Contents

[OPEN AI 2](#_Toc205139219)

[GPT3 VERSUS GPT4 2](#_Toc205139220)

[INTERACT WITH OPEN AI 2](#_Toc205139221)

[OPEN AI – MODELS 2](#_Toc205139222)

[USING OPENAI MODELS 3](#_Toc205139223)

[SIMPLE API CALL TO OPEN AI 3](#_Toc205139224)

[CREATING EMBEDDINGS 3](#_Toc205139225)

[IMAGE GENERATION USING DALL-E 4](#_Toc205139226)

[SPEECH TO TEXT CONVERSION 5](#_Toc205139227)

[AZURE OPEN AI & AZURE FOUNDRY 5](#_Toc205139228)

[AZURE OPEN AI 6](#_Toc205139229)

[SETTING UP OPEN SERVICE 6](#_Toc205139230)

# OPEN AI

## GPT3 VERSUS GPT4

|  |  |
| --- | --- |
|  |  |

# INTERACT WITH OPEN AI

## OPEN AI – MODELS

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Type | Capabilities | Use Cases |
| GPT-4o | Multimodal LLM | Text, image, audio input/output; real-time reasoning | Chatbots, assistants, multimodal search |
| o3-series | Text-only LLM | Advanced reasoning, long-context understanding | Research, technical writing, complex workflows |
| o1-series | Lightweight LLM | Fast, efficient responses with decent reasoning | Mobile apps, real-time systems |
| Whisper | Audio model | Speech-to-text, multilingual transcription | Voice assistants, accessibility tools |
| DALL·E | Image model | Text-to-image generation and editing | Design, prototyping, creative content |
| TTS (Text-to-Speech) | Audio model | Converts text to natural-sounding speech | Voice synthesis, accessibility |
| Embedding Models | Vector model | Converts text into high-dimensional vectors for semantic similarity | Search, clustering, recommendation systems |

## USING OPENAI MODELS

### SIMPLE API CALL TO OPEN AI

|  |  |
| --- | --- |
| **INSTALL OPEN AI MODULE** | pip install openai |
| **from openai import OpenAI**  **from dotenv import load\_dotenv**  **import os**  **load\_dotenv('openai.env')**  **# Access the environment variables from the .env file**  **api\_key = os.environ.get('OPENAI\_API\_KEY')**  **from openai import OpenAI**  **client = OpenAI()**  **response = client.chat.completions.create(**  **model="gpt-4",**  **messages=[**  **{"role": "user", "content": "Who is Prime Minister of India and give some bullet points of his achievements"},**  **]**  **)**  **print(response)** | Interaction with OpenAI's GPT-4 model using the official OpenAI Python library.   * [**load\_dotenv('openai.env')**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) **:** Loads environment variables from a file named [openai.env](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), which contains the API keys. * [**os.environ.get('OPENAI\_API\_KEY')**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Retrieves the OpenAI API key from the environment variable * The [OpenAI](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) client is then instantiated, which will be used to send requests to the API. * A chat completion request is made using the **[client.chat.completions.create()](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")** method. The request specifies the model (gpt-4) and provides a list of messages, simulating a conversation * The model processes this input and generate a response. |

### CREATING EMBEDDINGS

|  |  |
| --- | --- |
| **from openai import OpenAI**  **from dotenv import load\_dotenv**  **import os**  **load\_dotenv('openai.env')**  **# Access the environment variables from the .env file**  **api\_key = os.environ.get('OPENAI\_API\_KEY')**  **from openai import OpenAI**  **client = OpenAI()**  **response = client.embeddings.create(**  **input="cat",**  **model="text-embedding-3-small"**  **)**  **print(response.data[0].embedding)** | * Embeddings are a way to represent data—especially text, images, or other complex inputs—as numerical vectors in a high-dimensional space. These vectors capture the **semantic meaning** or **contextual relationships** of the data, making it easier for machines to process and understand. * In Simple terms - Imagine we want to teach a computer what words mean. Instead of giving it dictionary definitions, we give each word a unique set of numbers (a vector) that reflects how it's used in language. * Words with similar meanings or contexts will have similar vectors |

|  |  |
| --- | --- |
| What do we mean by vectors in a high-dimensional space,  It means it’s a representation of data (like words, images, or sentences) as points in a space with many dimensions—often **hundreds or even thousands**.  What is a Vector?  A **vector** is just a list of numbers. For example:   * A 2D vector: [3.5, -1.2] → can be plotted on a flat plane. * A 3D vector: [2.1, 0.5, -3.3] → can be visualized in 3D space.   But embeddings often use **300, 768, or even 4096 dimensions**—far beyond what we can visualize.  Why High-Dimensional?  Because more dimensions allow us to **capture more nuanced relationships**. For example:   * In a 300-dimensional space, each dimension might encode something like:   + Gender association   + Verb tense   + Sentiment   + Topic relevance   + Contextual usage   So the word **“apple”** might be close to:   * “fruit” in one context * “iPhone” in another   Depending on how the embedding is trained. | |
| Imagine each word is a dot in a huge invisible cloud. Words that **mean similar things** or are **used in similar contexts** are **closer together** in this cloud. For example - Imagine a 2D scatter plot where each **dot** represents a word. Words with similar meanings or contexts would appear **close together**.  For example:   * **Cluster 1 (Royalty)**: king, queen, prince, princess * **Cluster 2 (Fruits)**: apple, banana, orange, grape * **Cluster 3 (Vehicles)**: car, bus, train, truck * **Cluster 4 (Animals)**: cat, dog, lion, tiger   *Each cluster forms because the embedding vectors for those words are similar in high-dimensional space, and dimensionality reduction (like t-SNE or PCA) helps us visualize that in 2D.* |  |

### IMAGE GENERATION USING DALL-E

|  |
| --- |
| **from openai import OpenAI**  **from dotenv import load\_dotenv**  **import os**  **load\_dotenv('openai.env')**  **# Access the environment variables from the .env file**  **api\_key = os.environ.get('OPENAI\_API\_KEY')**  **from openai import OpenAI**  **client = OpenAI()**  **response = client.images.generate(**  **model="dall-e-3",**  **prompt="a Black furry Dog with black eyes with a dog collar and a white cat with blue eyes with a necklace",**  **size="1024x1024",**  **quality="standard",**  **n=1,**  **)**  **image\_url = response.data[0].url**  **print(image\_url)** |

### SPEECH TO TEXT CONVERSION

|  |
| --- |
| **from openai import OpenAI**  **from dotenv import load\_dotenv**  **import os**  **load\_dotenv('openai.env')**  **# Access the environment variables from the .env file**  **api\_key = os.environ.get('OPENAI\_API\_KEY')**  **client = OpenAI()**  **audio\_file= open("/Users/kshitijjoy\_1/Downloads/deep\_fake\_video.mp4", "rb")**  **transcription = client.audio.transcriptions.create(**  **model="whisper-1",**  **file=audio\_file**  **)**  **print(transcription.text)** |

# AZURE OPEN AI & AZURE FOUNDRY

* Azure OpenAI Service is part of the Azure AI Foundry ecosystem, but Foundry offers more tools and flexibility for building full-fledged AI applications.

|  |  |
| --- | --- |
| Platform | Scope |
| Azure AI Foundry | * **Azure AI Foundry** is a **comprehensive platform** designed to help you build **generative AI applications**—like chatbots, copilots, and intelligent agents. * It’s like a **workshop** where we get all the tools, models, and infrastructure needed to create smart, interactive AI systems. |
| Azure OpenAI Service | * **Azure OpenAI Service** is a **specialized service** within Azure that gives you **direct access to OpenAI’s models** (*like GPT-4, GPT-4o, DALL·E, Whisper*). * We use it when we want to **generate text, images, transcribe audio**, or perform semantic search using these models. |

HOW THE ARE RELATED

* **Azure OpenAI Service** is a **toolbox** with powerful tools (models).
* **Azure AI Foundry** is the **entire workshop** that includes:
  + That toolbox (OpenAI models)
  + Other toolboxes (Meta, Hugging Face models)
  + Workbenches (SDKs, orchestration tools)
  + Safety gear (governance, monitoring)
  + Collaboration zones (project workspaces)
* **Azure OpenAI Service is a subset of Azure AI Foundry**—we can use it **inside Foundry**, but Foundry gives us **much more**.

# AZURE OPEN AI

* Microsoft’s unified platform-as-a-service designed to simplify and accelerate the development of **generative AI applications** and **AI agents**—especially for enterprise use cases
* **Azure OpenAI Service** is a cloud-based offering from Microsoft that provides access to OpenAI’s powerful language models (like GPT-4, GPT-4o, DALL·E, Whisper, etc.) through the **Azure platform**

## SETTING UP OPEN SERVICE

1. Step 1: Create Azure Open AI Service from Azure Portal