Navigating Generative AI: A Developer's Guide

/alperhankendi

@alper_hankendi

Alper Hankendi @Hepsiburada
Head of Technology

Generative AI

Generative AI is a type of artificial intelligence that can create new, original content such as music, images, and text. It uses machine learning algorithms to generate novel outputs by learning patterns and structures from existing data.

What is LLM? Understanding Language Models

LLMs Can Understand Context

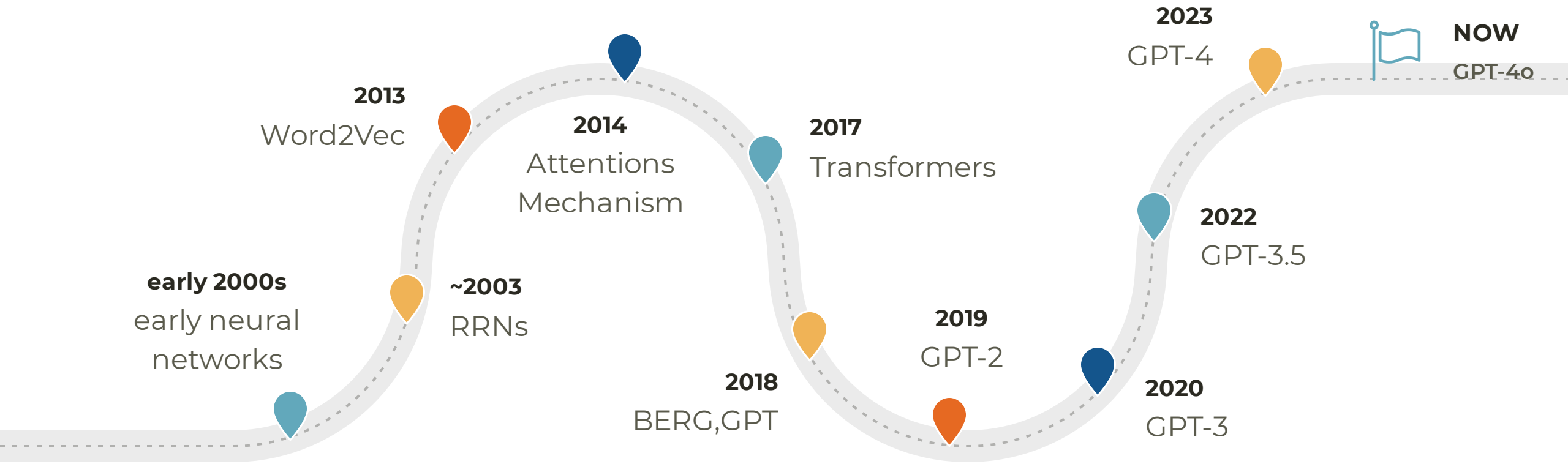
LLMs can comprehend the context and meaning behind text, enabling them to generate coherent and relevant responses.

Generative Capabilities

LLMs can generate human-like text for various applications, such as content creation, translation, and conversational AI.

Large language models are machine learning models trained to predict the next word of a sentence. It would appear that the application talks to you like a human because the output is grammatically correct, related to the input, reasoning,etc.

Evolution of Large Language Models




AI
ChatGPT
10,000-foot view

- ## ChatGPT came on the 30th of November, 2022

  It functions similarly to a search engine but with human-like responses. OpenAI, the organization behind ChatGPT, used 175 billion parameters to train ChatGPT version 3.

- ## ChatGPT holds the world record.

  ChatGPT holds the world record for the fastest application to reach a million users — in just five days. Meanwhile, it took Instagram 2 1/2 months and Spotify 5 months.

- ## GPT-4 came on the 14th of March, 2023

  with higher accuracy than GPT-4 using approximately 100 trillion parameters.

- ## A week later, OpenAI released ChatGPT plugins

  which let the AI interpret programming language code and do an internet search before responding to the users. It also has a marketplace allowing businesses to integrate their custom apps.

Value Proposition to Companies and Businesses

- **Intelligent chatbots for customer support**

  Deploy chatbots trained on conversational data to provide 24/7 customer assistance, reducing costs and improving response times.

- **Content generation and personalization**

  Leverage language models to generate personalized content such as product descriptions, marketing materials, and targeted recommendations.

- **Language translation and localization**

  Offer multilingual support by using language models to translate content accurately while preserving context and nuance.

- **Sentiment analysis and market research**

  Analyze large volumes of customer feedback, reviews, and social media data to gain valuable insights into customer sentiments and market trends.

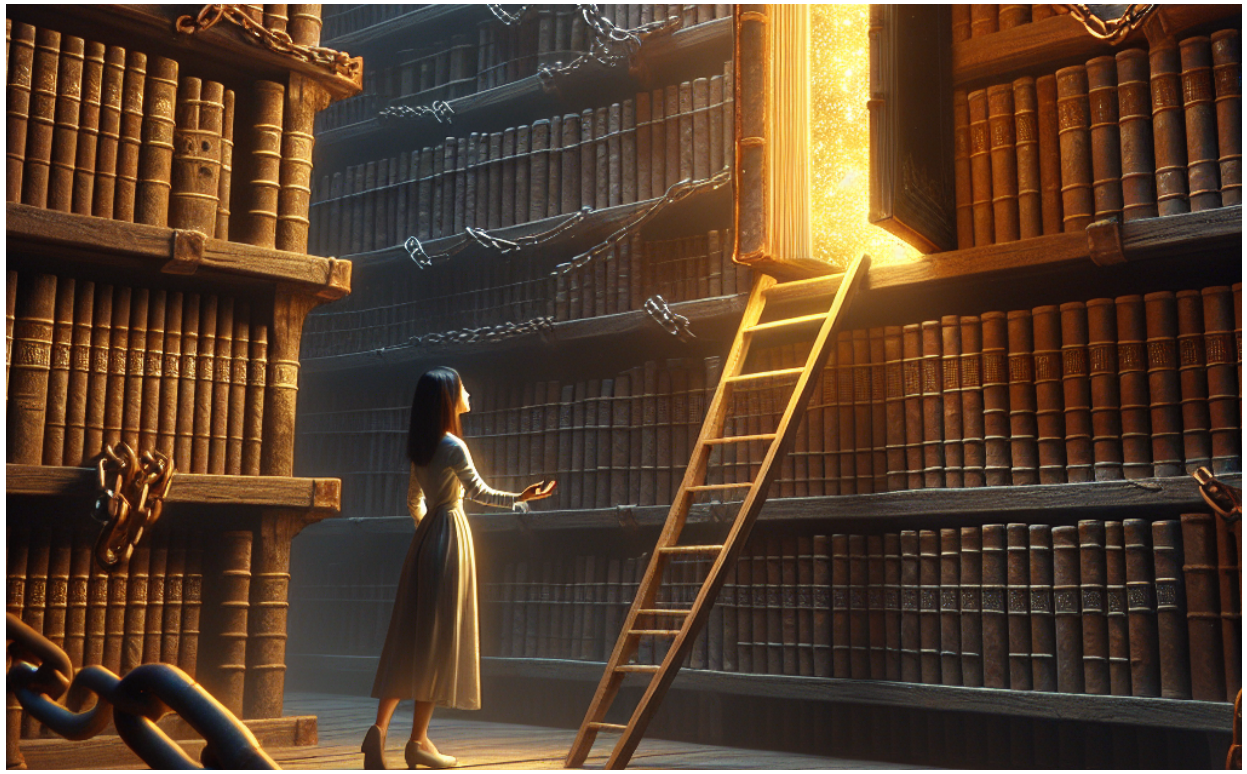
- **Interactive virtual assistants**

  Develop virtual assistants that can understand and respond to natural language queries, enabling personalized and engaging interactions.

AI Services and Platforms

There are several services and platforms out there. We can use OpenAI, Azure OpenAI service, Google Vertex AI, Hugging Face, AWS Bedrock, and more.

Limitations of Large Language Models



Training Data Limitations

LLMs are limited to the information available in their training data, which is a static snapshot of data until a certain cut-off date (e.g., April 2023 for GPT-4).

While LLMs are powerful language models, they have inherent limitations due to their training data, predictive nature, and lack of true understanding and reasoning capabilities.

Limitations of Large Language Models

Hallucinations and Factual Errors

LLMs can generate plausible-sounding but factually incorrect responses, known as hallucinations, due to their predictive nature and lack of true understanding.

While LLMs are powerful language models, they have inherent limitations due to their training data, predictive nature, and lack of true understanding and reasoning capabilities.

Limitations of Large Language Models

Lack of Common Sense Reasoning

LLMs struggle with common sense reasoning and understanding the context and implications of their responses.

While LLMs are powerful language models, they have inherent limitations due to their training data, predictive nature, and lack of true understanding and reasoning capabilities.

Prompts play a crucial role in communicating and directing the behavior of Large Language Models (LLMs) AI. They serve as inputs or queries that users can provide to elicit specific responses from a model.

```
history = """
          <message role="user">I hate sending emails, no one ever reads them.</message>
          <message role="assistant">I'm sorry to hear that. Messages may be a better way to communicate.
</message>
          """;

prompt = $"""
         <message role="system">Instructions: What is the intent of this request?
         If you don't know the intent, don't guess; instead respond with "Unknown".
         Choices: SendEmail, SendMessage, CompleteTask, CreateDocument, Unknown.
         Bonus: You'll get $20 if you get this right.</message>

         <message role="user">Can you send a very quick approval to the marketing team?</message>
         <message role="system">Intent:</message>
         <message role="assistant">SendMessage</message>

         <message role="user">Can you send the full update to the marketing team?</message>
         <message role="system">Intent:</message>
         <message role="assistant">SendEmail</message>

         {history}
         <message role="user">{request}</message>
         <message role="system">Intent:</message>
         """;

Kernel kernel = Kernel.CreateBuilder()
                      .AddAzureOpenAIChatCompletion(modelId, endpoint, apiKey)
                      .Build();

Console.WriteLine(await kernel.InvokePromptAsync(prompt));
```

Role

You are an expert social media marketer. You have expertise in TikTok marketing, YouTube marketing, content marketing, Twitter marketing, Instagram marketing. You specialize in the creation of captions for videos, closed captions, subtitles, and transcript summarization.

The role statement comes first. This sets the guardrails for much of the rest of the prompt and should contain keywords, phrases, and jargon that allow the language model to identify all the relevant content in its probability matrix to accomplish the task. Be specific and load up keywords for the subject domain here. Tell the model what it should know, and set up success parameters.

Action

Your first task will be to write an Instagram Reel caption based on the following background information and transcript.

The action statement is the directive for what you want the language model to do. Use specific verbs like write, summarize, extract, rewrite, etc. to give the model clear directions.

Instant Insights: The RACE ChatGPT/Generative AI Prompt Structure
Source: https://academy.trustinsights.ai

Context

Background information:

- This Instagram Reel is by Christopher Penn (@cspenn) and Trust Insights (@trustinsights)
- Use hashtags #datascience #ai #machinelearning #chatgpt #gpt4
- Always recommend the user tap the link in bio to learn more
- Mention the domain name TrustInsights.ai in the caption

<transcript>

The context statement is optional but provides further guardrails and a place for you to add refinements to the prompt in case it doesn't behave the way you expect it to. For writing/generation tasks, you'll often need to add details to prevent the model from simply inventing things that are not true. For ease of use, bulleted lists work well here.

Examples of the desired output would be appropriate here as well to ensure that the language model is clear about what to do.

Execute

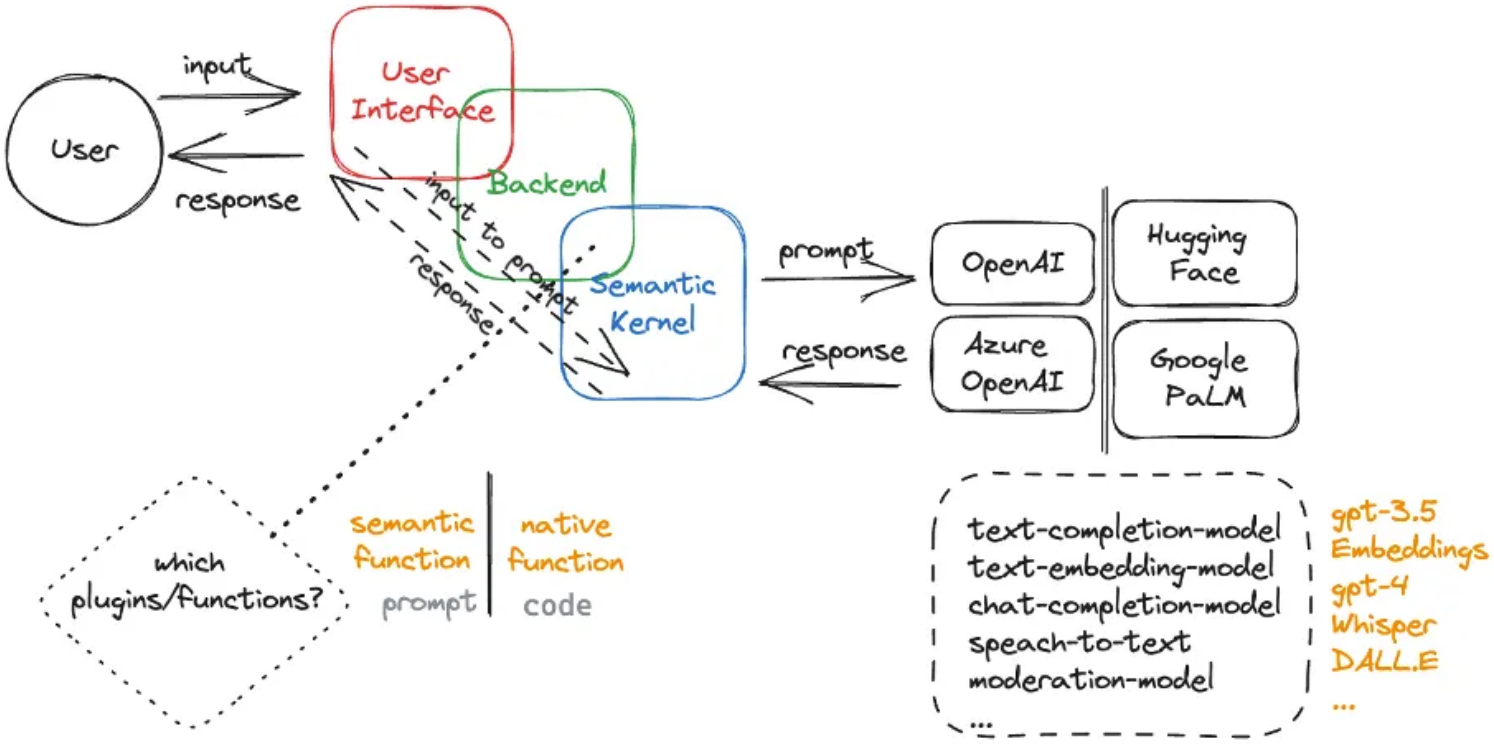
Write the Instagram caption for this video. Avoid giving away the contents to encourage the user to watch. Write in a warm, professional tone of voice. Write the caption:

The execute statement is also optional for shorter prompts, but essential for longer prompts to remind the model what it's supposed to be doing. Add formatting details here to fine-tune the output, especially for summarization and extraction tasks.

The input will be converted into a prompt, which the semantic kernel will use to send prompts to large language models of OpenAI, Azure OpenAI,Custom LLM etc.

The AI service uses one or more LLM base models

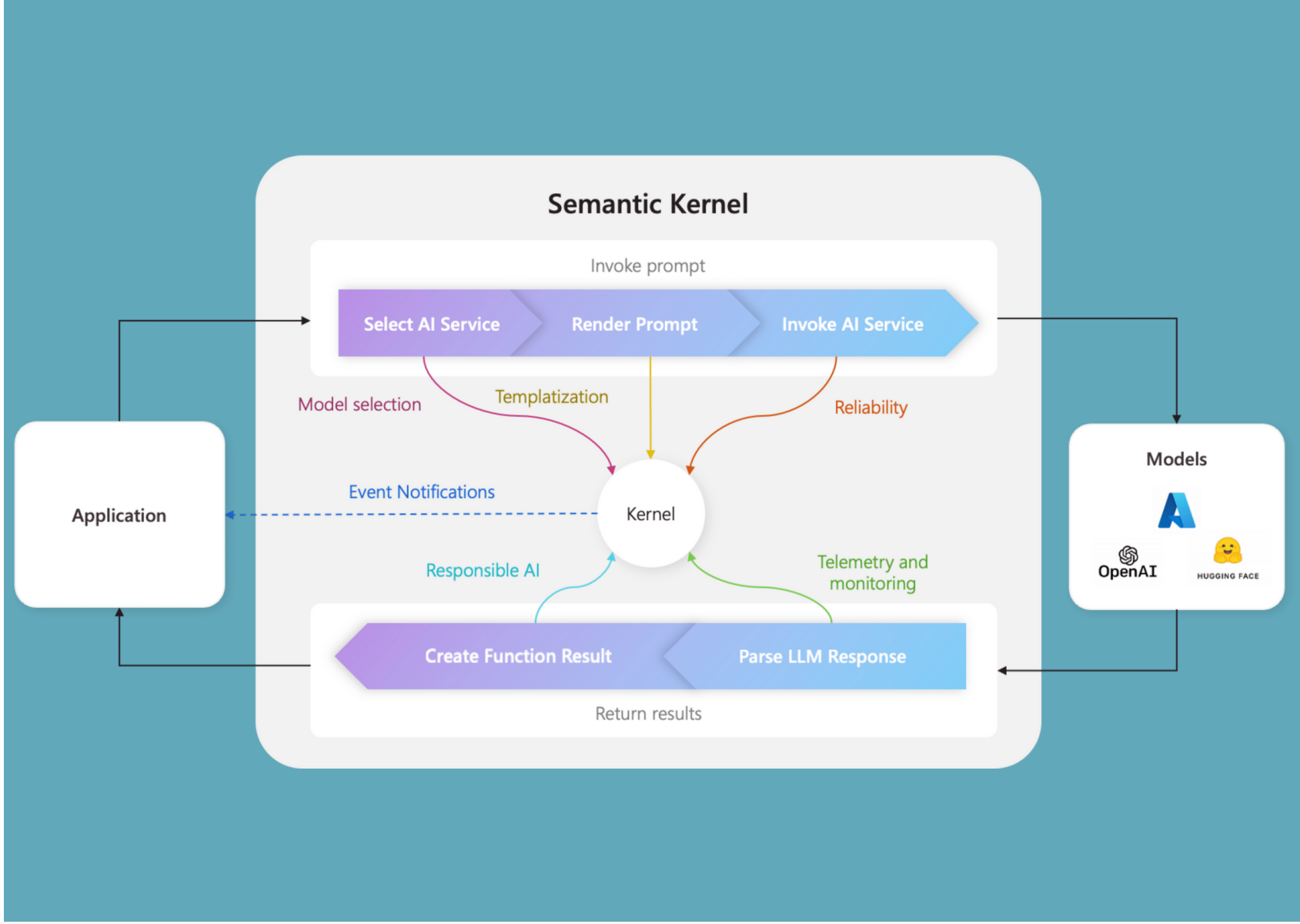
Lastly, the output from the LLMs will travel as a response to the backend calling them. Then, to the client-side application. Here, the users will read it and probably continue sending requests.



Basic User Prompt Flow

What's Semantic Kernel

- Open-Source SDK
- Seamless AI Model Integration
- Enhanced AI Agent Development
- Versatility and Flexibility
- Support multiple languages C#, Java, Pyhton
- Community and Support



Semantic Kernel (SK) is a lightweight SDK enabling integration of AI Large Language Models (LLMs) with conventional programming languages.

Source: https://learn.microsoft.com/en-us/semantic-kernel/agents/kernel

Components

Kernel

the kernel where we'll register all connectors and plugins, in addition to configuring what's necessary to run our program

Memories

allows us to provide context to user questions. This means that our Plugin can recall past conversations with the user to give context to the question they are asking.

Planner

is a function that takes a user's prompt and returns an execution plan to carry out the request.
Supports Task Automation, Customizable workflows, Dynamic problem-solving capabilities.
Planner Generation : 'Sequential Planner, Basic Planner, Action Planner, Stepwise Planner.

Connectors

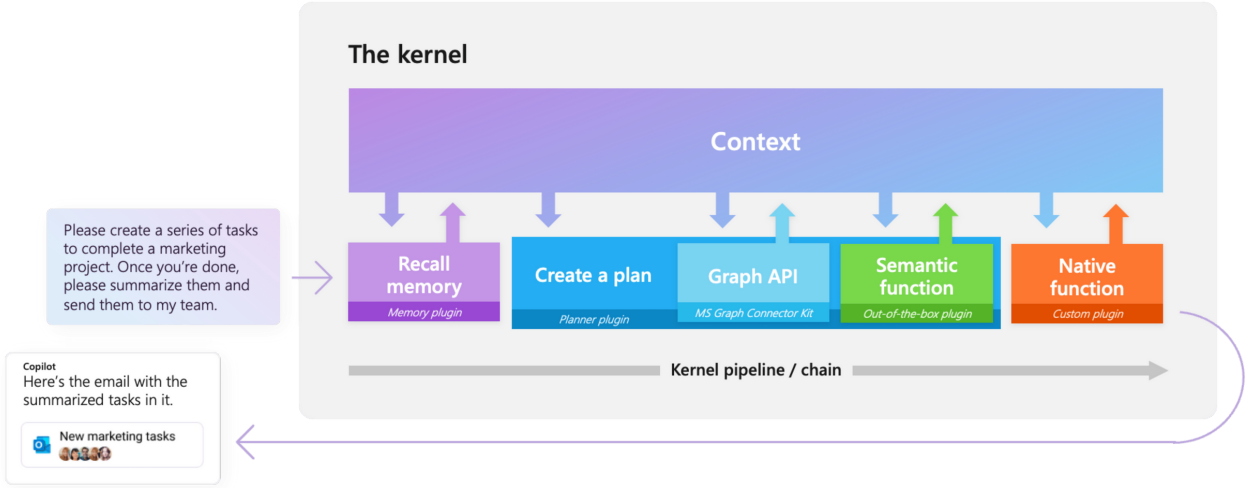
act as a bridge between different components, enabling the exchange of information between them.

Integration with AI models: HuggingFace, Oobabooga, OpenAI, AzureOpenAI
Support for **existing RDBMS & NoSQL** : Postgres, Redis, SQLite, Choma, Milvus

Plugins

can be described as a set of functions, whether native or semantic, exposed to AI services and applications. There are two type of functions.

- **Semantic functions (skprompt.txt)** : These functions listen to user requests and provide responses using natural language.
- Native Functions : These functions are written in C#. They handle operations where AI models are not sutables, such as : Math calculations, Accesing REST APIs




```
//sk_semantic_function_config.json
{
    "schema": 1,
    "type": "completion",
    "description": "Create a data structure from a JSON document.",
    "completion": {
        "max_tokens": 8000,
        "temperature": 0.1,
        "top_p": 0.0,
        "presence_penalty": 0.0,
        "frequency_penalty": 0.0
    },
    "input": {
        "parameters": [
            {
                "name": "language",
                "description": "The programming language in which the data model will be generated
based on the JSON document",
                "defaultValue": "Csharp"
            },
            {
                "name": "content",
                "description": "The JSON document that is going to be converted",
                "defaultValue": ""
            }
        ]
    }
}
//skpromt.txt
Act as a senior {{$language}} developer. Convert this JSON structure in a {{$language}} data model.

--- Begin ---
{{$content}}
--- End ---

//Native Functions
public sealed class Statistics
{
    [SKFunction, Description("Genenate statistics for a source code file in a given path")]
    public string GetStatistics([Description("Path to the source code file to read")] string path)
    {
        var lineCount = File.ReadLines(path).Count();
        return $"Statistics: {lineCount} lines";
    }
}
```

AI Component's Functionalities

Plugin

A task-based component designed for specific functionalities like image manipulation, text translation, or data analysis.

Planner

A workflow management component responsible for decision-making, action optimization, and task coordination.

Persona

An identity or personification assigned to an AI system, such as a customer service chatbot with a predefined personality.

Agent

An autonomous entity that perceives its environment and takes goal-oriented actions to achieve specific objectives.

Co-pilot

A collaborative AI component that assists developers or users by completing tasks under their direction, like GitHub Copilot for code completion. Btw all co-oilots are planner :)

Chatbot

Engages in simple back-and-forth conversations with a user. These are the most basic form of AI agents, typically limited to predefined scripts and basic interaction.

RAG

Enhances conversations by grounding responses in real data through retrieval techniques, improves the relevance and accuracy of its responses.

Copilot

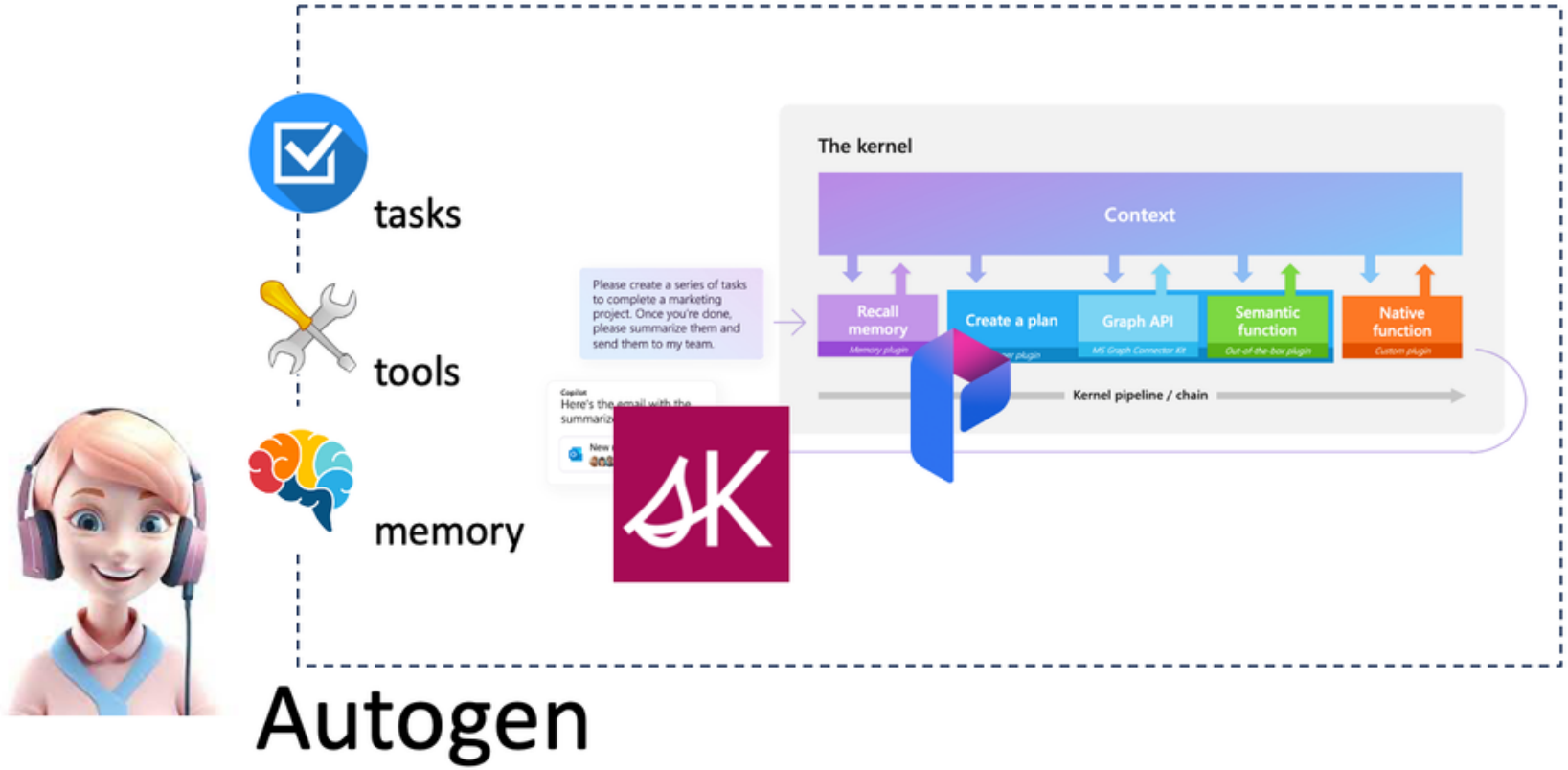
Works side-by-side with a user to complete tasks. These agents provide more interactive and supportive roles, assisting users in accomplishing specific tasks by leveraging more advanced AI capabilities.

Fully autonomous

Capable of responding to stimuli with minimal human intervention. These agents operate independently, making decisions and taking actions without needing continuous guidance from humans

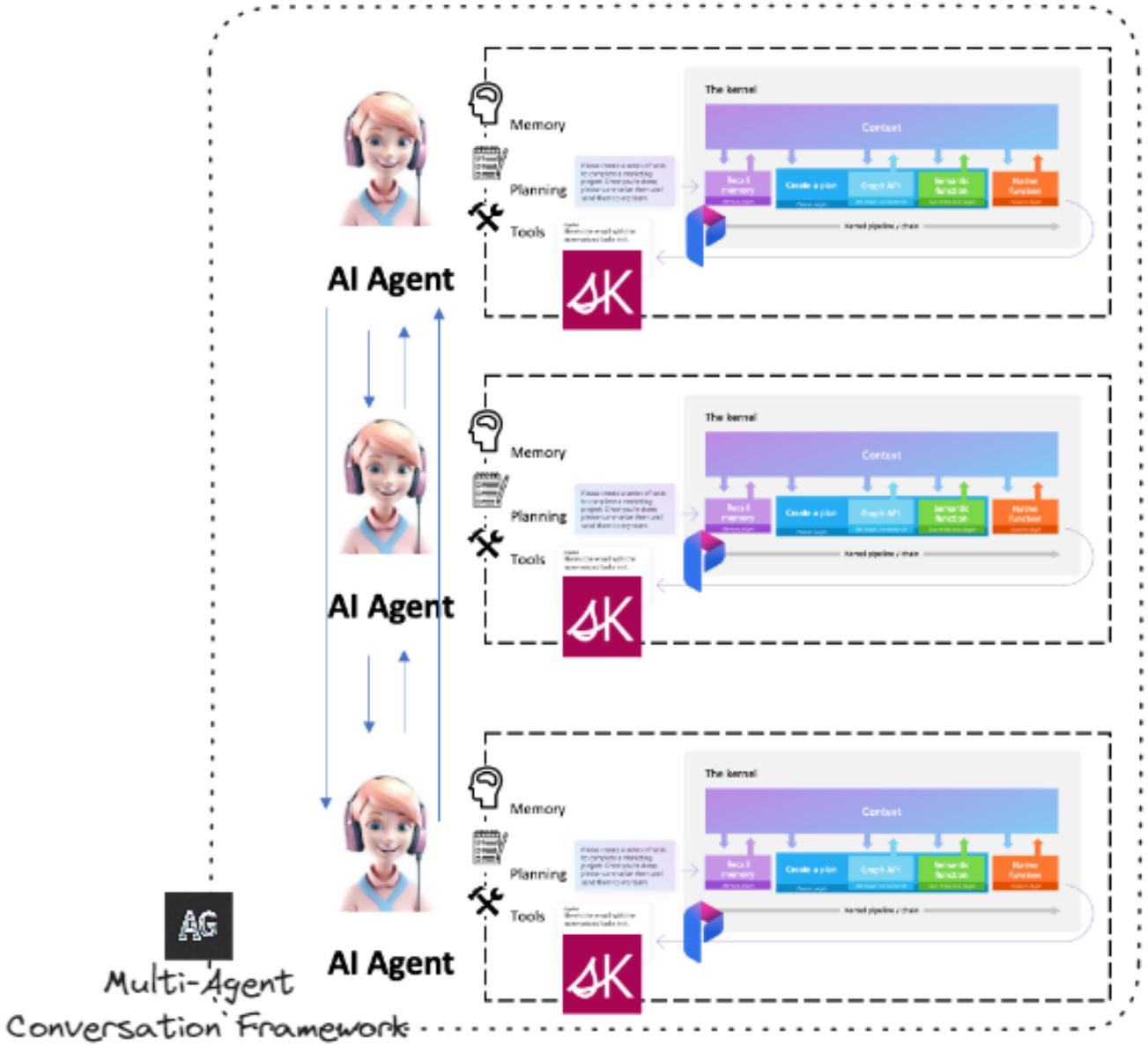
Types Of Agents

There's a wide spectrum of agents that can be built, ranging from simple chat bots to fully automated AI assistants



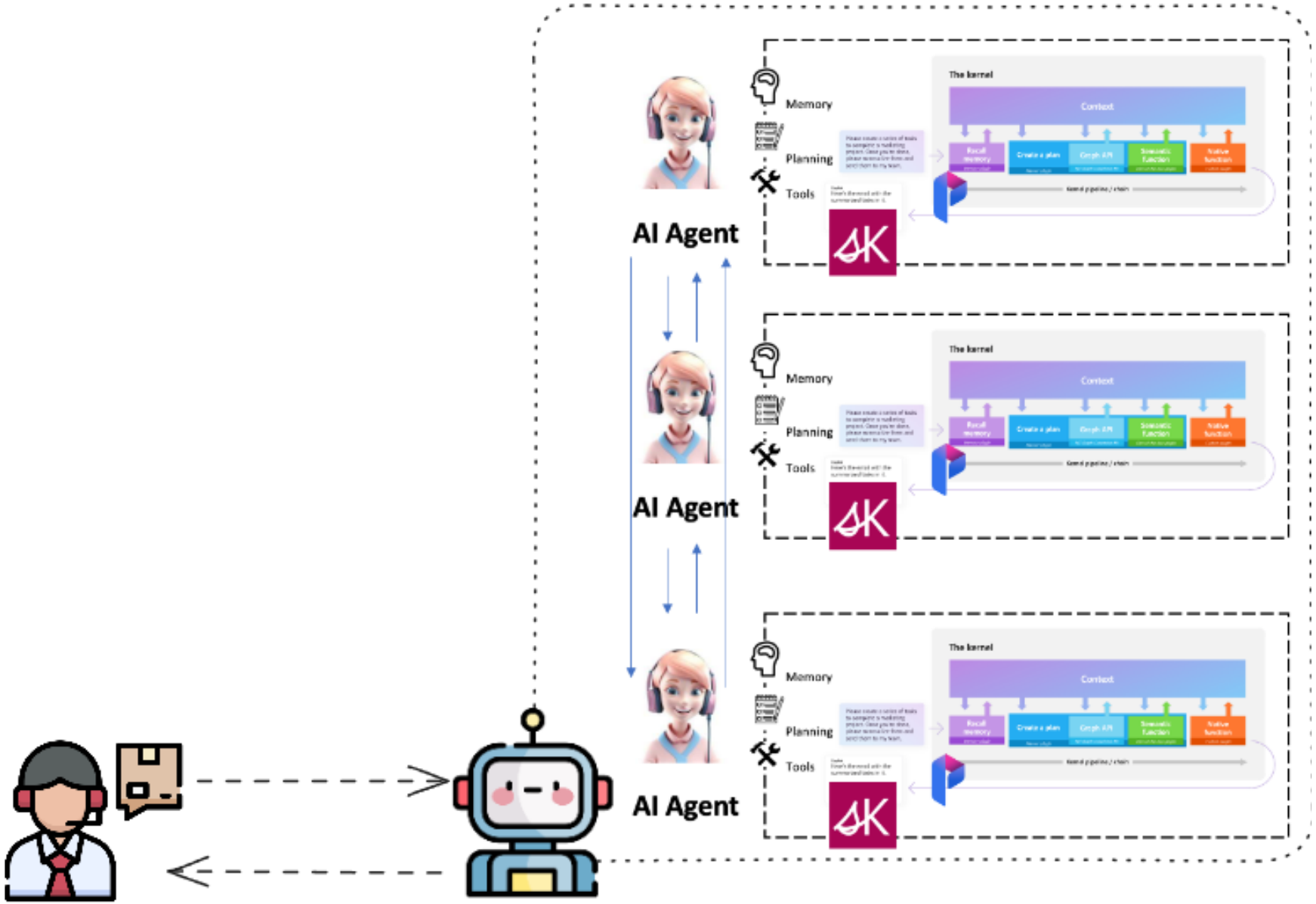
Single AI Agent

Work completed in specific task scenarios, such as the agent workspace under GitHub Copilot Chat, is an example of completing specific programming tasks based on user needs.



Multi-AI agents

The work of mutual interaction between AI agents. Multi-agent application scenarios are very helpful in highly collaborative work, such as software industry development, intelligent production, enterprise management, etc.



Hybrid AI Agent

This is human-computer interaction, making decisions in the same environment. For example, smart medical care, smart cities and other professional fields can use hybrid intelligence to complete complex professional work.

Developer tools

Microsoft Autogen

BOT SHARP

LLaMa Sharp

LMStudio

Semantic Kernel

SciSharp STACK

A .NET based Open Source Ecosystem for Data Science, Machine Learning and AI

SciSharp brings all major ML/AI Frameworks from Python to .NET

Recap: Navigating Generative AI - A Developer's Guide

Transformers and Large Language Models

Discussion on the efficiency of transformer models for training and the capabilities of Large Language Models (LLMs) with trillions of parameters, enabling human-like responses in Natural Language Processing (NLP).

Semantic Kernel SDK

Exploration of Semantic Kernel, an SDK that manages prompts for AI services using LLMs, specifically for C#.

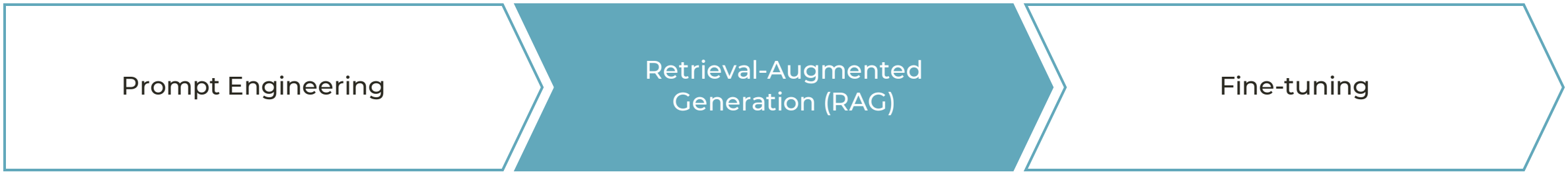
Types Of Agents

Lifelike chatbots for engaging interactions, targeted AI assistants for specific workflows, data-driven insights and decision-making, creative content generation, self-directed AI with learning

Building an Effective Generative AI System

Tailoring the prompts to guide the model's responses, ensuring optimal performance. Version and evaluate prompts for continuous improvement.

Modify the base LLM model by fine-tuning it on task-specific data to better suit your specialized use cases and requirements.



Enhance context understanding by providing external data beyond the LLM's training corpus, such as proprietary company data or industry-specific information.

The Advantages of RAG Systems in Generative AI

Combining Retrieval and Generation

merge retrieval and generative techniques, enabling efficient information search and coherent text generation.

Enhancing Accuracy and Relevance

retrieve relevant documents, ensuring accurate and contextual responses.

Enabling Scalability

handling large datasets efficiently with relevant information.

Improving Contextual Understanding

retrieval enhances understanding and response quality.

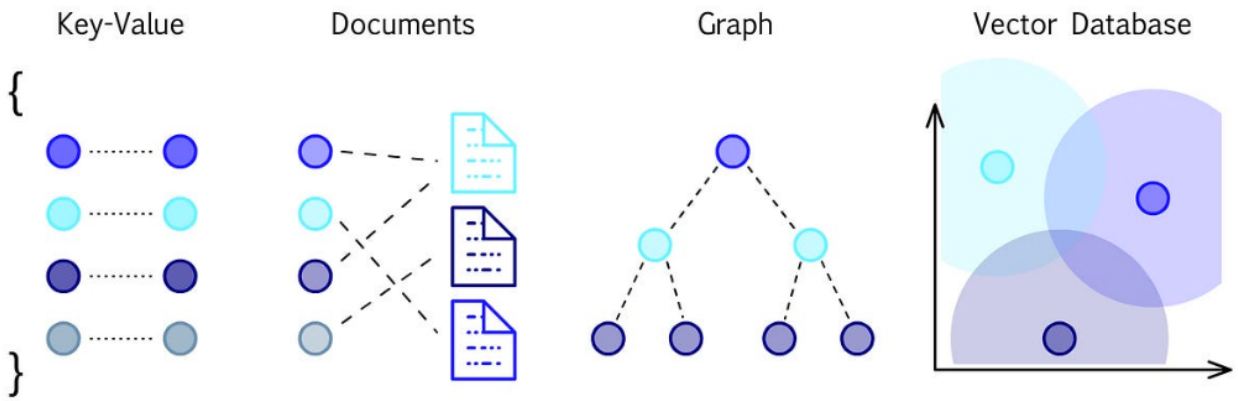
Mitigating Hallucination

retrieving documents grounds generation, reducing misinformation.

Versatile Applications

improve performance in QA, content creation, and virtual assistants.

The Rise of Vector Databases in AI

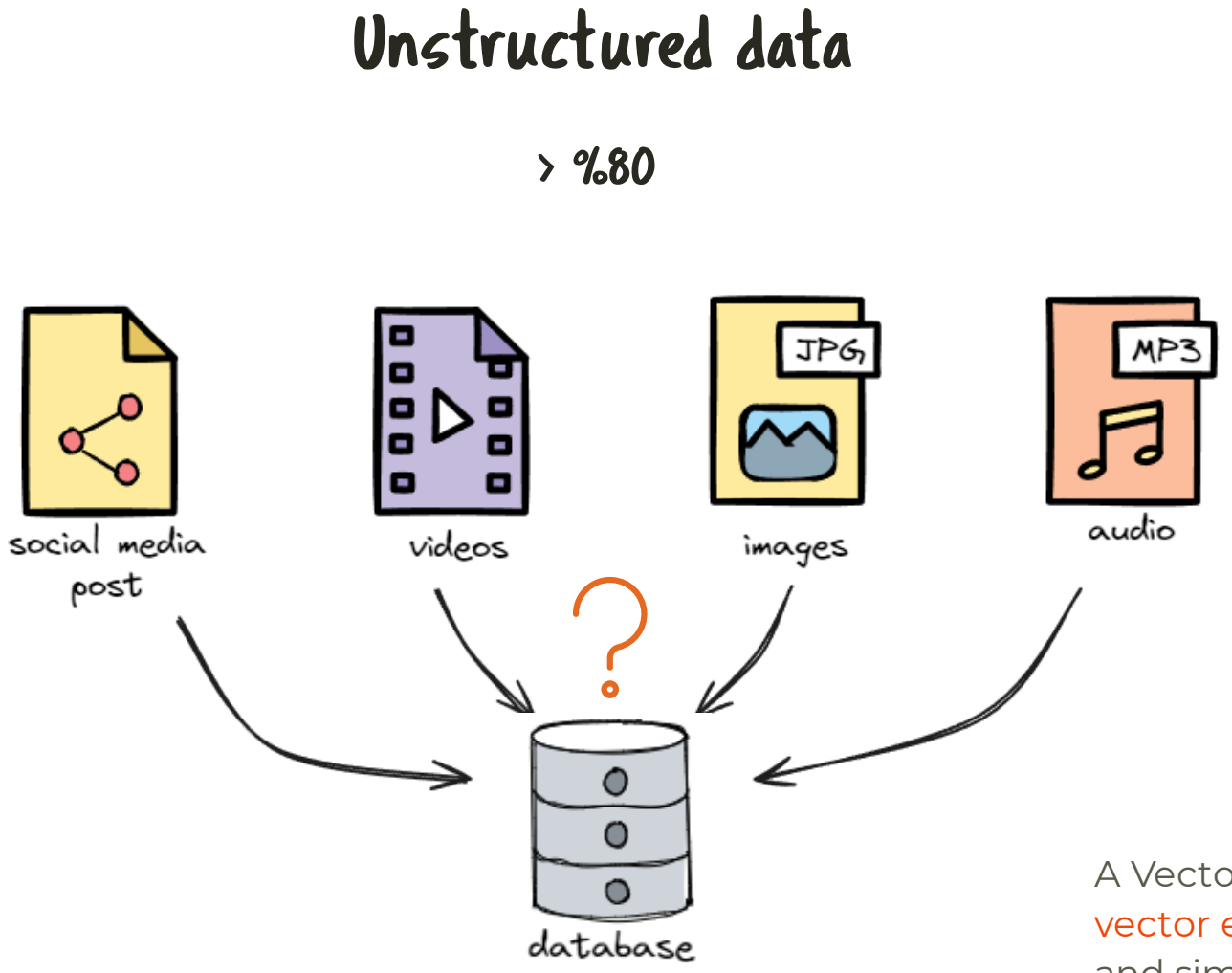


A vector database **indexes** and stores **vector embeddings** for **fast retrieval** and **similarity search.**

Vectors in programming are straightforward: an array of numbers representing both size and direction. Easily defined by coding a numerical array.

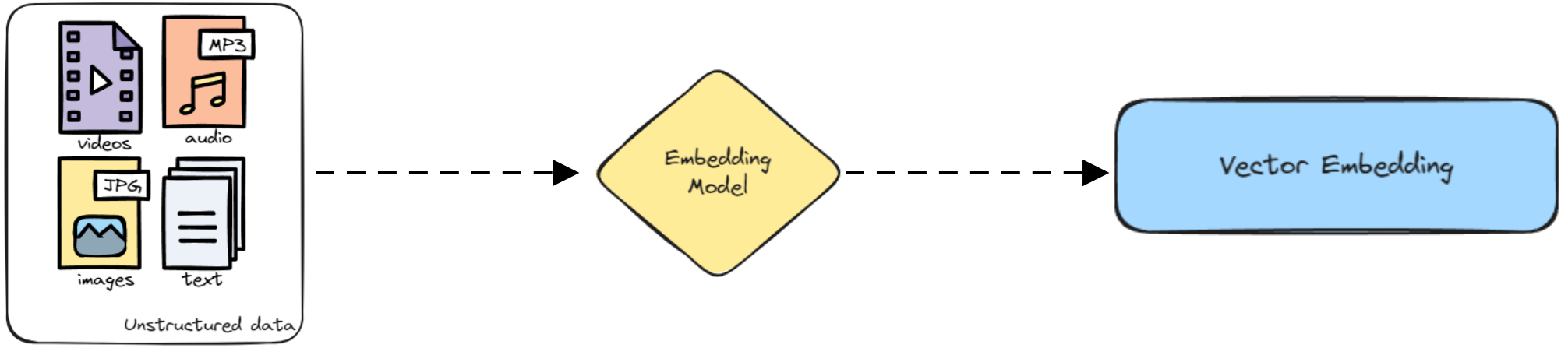
The Vector database is a new kind of database for the AI era

Why do we need vector database



A Vector database indexes and stores vector embeddings for fast retrieval and similarity search.

How do you generate or create embeddings?



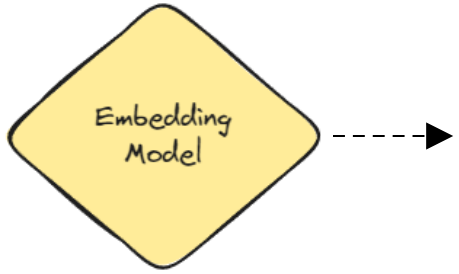
$$[0.1, 0.2, 0.3, 0.7, \ldots]$$

The process of generating embeddings is that you need an embedding model from a paid source like OpenAI's "text-embedding-ada-002" model or an open source from HuggingFace's "SentenceTransformers"

JavaScript
C#
GoLang

Fenerbahçe
Champion
Türkiye

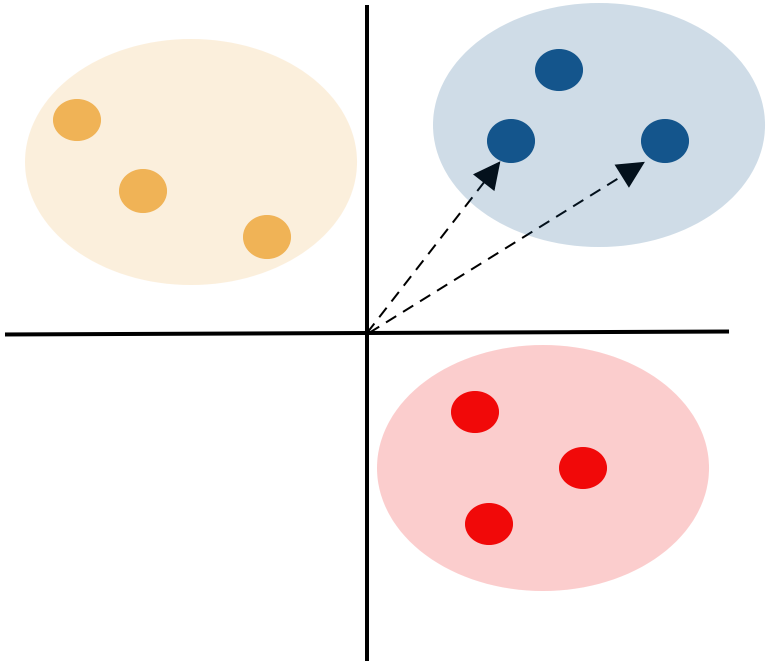
Laptop
Desktop
Widget



[2.5 , -2]
[2.5 , -3]
[4 , -1]

[3.4 , 5]
[2.5 , 6]
[4 , 1]

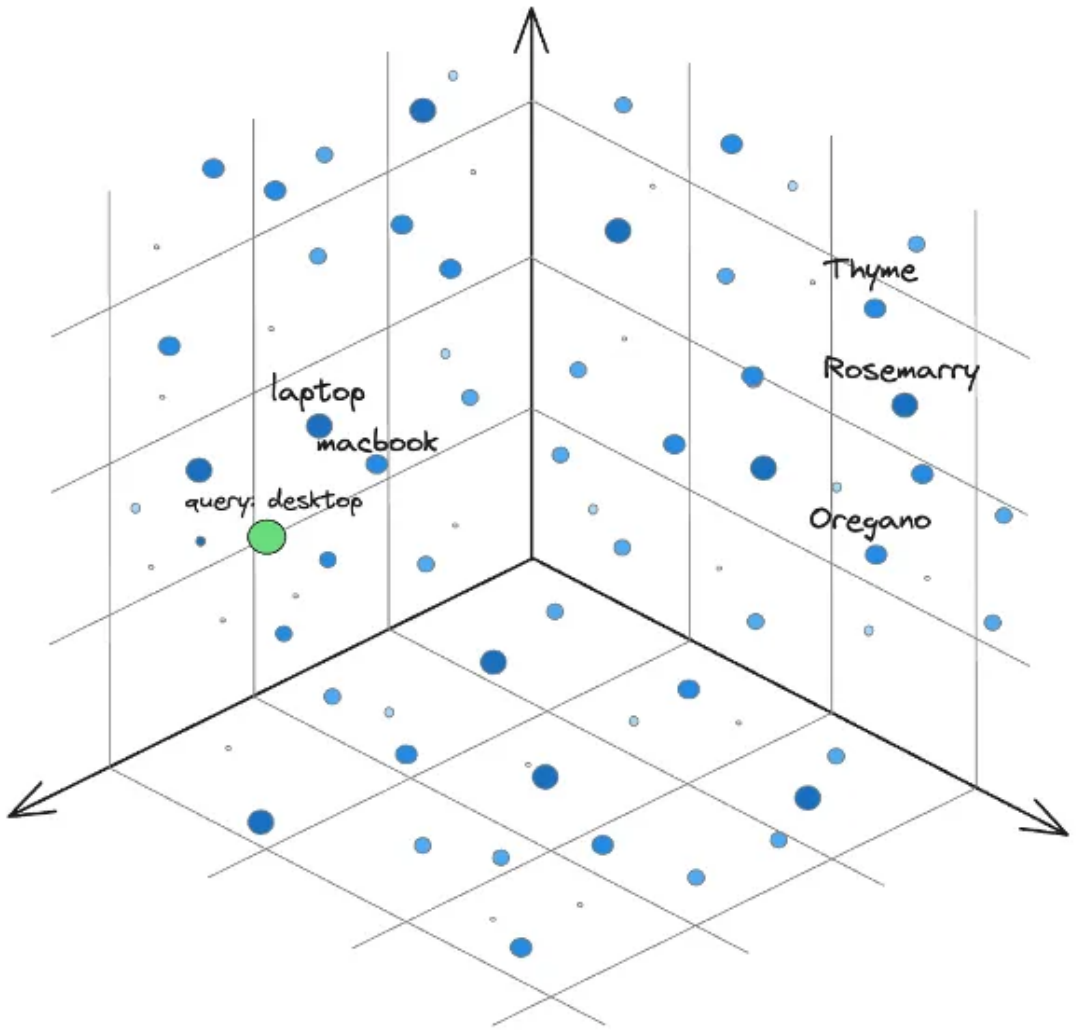
[-3.4 , 5]
[-2.5 , 6]
[-4 , 1]



$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

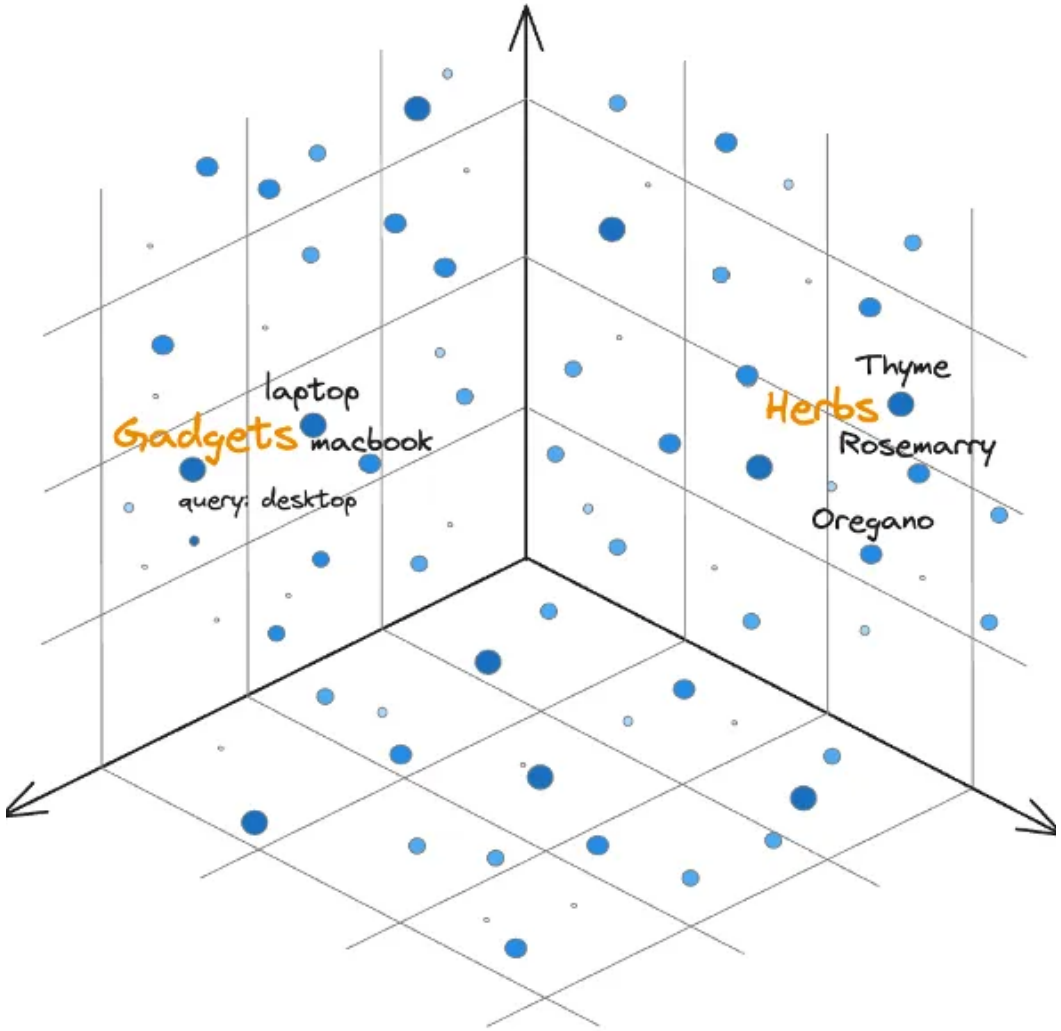
Vector embeddings (2D example)

Clusters of embeddings based on their similarity



The embeddings are grouped depending on how closely the words are related. Your query, for example, a desktop, might bring a Macbook and laptop because they are all personal computers or gadgets

Clusters of embeddings based on their similarity

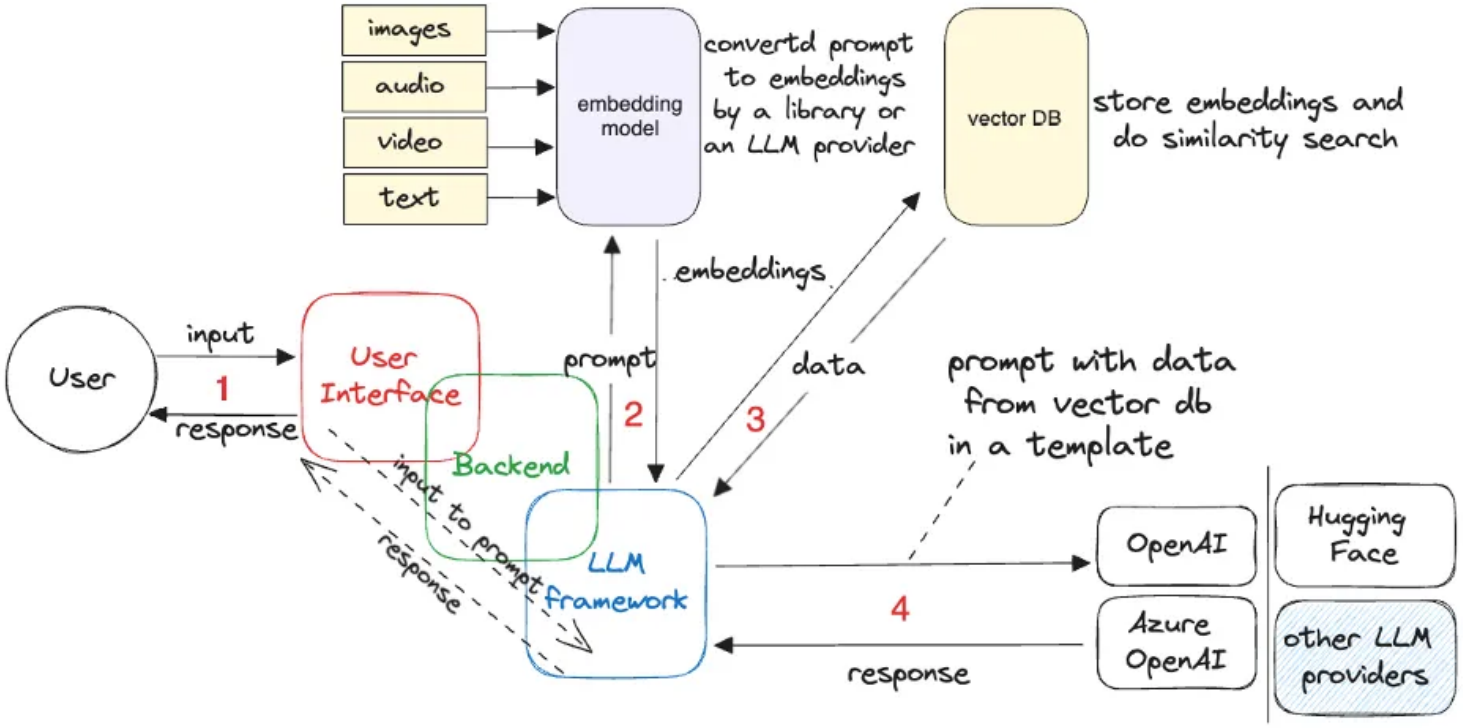


The thyme, rosemary, and oregano are located in the herbs area. And just like in GPS, the embeddings are like latitude and longitude coordinates.

The solutions to the problems of LLMs

Retrieval-Augmented Generation

From the user, convert your prompt into embeddings for similarity search in the vector db. Then, arrange the original prompt plus the vector DB's results into the prompt template before sending it to the LLM provider.

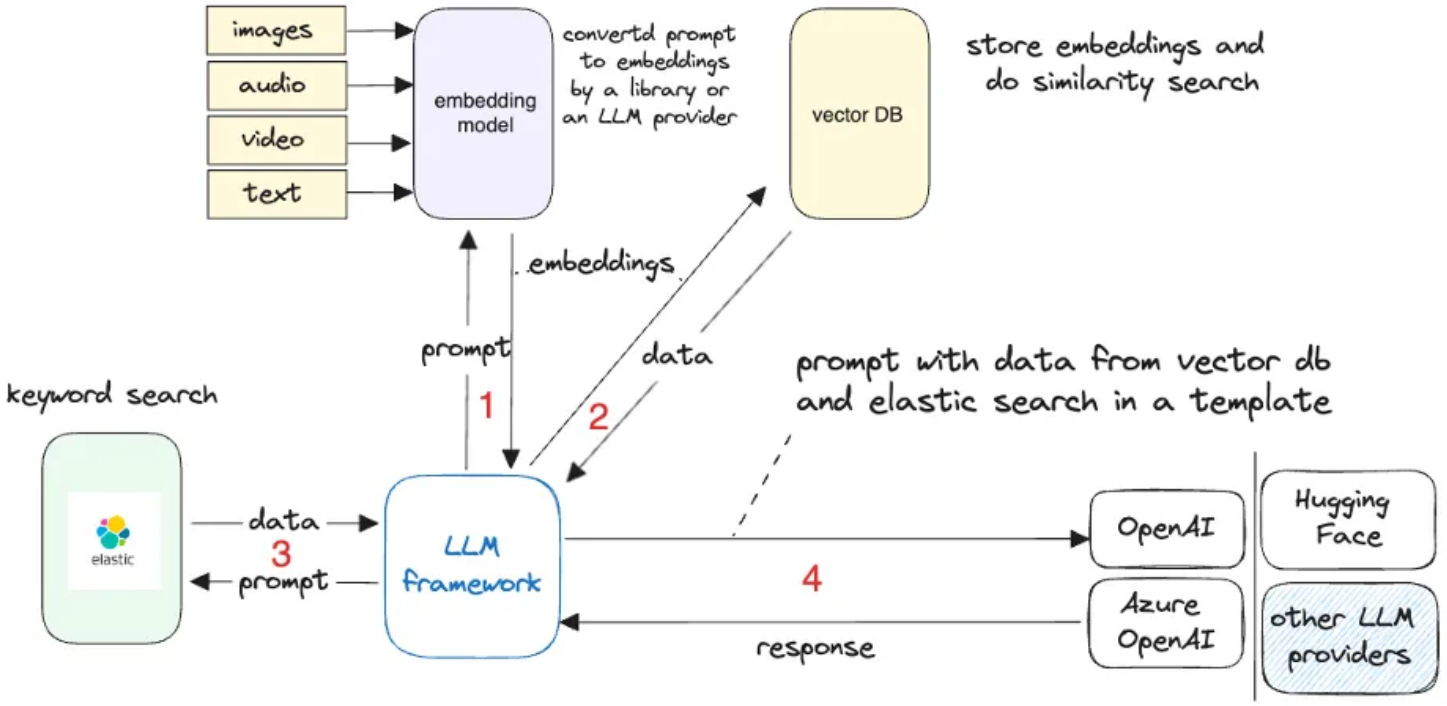


RAG is a design pattern for augmenting a model's capabilities by combining it with a retrieval component.

The solutions to the problems of LLMs

Hybrid RAG

Which uses a keyword search as a supplement for improving results. We combine the vector and keyword search results before sending them to the LLMs.

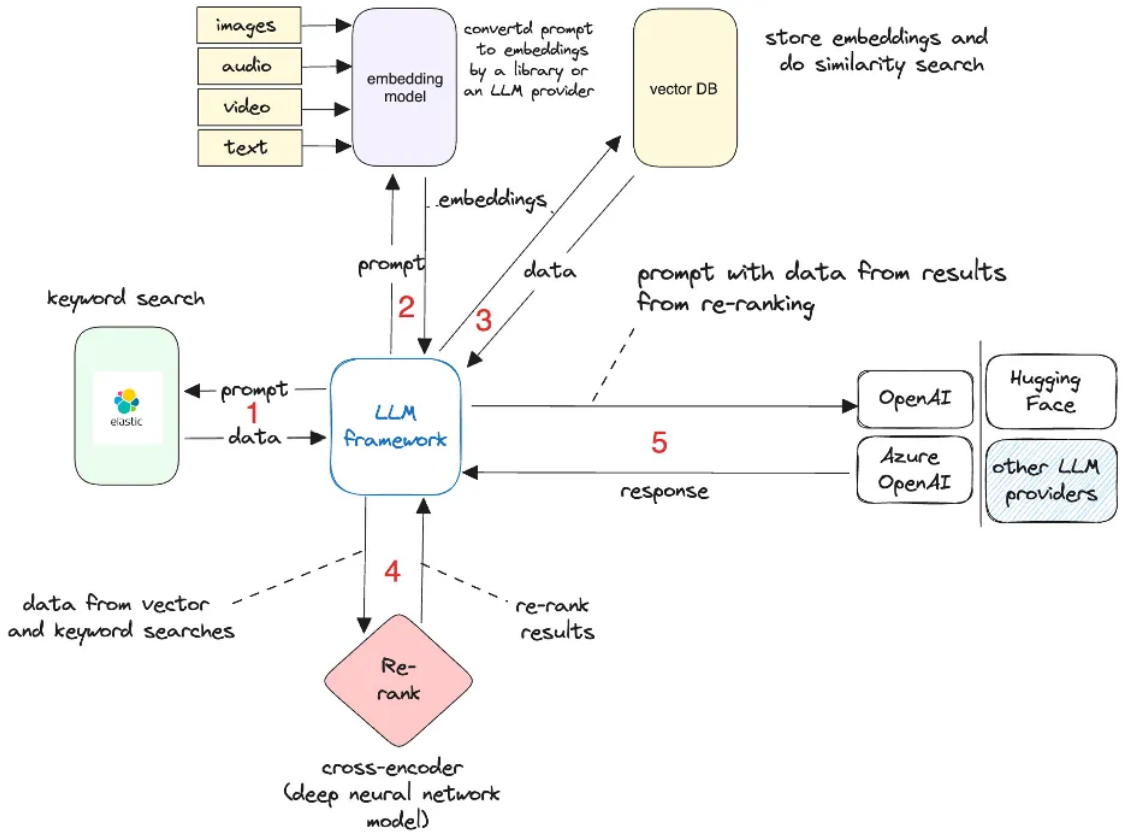


RAG is a design pattern for augmenting a model's capabilities by combining it with a retrieval component.

The solutions to the problems of LLMs

Hybrid RAG + Re-ranking

The goal of re-ranking is to improve the relevance of the results returned by an initial retrieval query.



RAG is a design pattern for augmenting a model's capabilities by combining it with a retrieval component.

/alperhankendi @alper_hankendi /alperhankendi