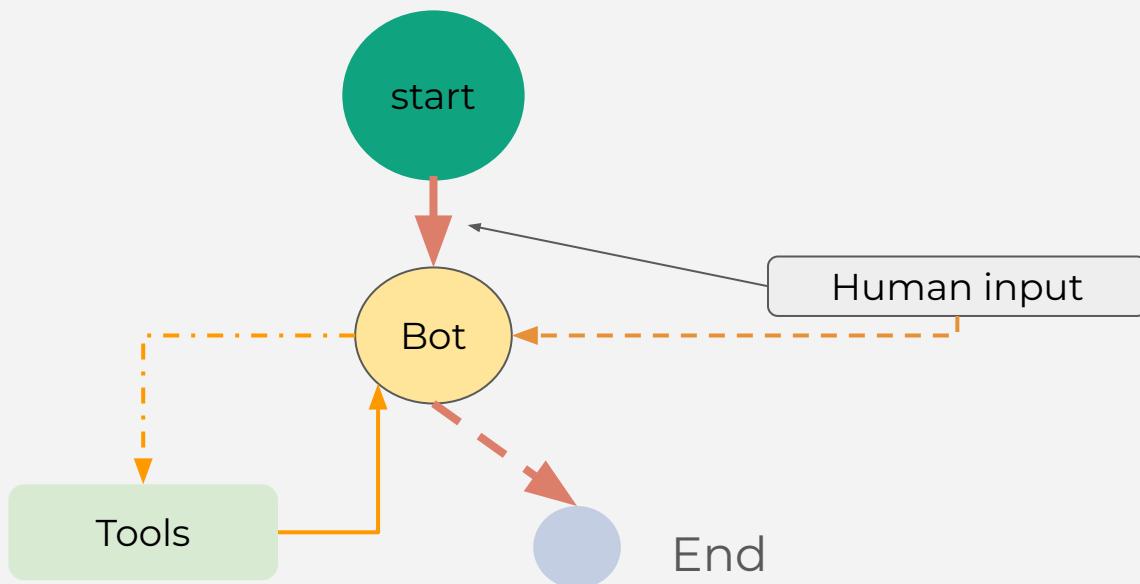


# LangGraph - Basic Agent - Add Memory



**Checkpointer** - checks the state regularly...

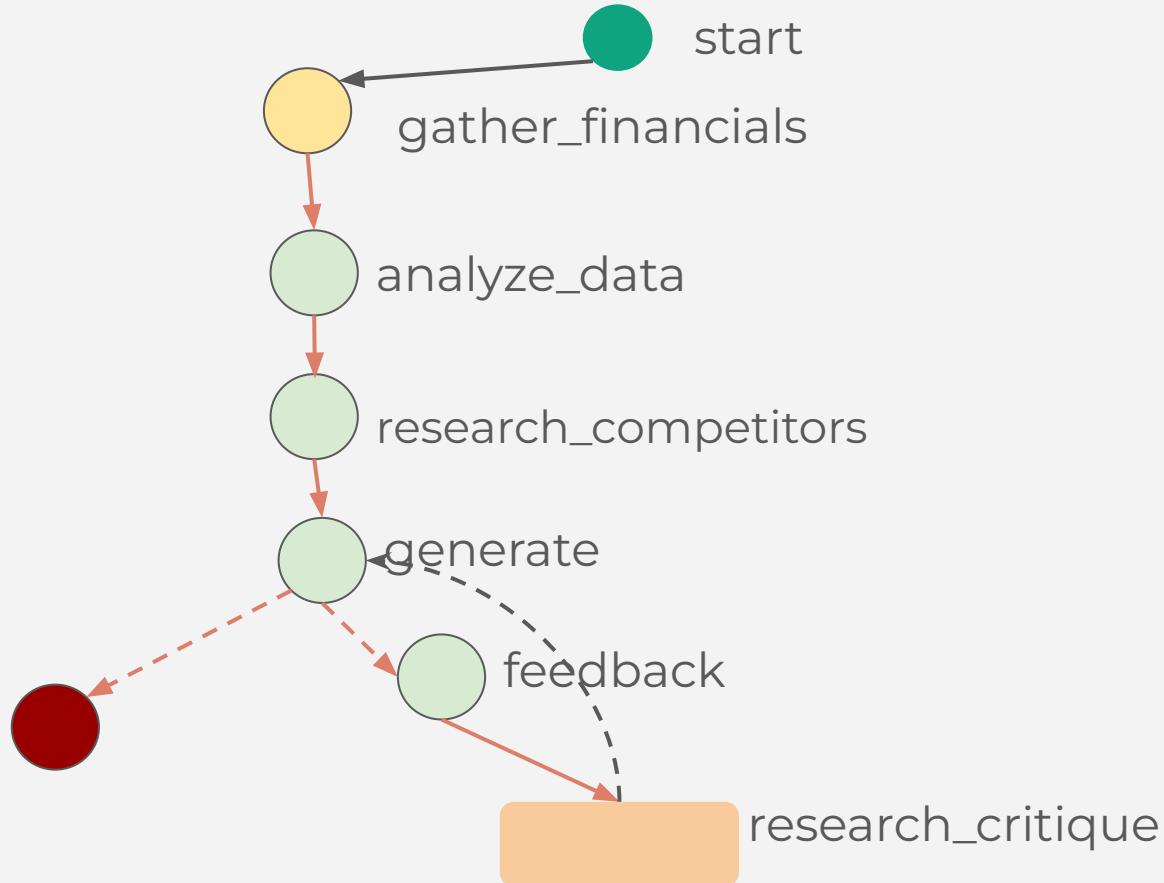
# LangGraph - Basic Agent - Human-in-the-loop



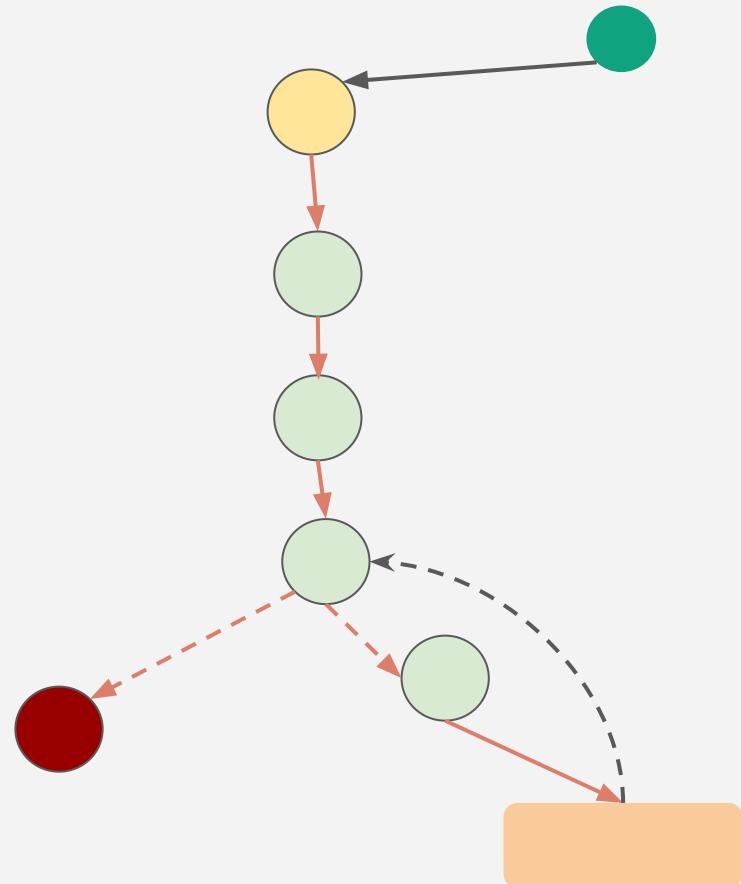
# Build a Full AI Agent

Build a full AI Agent  
**Financial Performance**  
**Reporting Writer Agent**

# Financial Report Writer Agent



# Agent Optimization - Overview



## 1. Model Optimization

- a. Use efficient models
  - i. Utilize smaller, efficient models for less intensive tasks
- b. Model Quantization
- c. Fine-Tuning

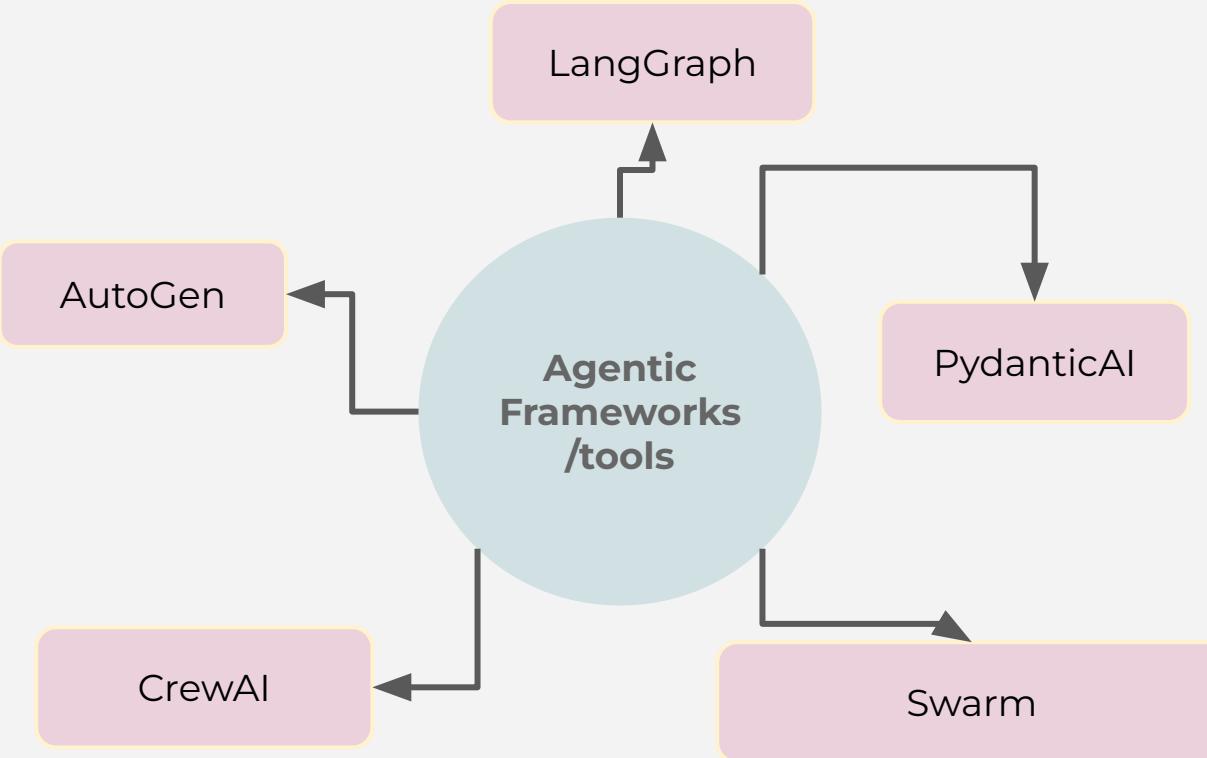
## 2. Parallel Processing

- a. Distributed computing
- b. Batch processing

## 3. Caching and Preprocessing

- a. Data caching
- b. Preprocessing

# Agentic Frameworks



# Demo a few...

- OpenAI Swarm
- PydanticAI
- CrewAI
- Phidata
- ..

# Multimodal Document Analysis

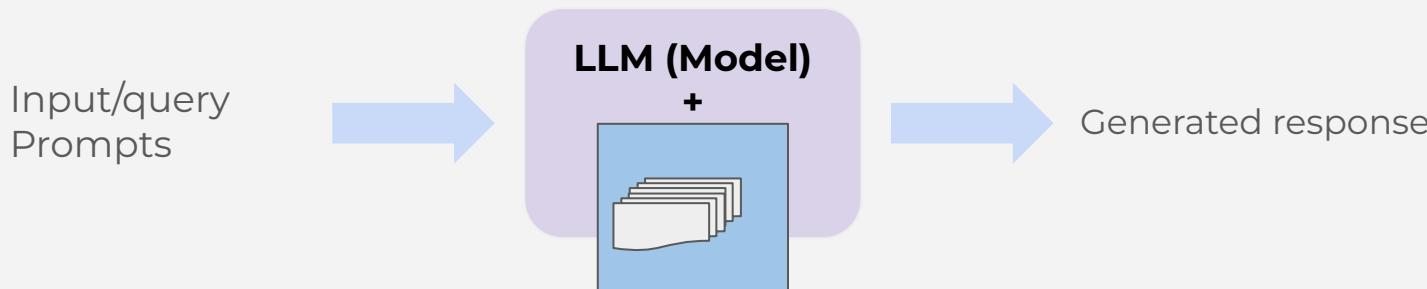
**Implementing RAG with  
Multimodal Data**

# ***RAG & Multimodal RAG Deep Dive***

- What is RAG (overview)
- How it works?
- Multimodal RAG - overview
- Advantages
- Disadvantages
- General RAG vs Multimodal RAG

# Practical Applications of RAG

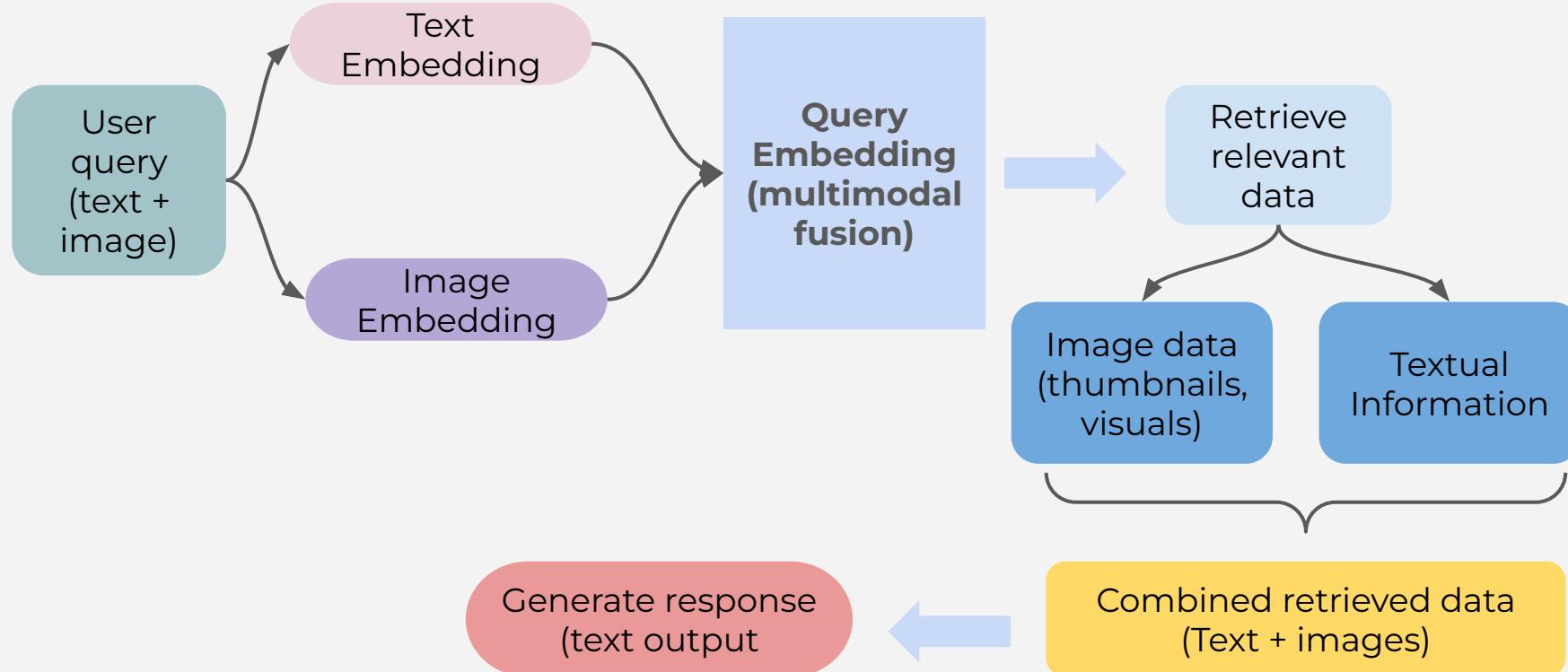
- **Question answering systems:** Q&A systems that retrieve factual information from a knowledge base.
- **Search engine enhancement:** users get direct, well-formed answers derived from multiple sources
- **Document summarization:** synthesizes summaries on sections from specific document
- **Education and learning tools:** build systems to help students with questions related to relevant educational materials.



# Benefits of Retrieval-Augmented Generation

- **Enhanced factuality:** now we have up-to-date and factual information from external sources (no more solely relying on the pre-trained knowledge of the model).
- **Improved accuracy and specificity:** retrieves relevant context and grounds model's response in factual data
- **Reduced hallucination**
- **Adaptability:** no need for retraining the model anymore

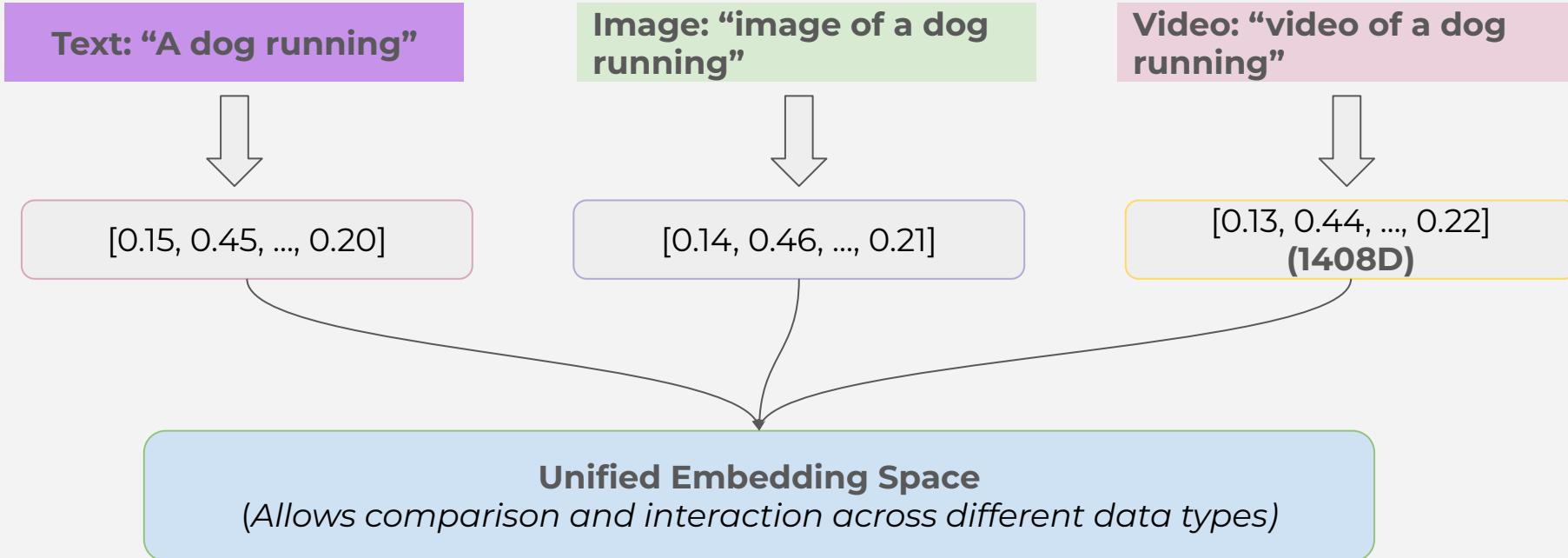
# Multimodal RAG (motivation)



# Two options for multimodal RAG

- **Multimodal embeddings**
  - This model generates a high-dimension vectors. Image and text embedding are in the same semantic space with the same dimensionality
- **Text embeddings**
  - The multimodal LLM generates text summaries from visual inputs.

# Multimodal embeddings

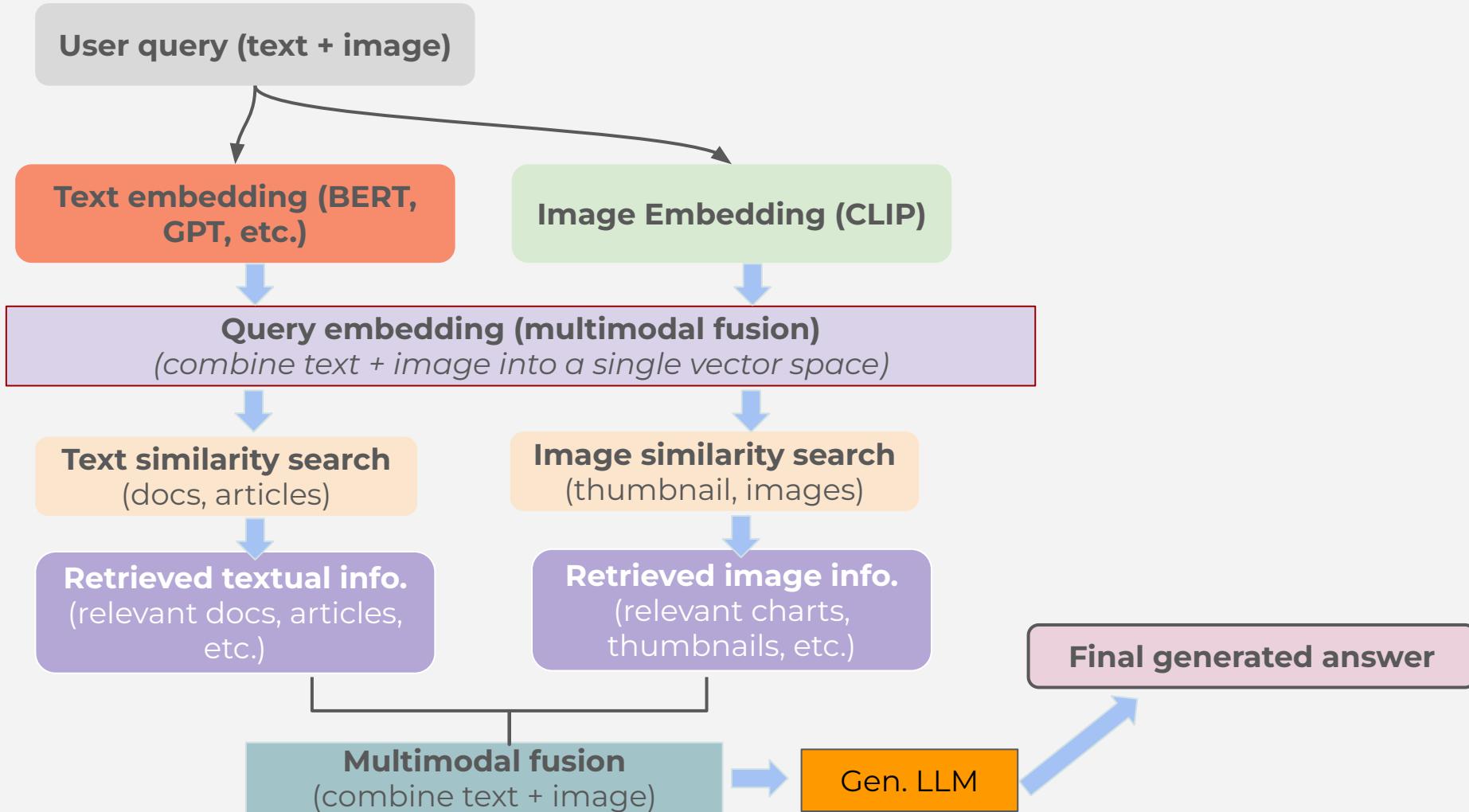


# Why multimodal RAG

- **Rich data sources**
  - Information in real world comes in multiple forms (text, images, audio, etc) - we must not overlook a vast amount of information.
- **Better context understanding**
  - Data is being pulled from different sources, increasing data contextuality
- **Complex query handling:** multimodal queries often combine different forms of input (textual, images)
- **Real-world scenarios:** real-world use cases involve multimodal data.

# *Search in a Multimodal RAG System*

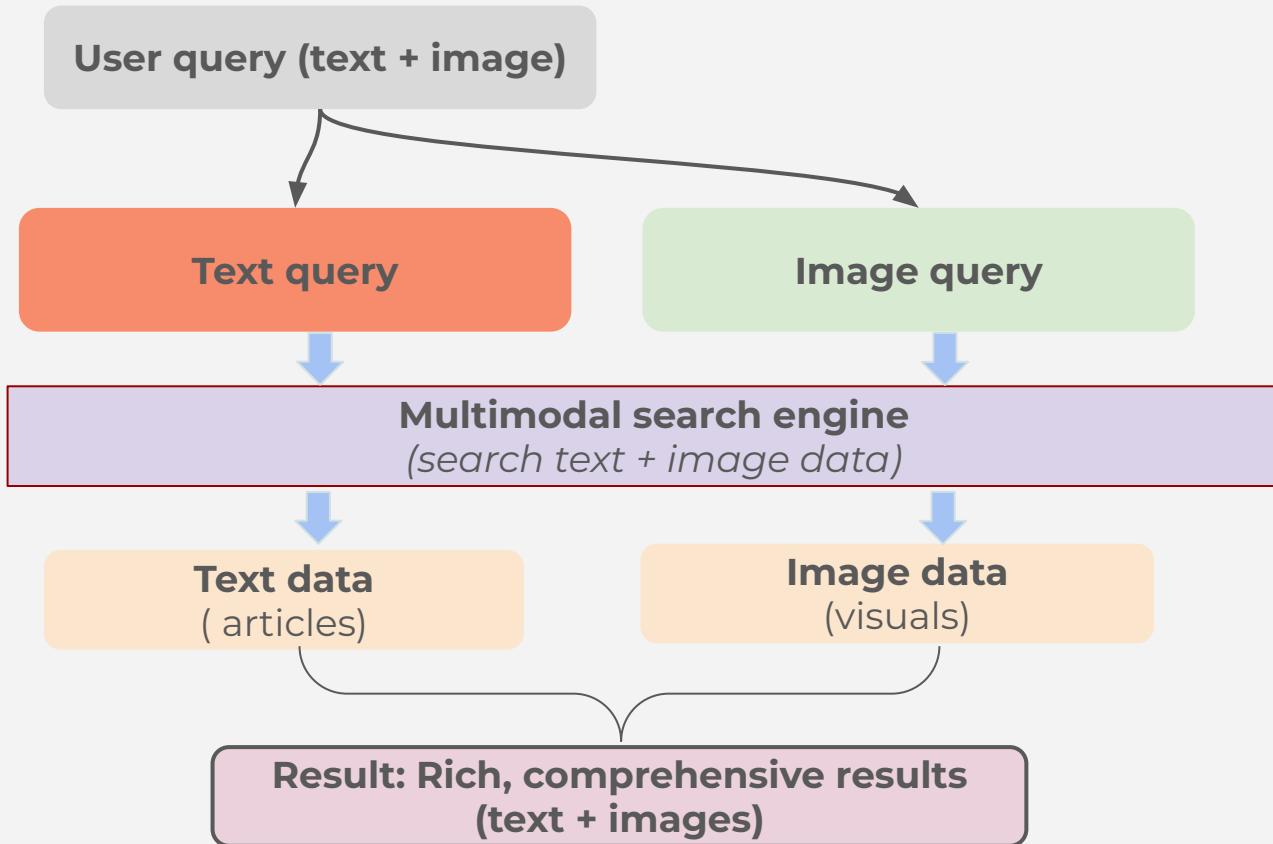
How search is integrated  
into a **multimodal RAG**  
system



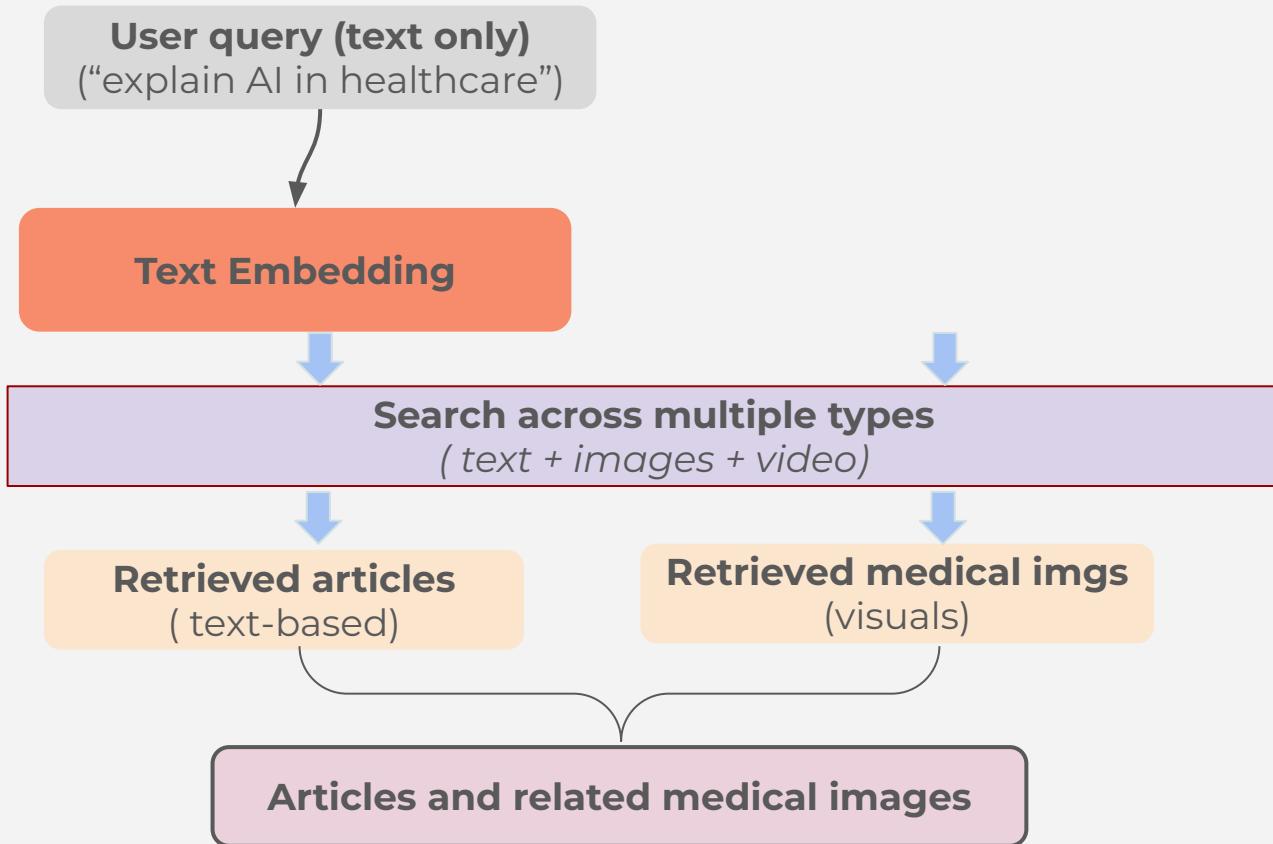
# Why multimodal search is powerful

- **Richer information sources:** various information formats - cross-modality search yields better results.
- **Cross-modal retrieval:** cross-reference modality - retrieve non-text content using text queries and vice-versa.
- **Enhanced user experience:** users can upload an image, type in text, or audio to retrieve content.
- **Improved accuracy and personalization:** the combination of different data types, multimodal search provides more accurate and contextually relevant results.
- **Real-world use cases:**
  - Medical field, education, e-commerce...

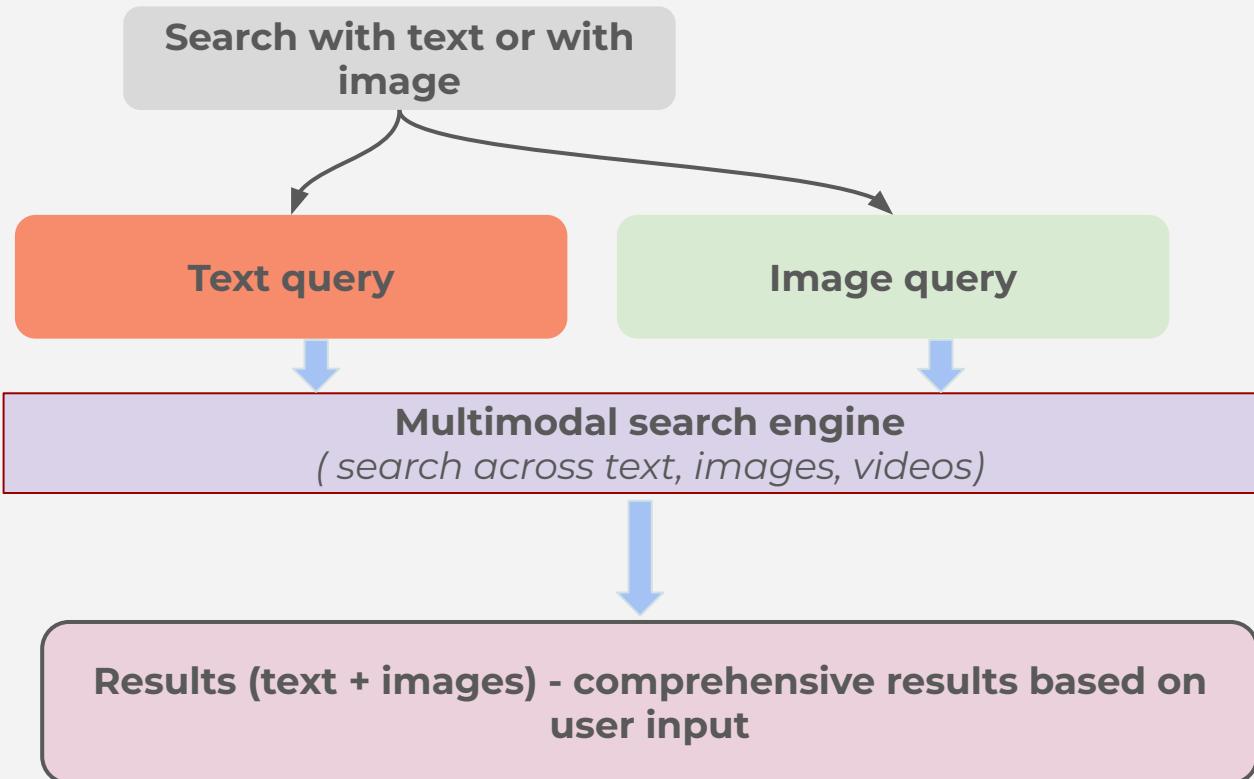
# 1. Richer information sources



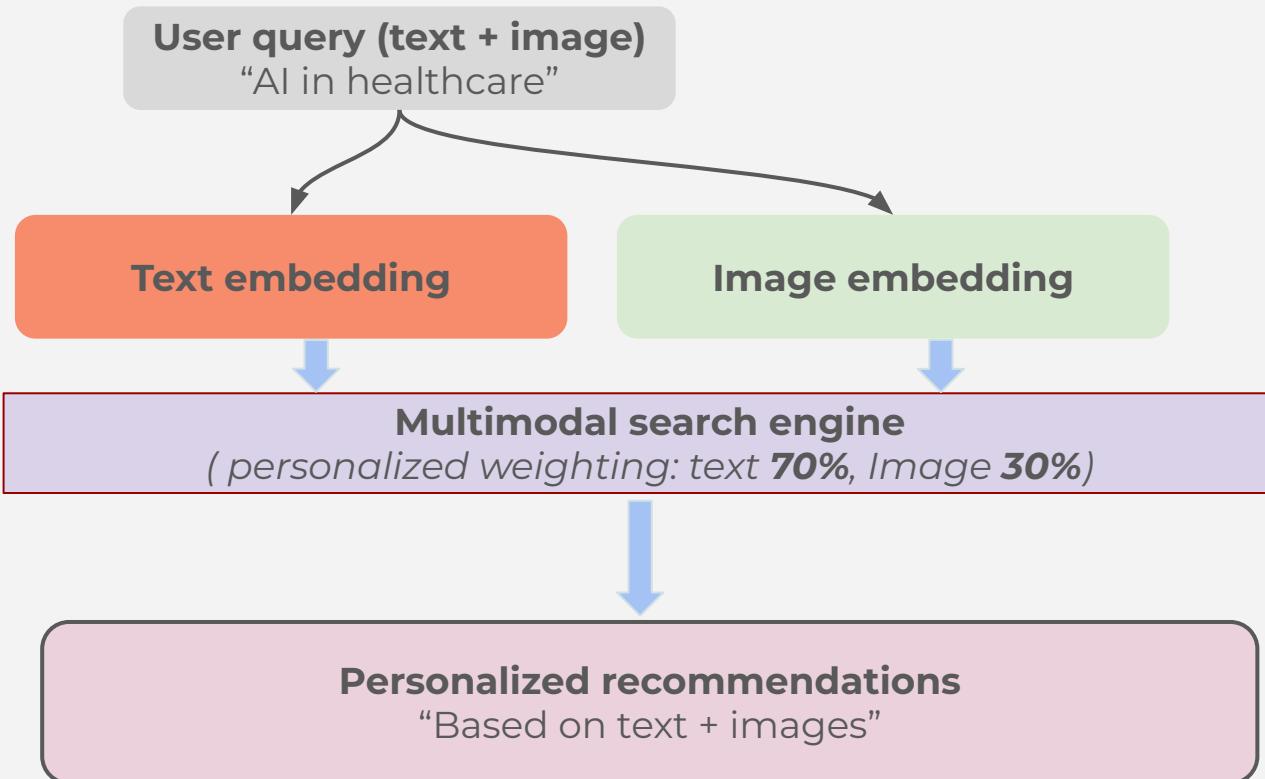
## 2. Cross-Modal Retrieval



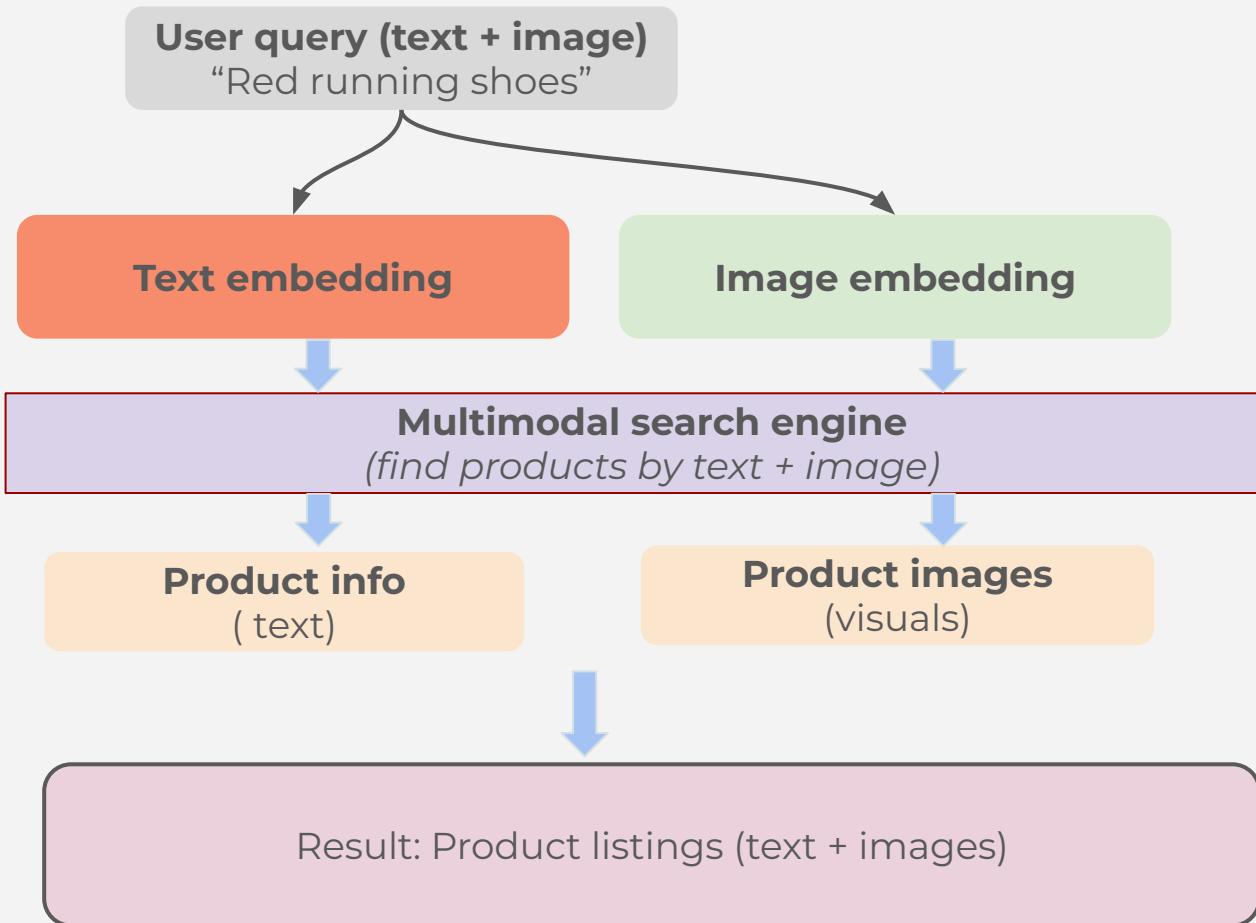
### 3. Enhanced user experience



## 4. Improved Accuracy and Personalization



# 5. Real-World Use Cases (e-commerce prod. search)



# Hands-on: Simple multimodal search

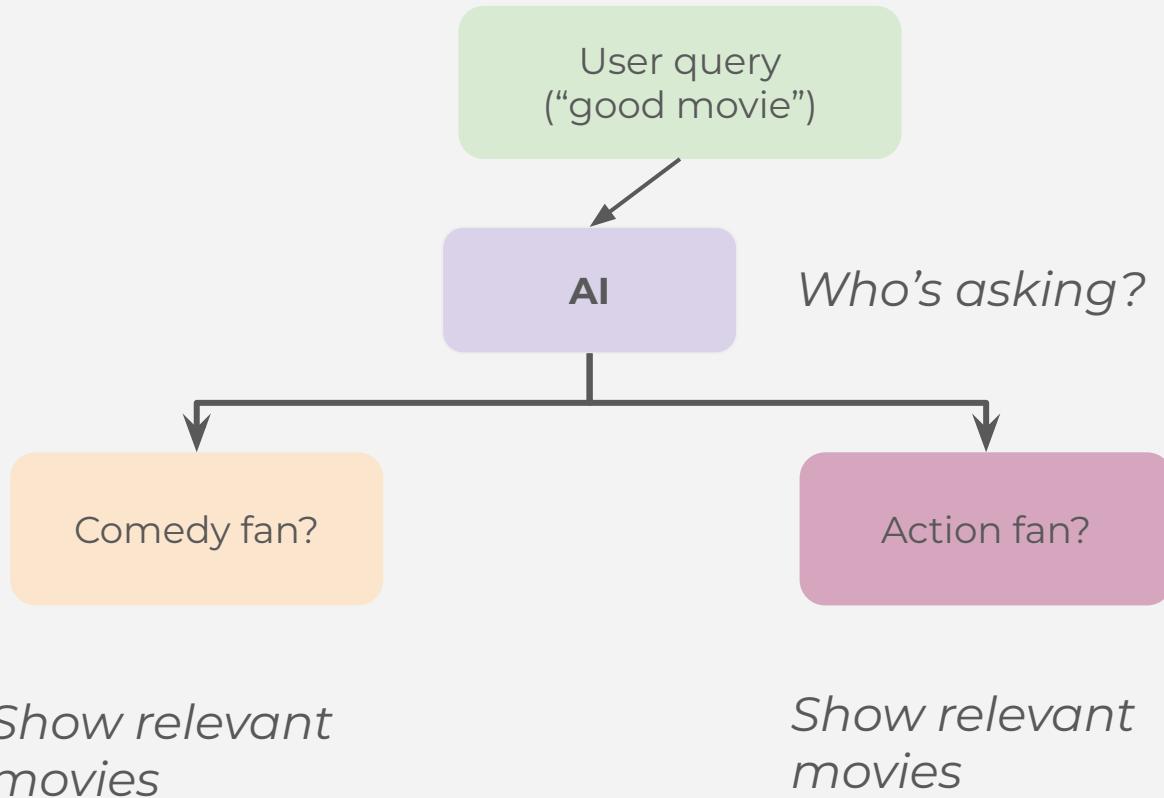
-

**Search is** *objective*

As long as the results match the search, the search is successful!

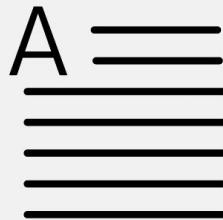
**Recommendation is** *subjective*

Always *depends on* **who** is asking the question.



# Multi-vector recommender system

What's your favorite dish?



Dish description (text)



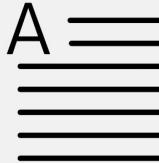
A picture of the dish

<b>Nutrition Facts</b>		
30 servings Per Container		
Serving Size 1 Pouch (0.34 FL OZ, 10mL)		
Amount Per Serving	Calories	15
	% Daily Value*	
Total Fat	0g	0%
Sodium	0mg	0%
Total Carbohydrates	4g	1%
Total Sugars	2g	
Includes 2g Added Sugars		4%
Protein	0g	0%
Not a significant source of saturated fat, trans fat, cholesterol, dietary fiber, vitamin D, calcium, iron and potassium.		
* The % Daily Value (DV) tells you how much a nutrient in a serving of food contributes to a daily diet. 2,000 calories a day is used for general nutrition advice.		

Nutrition facts

# Multi-vector recommender system

What's your favorite dish?



## Nutrition Facts

30 servings Per Container  
Serving Size 1 Pouch (0.34 FL OZ, 10mL)

Amount Per Serving	Calories	15
Total Fat	0g	0%
Sodium	0mg	0%
Total Carbohydrates	4g	1%
Total Sugars	2g	
Includes 2g Added Sugars		4%
Protein	0g	0%

Not a significant source of saturated fat, trans fat, cholesterol, dietary fiber, vitamin D, calcium, iron and potassium.

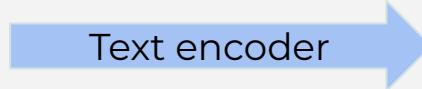
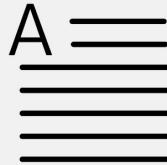
\* The % Daily Value (DV) tells you how much a nutrient in a serving of food contributes to a daily diet. 2,000 calories a day is used for general nutrition advice.



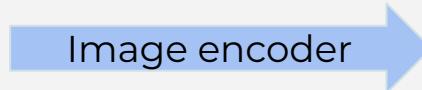
Personalized preference

# Multi-vector recommender system

## Specialized embedding models

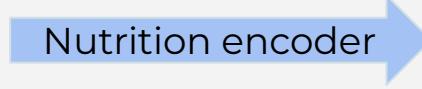


[ 0.87, ... 0.774 ]



[ 0.027, ... 0.004 ]

Nutrition Facts	
30 servings Per Container	
Serving Size 1 Pouch (0.34 FL OZ, 10mL)	
Amount Per Serving	Calories 15
	% Daily Value*
Total Fat	0g
Sodium	0mg
Total Carbohydrates	4g
Total Sugars	2g
Includes 2g Added Sugars	4%
Protein	0g
% Daily Value	
*Not a significant source of saturated fat, trans fat, cholesterol, dietary fiber, vitamin D, calcium, iron and potassium.	
*The % Daily Value (DV) tells you how much a nutrient in a serving of food contributes to a daily diet. 2,000 calories a day is used for general nutrition advice.	



[ 0.17, ... 0.304 ]

# This is important because...

Multimodal recommendation allows users to choose what they want in different forms/modalities

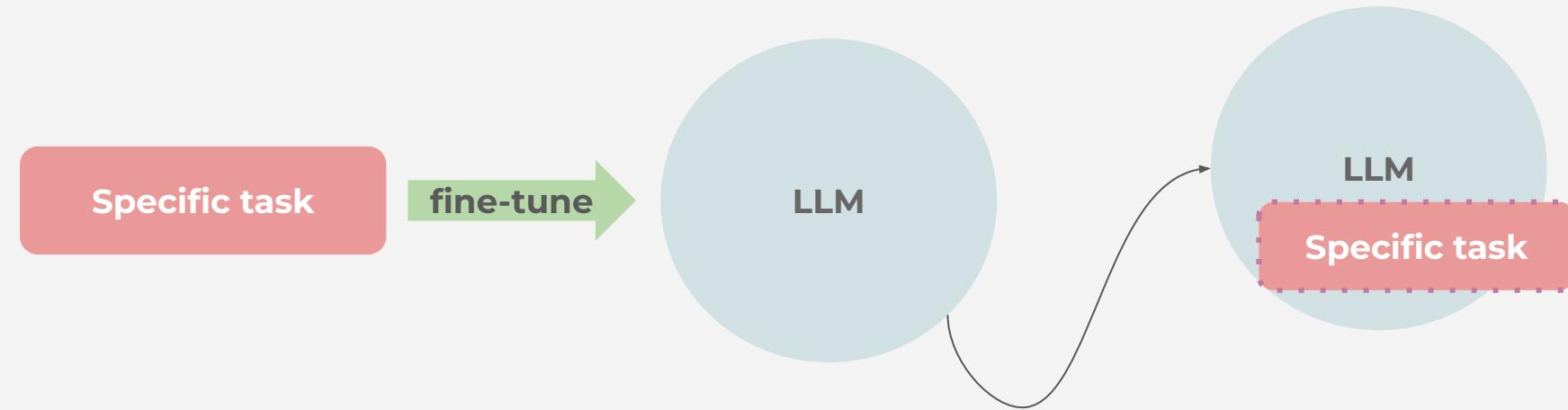
- Some users might order a dish by looking at a picture of a dish
- Or by reading the description of the dish
- Or by the list of nutrition facts

# Fine-tuning

# Fine-tuning

What is Fine-Tuning?

- **Adapt a pre-trained model** to a new, specific task



# Why fine-tuning?

**Efficiency** - training a model from scratch is **expensive!**

- Fine-tuning **reduces** these requirements

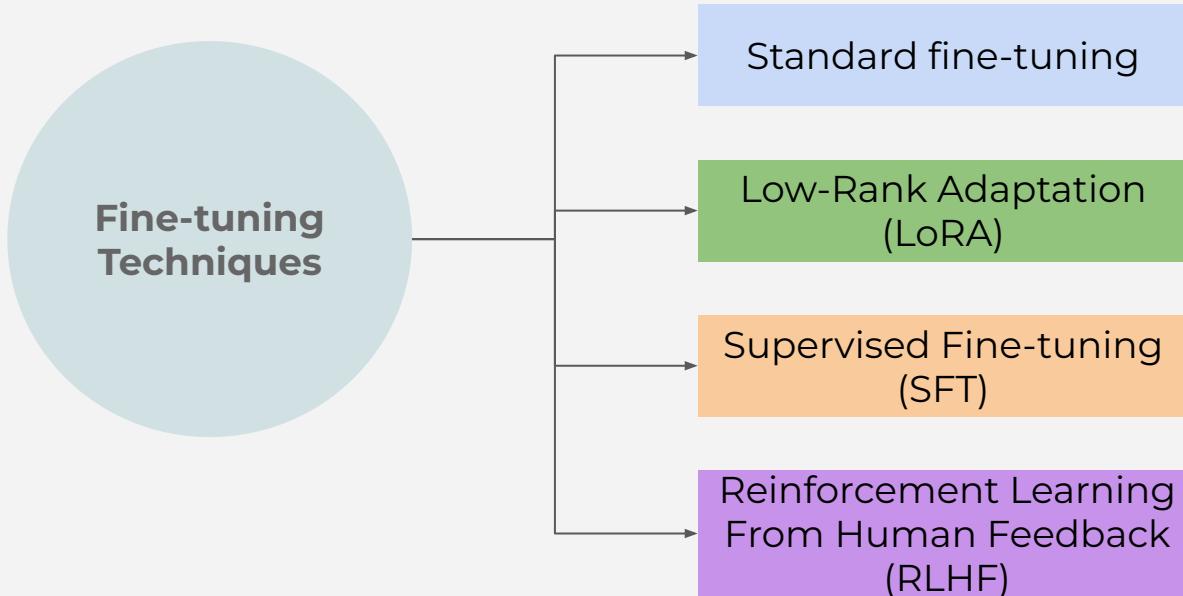
## Performance

- Pre-trained models already have pre-knowledge from large datasets
- Fine-tuning **adapts** these features to specific tasks, which leads to better performance.

## Data scarcity

- If dataset is small, fine-tuning helps achieve good results without needing millions of examples

# Fine-tuning techniques



Standard fine-tuning

Low-Rank Adaptation  
(LoRA)

- Adjusts all the parameters in LLM to increase performance to a specific task. Extremely **effective**, but also **expensive!**
- Parameter-efficient fine-tuning -- instead of being intrusive (updating all weights), it only adds a small, low-rank matrices to specific layer -- reducing the number of parameters that need to be updated.

## Supervised Fine-tuning (SFT)

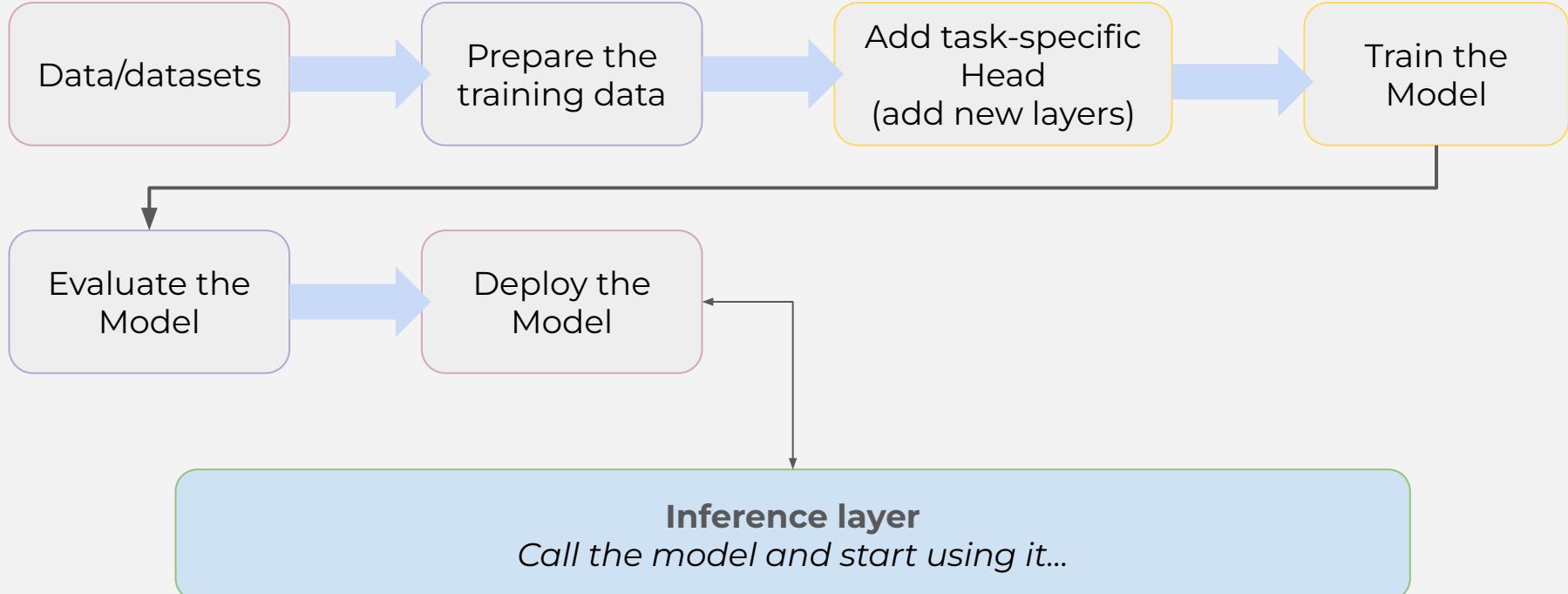
## Reinforcement Learning From Human Feedback (RLHF)

- It trains a base model on a new dataset under supervision (incl. Demonstration data, prompts, responses...)
- Train models with the help of humans -- facilitates continuous improvement based on human input.

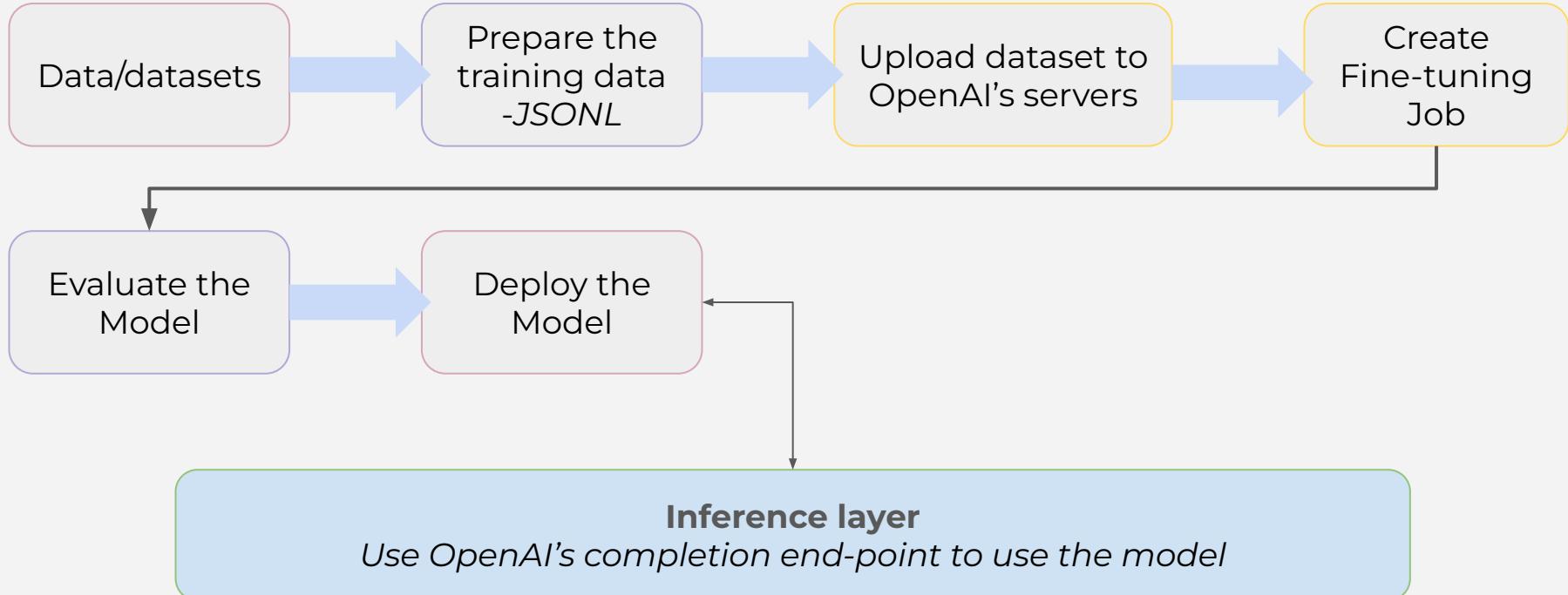
## Comparison of Techniques

Technique	Key Idea	When to Use
Standard Fine-Tuning	Update all (or some) weights of the pre-trained model.	Similar tasks, medium to large datasets.
LoRA	Add low-rank matrices to adapt specific layers.	Limited resources, multi-task adaptation, avoiding catastrophic forgetting.
SFT	Fine-tune using labeled data and supervised learning.	Labeled datasets, tasks requiring high accuracy.
RLHF	Fine-tune using reinforcement learning and human feedback.	Aligning outputs with human preferences, dialogue generation, content creation.

# General Fine-tuning process



# Fine-Tuning Using the OpenAI API

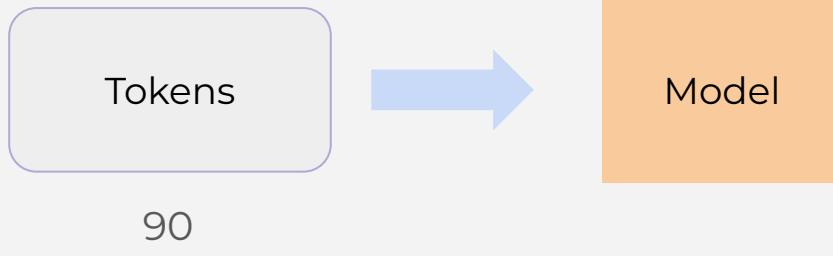


# Costs of Fine-tuning (OpenAI)

Check out the docs:

<https://platform.openai.com/docs/guides/fine-tuning>

# Costs of Fine-tuning (OpenAI)



1. API cost
2. Prompt length
  - a. **Context window** → *how many tokens a model can process.*

# Costs of Fine-tuning (OpenAI)

Tokens and characters:

GPT-4o & GPT-4o mini    GPT-3.5 & GPT-4    GPT-3 (Legacy)

This is a short sentence to demonstrate the difference between tokens and characters.

**Clear** **Show example**

Tokens	Characters
14	85

This is a short sentence to demonstrate the difference between tokens and characters.

# Costs of Fine-tuning (OpenAI) - Price estimation

1. Training the fine-tuned model
2. Using the fine-tuned model (inference)
  - a. **Visit:**  
<https://openai.com/api/pricing/> for up-to-date pricing (scroll down)

# Fine-tuning hands-on (OpenAI)

# **Fine-tuning hands-on (LoRA) - Open-source models**

# *LoRA Deep Dive*

- Understanding PEFT
- Fine-tuning LLMs with LoRA

# What is PEFT

PEFT

Parameter-Efficient Fine-Tuning - technique for fine-tuning only a small subset of a model's parameters. (***reduces memory and computational costs***)

# Benefits of LoRA?

LoRA

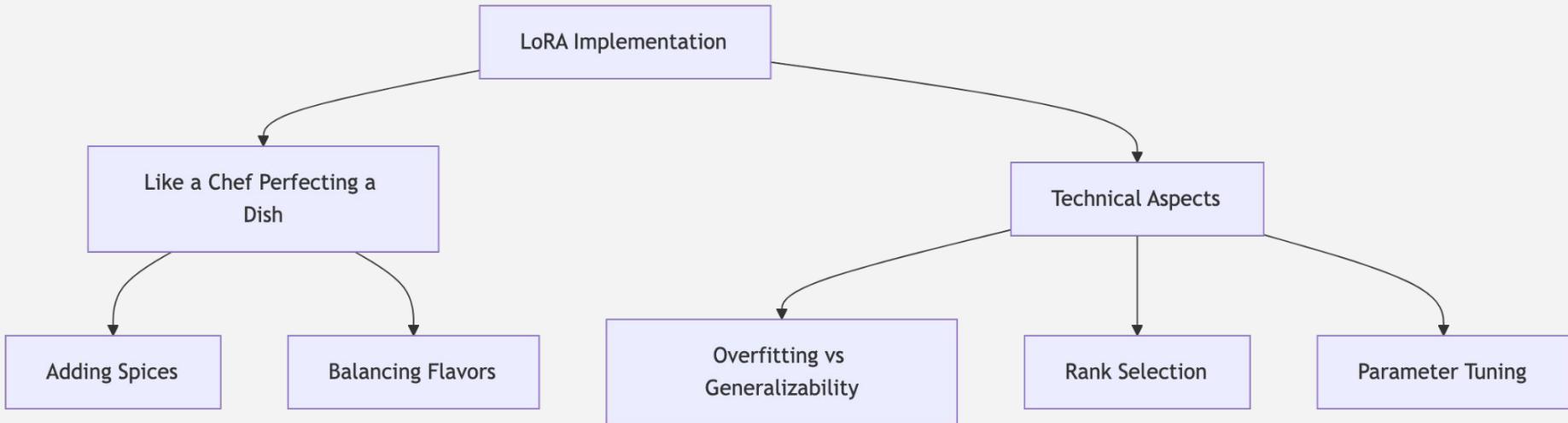
1. **Parameter Efficiency:**
  - a. LoRA adds only a small number of trainable params
  - b. Making fine-tuning feasible on smaller hardware setups.
2. **Memory Efficiency:**
  - a. Reduces memory during training.
3. **Task-specific Adaptation:**
  - a. Allows the model to adapt to new tasks without overwriting the pre-trained knowledge.
4. **Scalability**

# When to use LoRA?

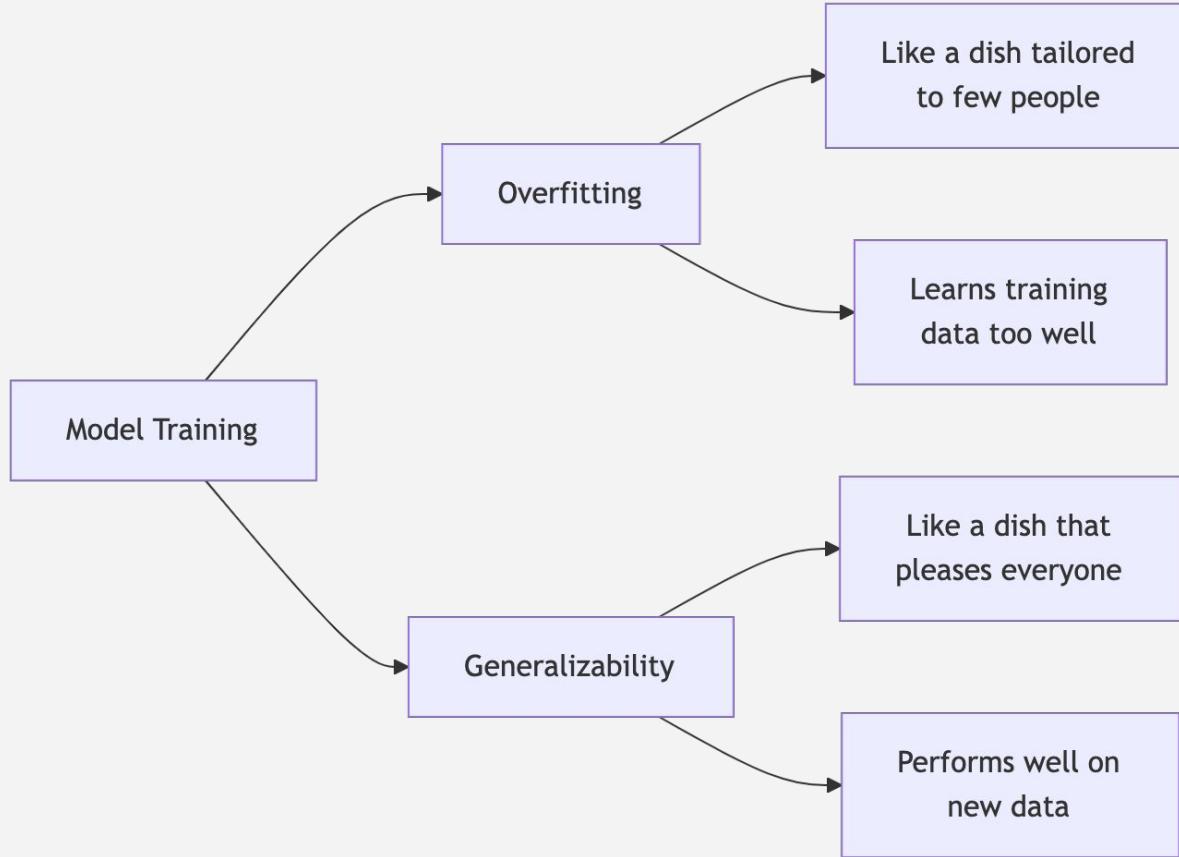
LoRA

1. **Limited compute resources:** no access to high-end GPUs or TPUs
2. **Multi-task Learning:**
  - a. Need to fine-tune a single model for multiple tasks w/o storing separate copies of the model
3. **Avoiding catastrophic forgetting:**
  - a. Want to retain the pre-trained knowledge while adapting to new tasks

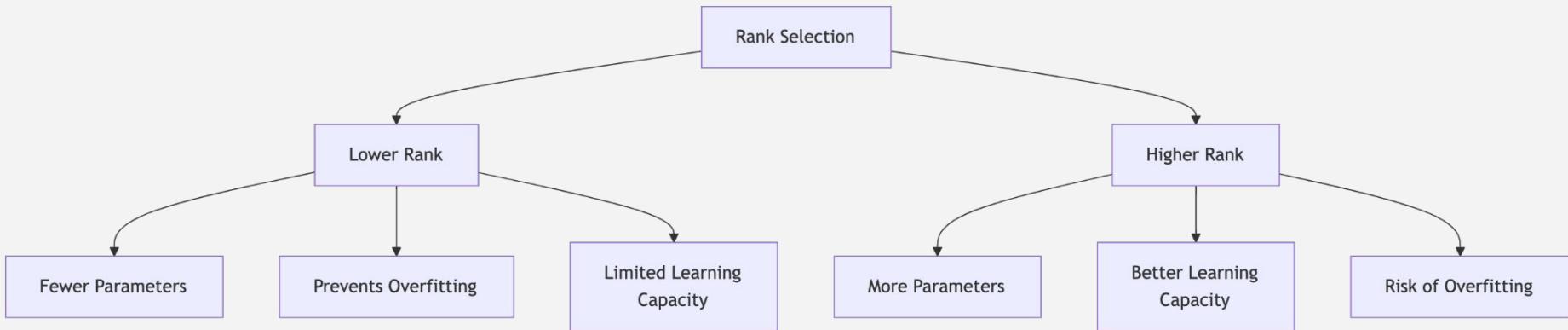
# LoRA Deep Analysis



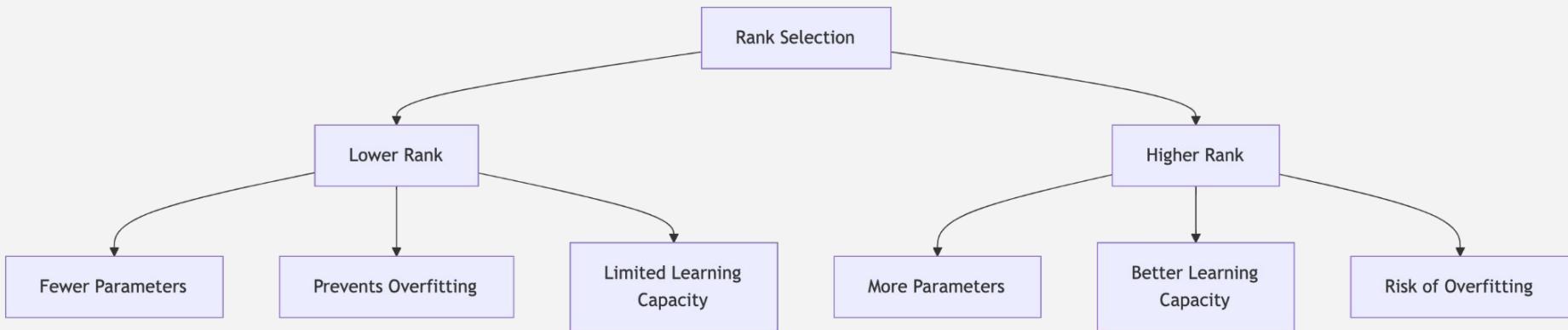
# LoRA Deep Analysis



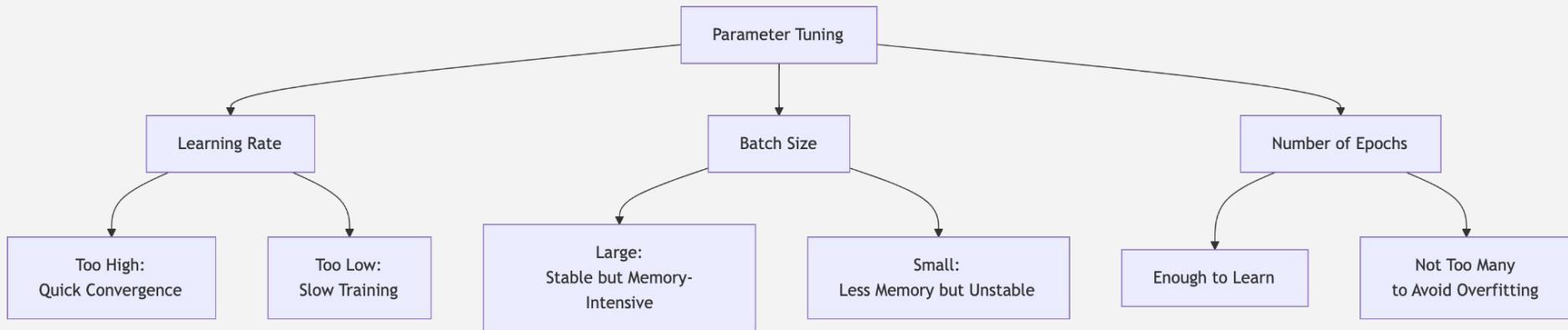
# LoRA Deep Analysis



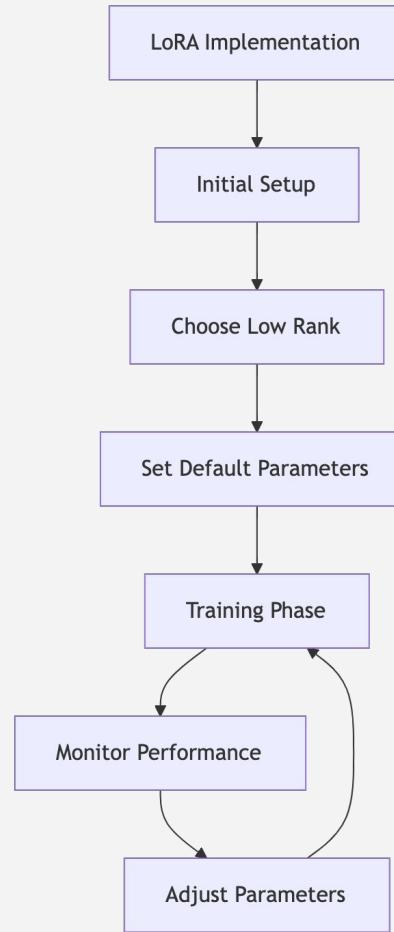
# LoRA Deep Analysis



# LoRA Deep Analysis



# Implementation Strategy

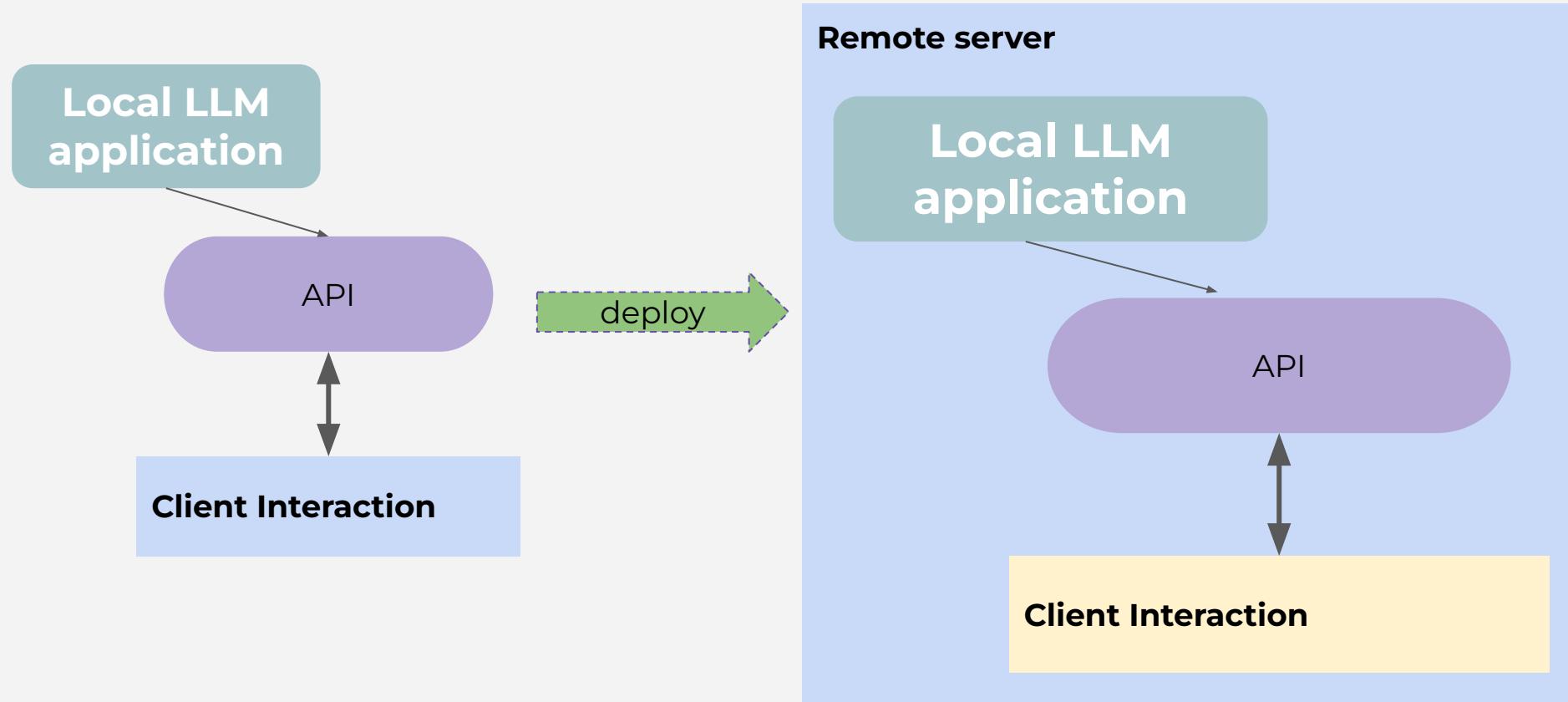


# **Best Practices**

# LoRA Hands-on?

Check the *fine-tuning-lora* directory for code!

# End-to-end (E2E) LLM application



# *Congratulations!*

You made it to the end!

- Next steps...

# Course Summary

- Generative AI Mastery & LLM Development
  - GenAI
  - LLMs (Large Language Models)
  - Deep Learning\*
  - Machine Learning\*
  - Prompting
  - RAG
  - Agents
  - Fine-tuning
  - Building AI/LLM-based Applications
  - **Comprehensive Course**

# Wrap up - Where to Go From Here?

- Keep learning
  - Get more ideas and build more applications and test different,
- Read more papers on RAG, Fine-tuning, LLMs, Building AI-based solutions...
- Challenge yourself to keep learning new skills!