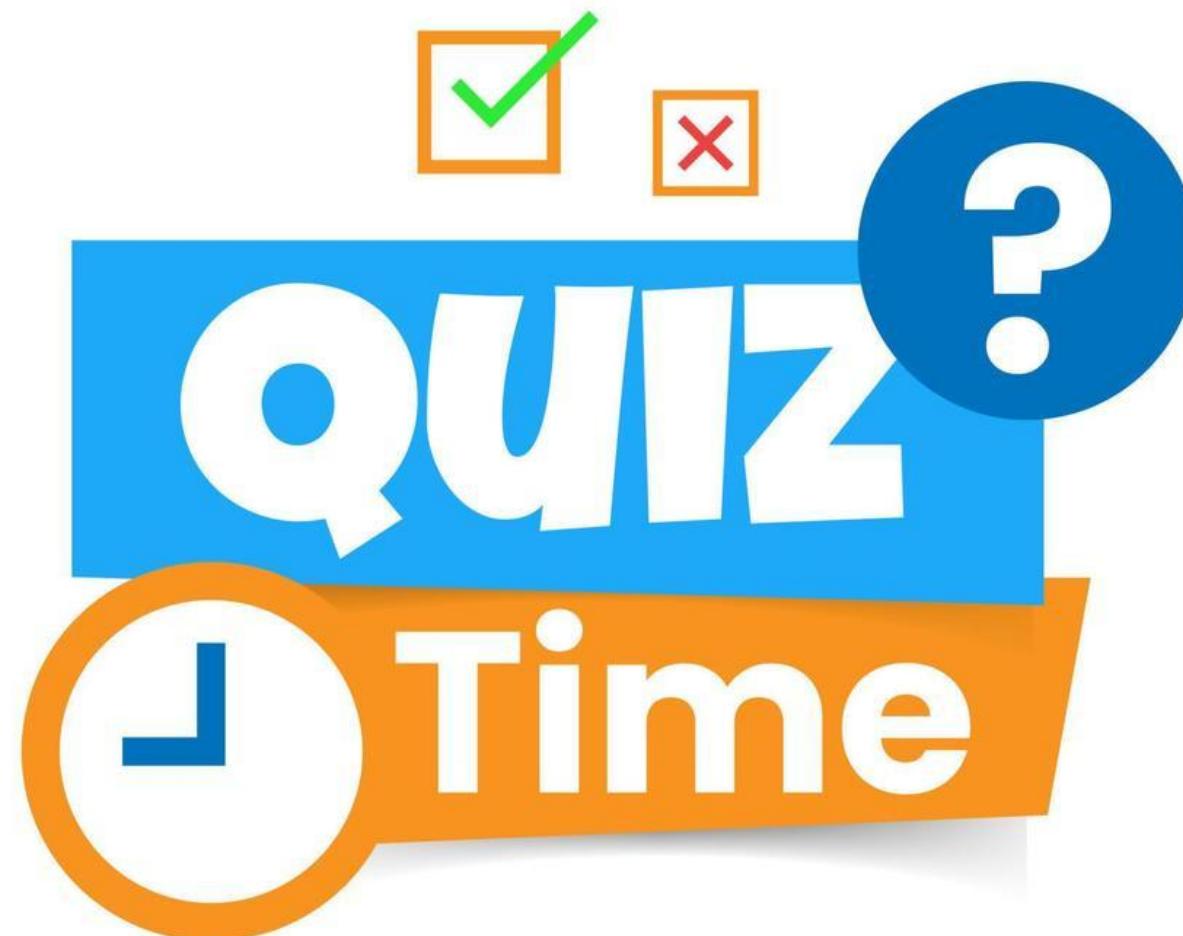




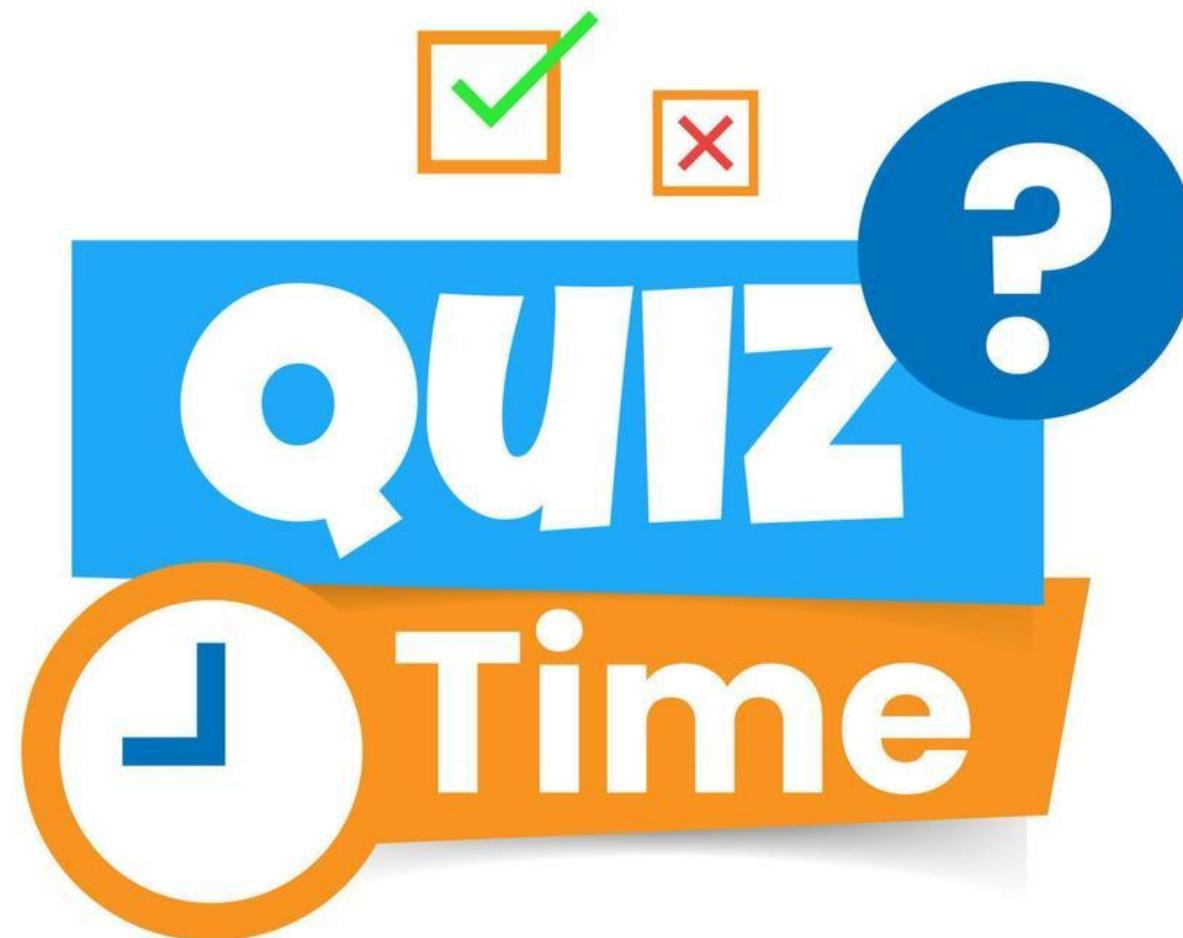
Part II - Prompt Engineering





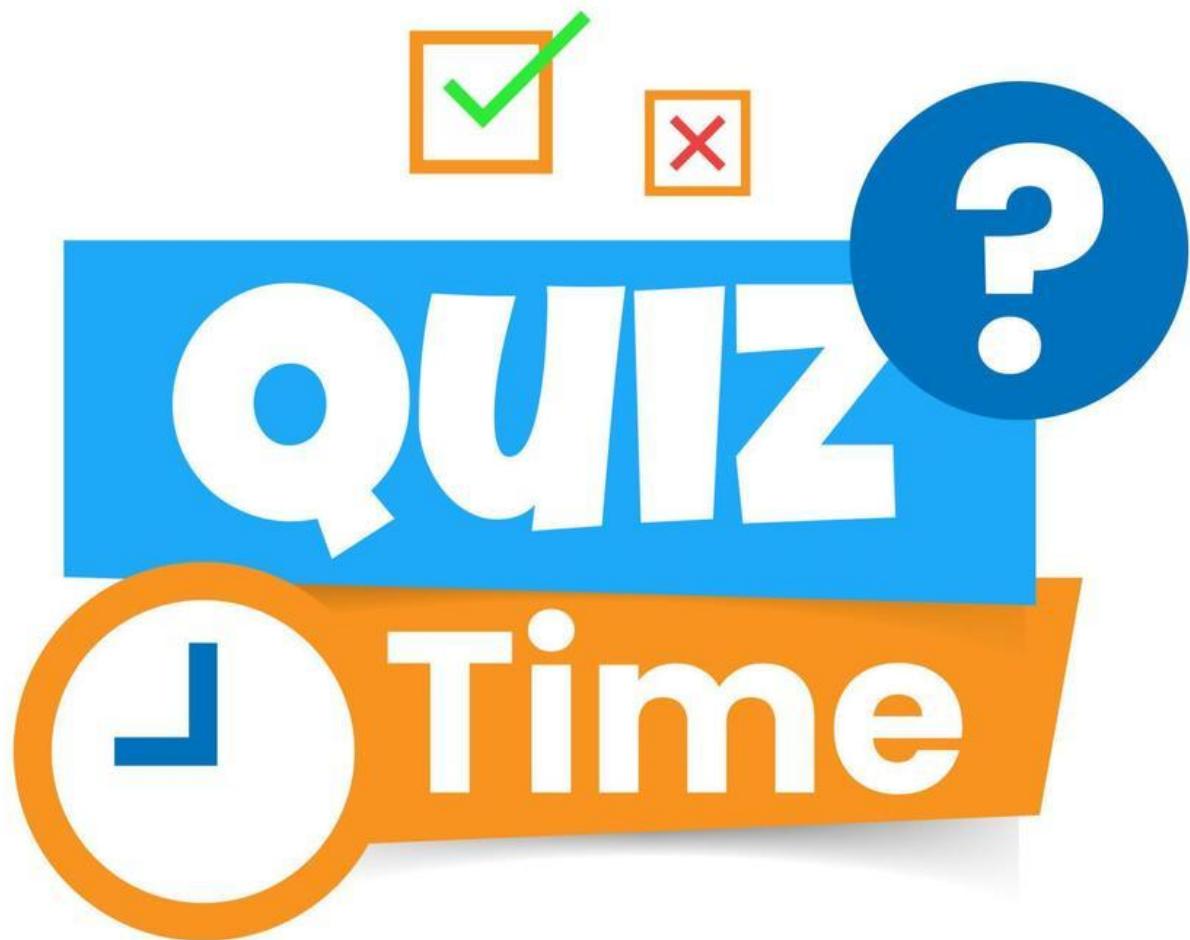
InnovateTech Solutions is developing an AI-driven platform to revolutionize content creation for digital marketing campaigns. The company aims to generate creative and personalized content to enhance user engagement. As the Chief Data Scientist, you are leading the team in selecting the most suitable ML model for this task. Which of the following key ML models is pivotal for InnovateTech Solutions for generating new and personalized content for digital marketing campaigns?

| | |
|---|-------------------|
| A | Descriptive Model |
| B | Generative Model |



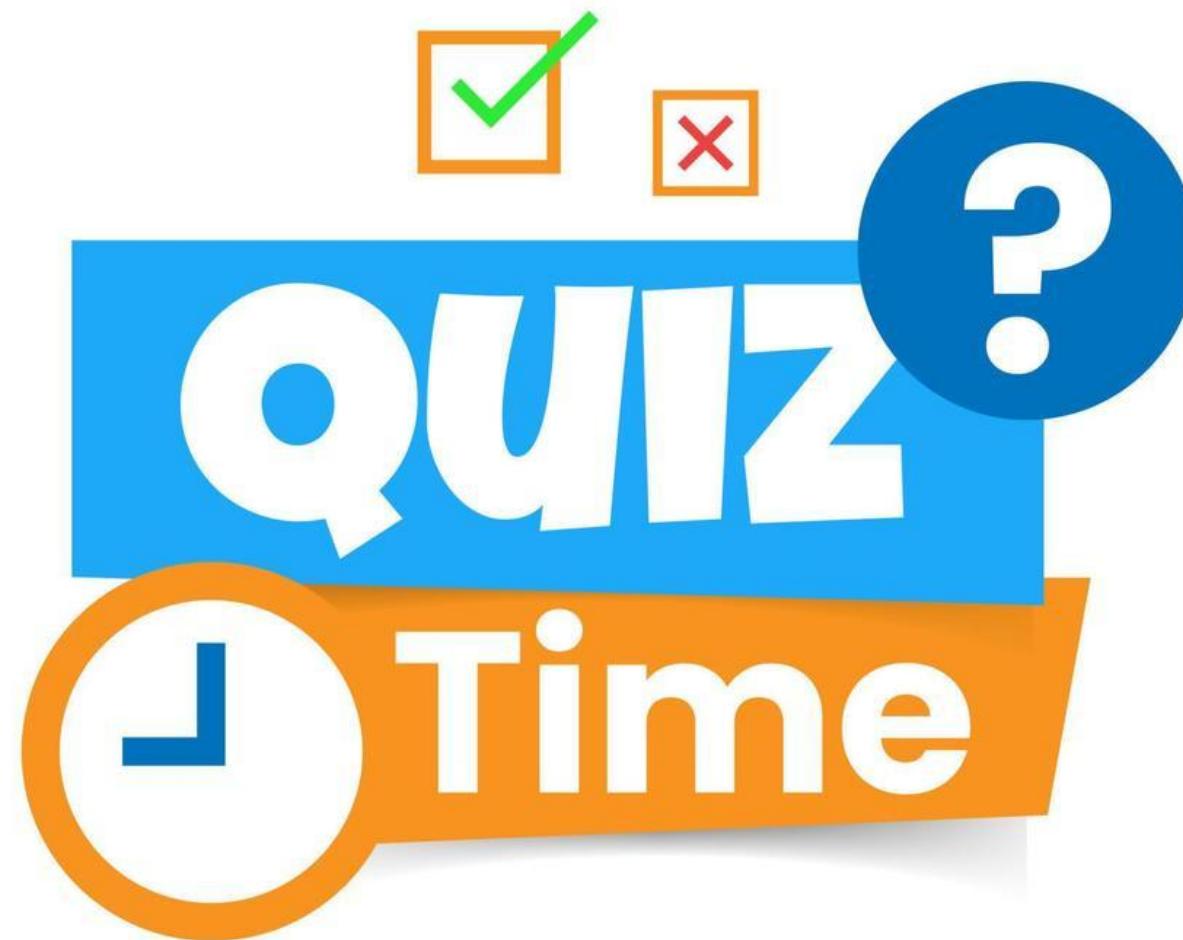
InnovateTech Solutions is developing an AI-driven platform to revolutionize content creation for digital marketing campaigns. The company aims to generate creative and personalized content to enhance user engagement. As the Chief Data Scientist, you are leading the team in selecting the most suitable ML model for this task. Which of the following key ML models is pivotal for InnovateTech Solutions for generating new and personalized content for digital marketing campaigns?

| | |
|---|-------------------|
| A | Descriptive Model |
| B | Generative Model |



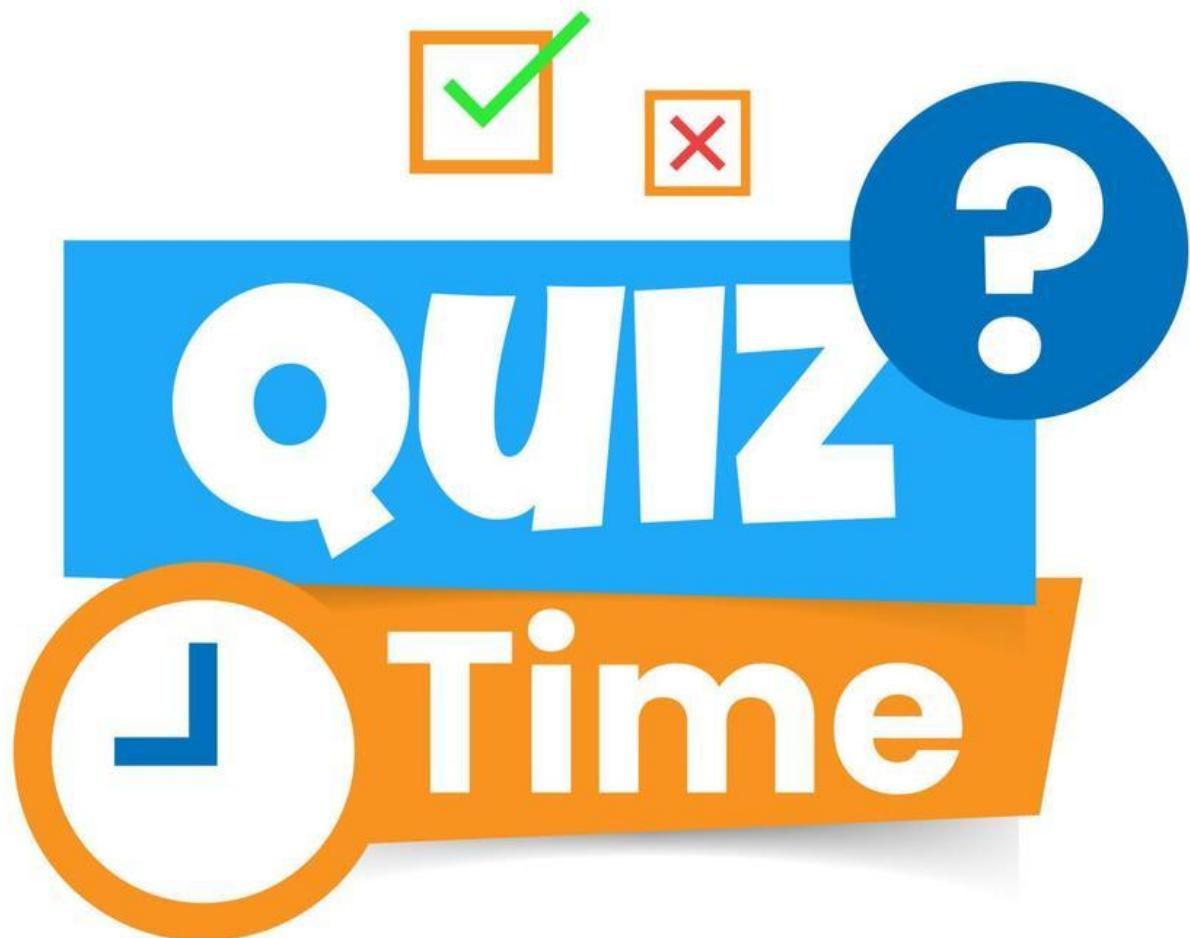
As the Chief Technology Officer (CTO) of TechSolutions Ltd., a leading IT organization specializing in digital transformation, you are tasked with enhancing the user experience and operational efficiency of your company's internal knowledge base. Considering the utilization of Azure Cognitive Services, which of the following use cases would be most impactful for the organization using Azure's AI capabilities?

| | |
|---|---|
| A | Implementing Azure Computer Vision to automatically tag and categorize internal scanned images of the document based on their content, facilitating streamlined document management and retrieval |
| B | Deploying Azure Text Analytics on technical manuals to automatically extract key insights and terminologies, aiding in the creation of a comprehensive knowledge base for IT support |
| C | Utilizing Azure Personalizer to customize the user interface of the organization's internal applications based on individual employee preferences and usage patterns |
| D | Implementing Azure QnA Maker to develop an intelligent IT support chatbot, capable of answering employees' queries and providing solutions to common technical issues |



As the Chief Technology Officer (CTO) of TechSolutions Ltd., a leading IT organization specializing in digital transformation, you are tasked with enhancing the user experience and operational efficiency of your company's internal knowledge base. Considering the utilization of Azure Cognitive Services, which of the following use cases would be most impactful for the organization using Azure's AI capabilities?

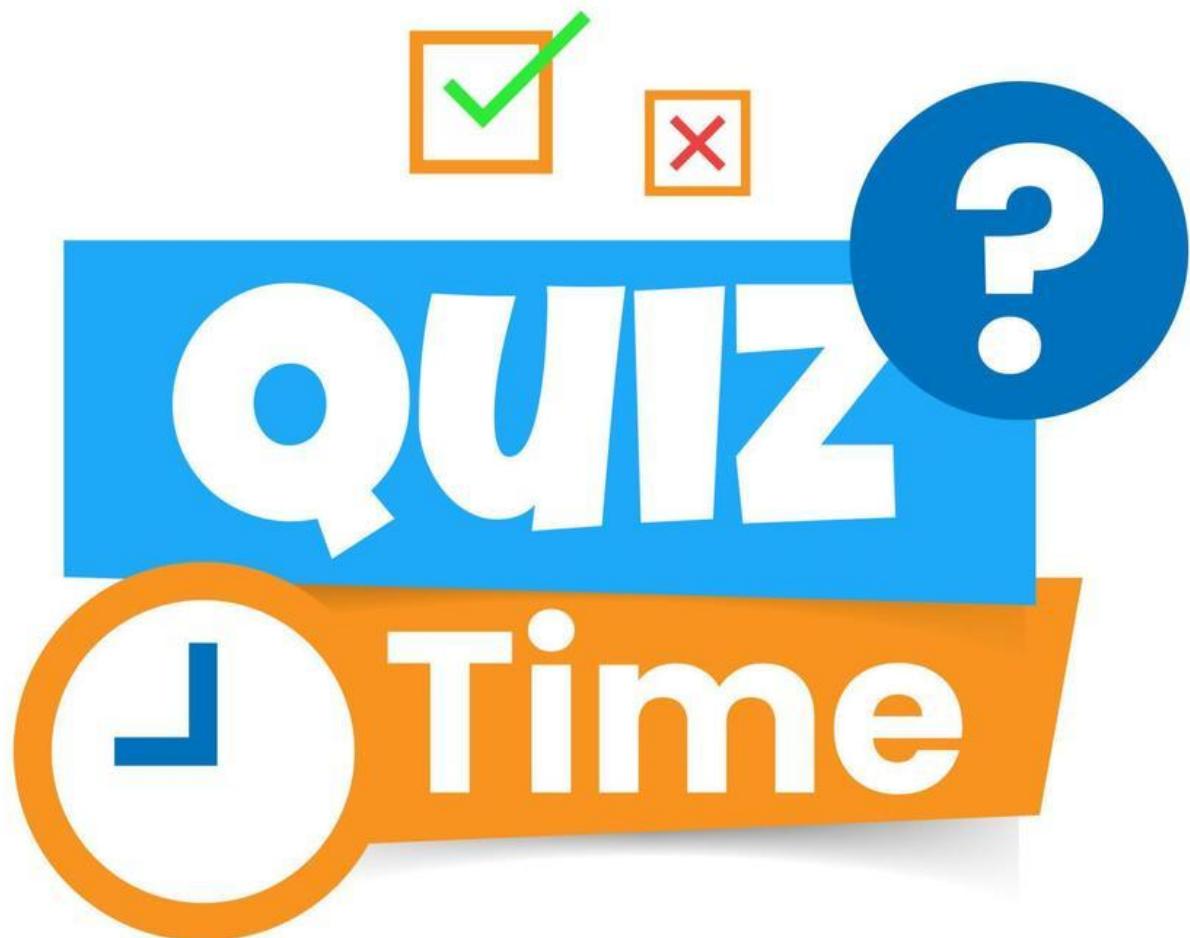
| | |
|---|---|
| A | Implementing Azure Computer Vision to automatically tag and categorize internal scanned images of the document based on their content, facilitating streamlined document management and retrieval |
| B | Deploying Azure Text Analytics on technical manuals to automatically extract key insights and terminologies, aiding in the creation of a comprehensive knowledge base for IT support |
| C | Utilizing Azure Personalizer to customize the user interface of the organization's internal applications based on individual employee preferences and usage patterns |
| D | Implementing Azure QnA Maker to develop an intelligent IT support chatbot, capable of answering employees' queries and providing solutions to common technical issues |



DataSolutions Tech (a leading IT organization focused on data-driven solutions) aims to boost its software development process by leveraging the best features of Generative AI. As the CIO, you want to explore how Generative AI can contribute to the innovation and efficiency of the organization's coding practices.

Which of the following best describes a key benefit or feature of Generative AI for an IT organization in the context of software development?

| | |
|---|--|
| A | Utilizing Generative AI to automate routine IT tasks and administrative processes within the organization |
| B | Integrating Generative AI to analyze and optimize network performance, ensuring seamless connectivity for all employees |
| C | Applying Generative AI for cybersecurity measures, automatically identifying and mitigating potential threats to the IT infrastructure |
| D | Incorporating Generative AI to dynamically generate code snippets and assist developers in rapid prototyping and software development |



DataSolutions Tech (a leading IT organization focused on data-driven solutions) aims to boost its software development process by leveraging the best features of Generative AI. As the CIO, you want to explore how Generative AI can contribute to the innovation and efficiency of the organization's coding practices.

Which of the following best describes a key benefit or feature of Generative AI for an IT organization in the context of software development?

| | |
|---|--|
| A | Utilizing Generative AI to automate routine IT tasks and administrative processes within the organization |
| B | Integrating Generative AI to analyze and optimize network performance, ensuring seamless connectivity for all employees |
| C | Applying Generative AI for cybersecurity measures, automatically identifying and mitigating potential threats to the IT infrastructure |
| D | Incorporating Generative AI to dynamically generate code snippets and assist developers in rapid prototyping and software development |

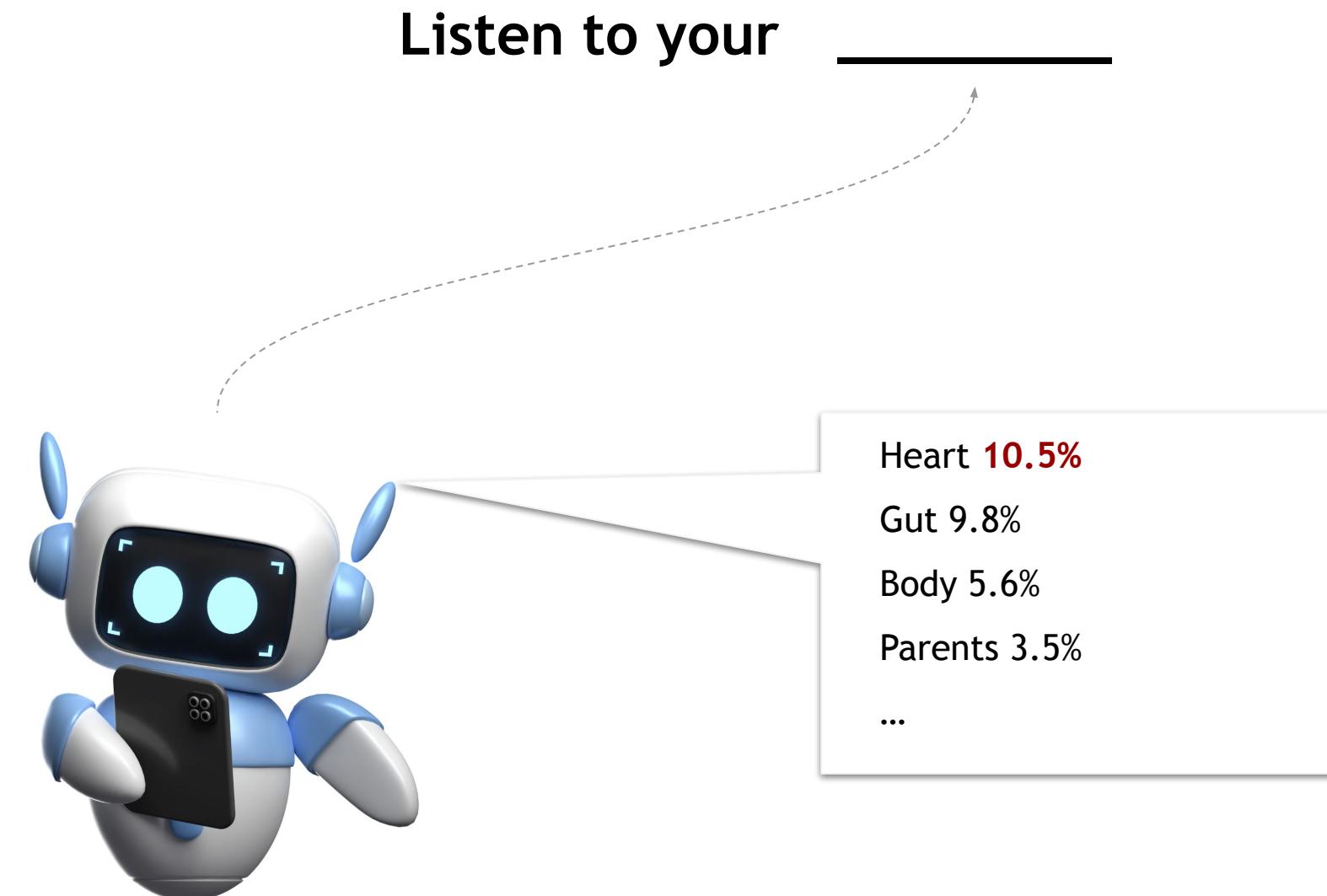
Agenda

- • **Recap of LLMs**
 - Understanding Prompt Design
 - Context, Examples, & Tuning Prompt Parameters
 - Prompt Design Best Practices

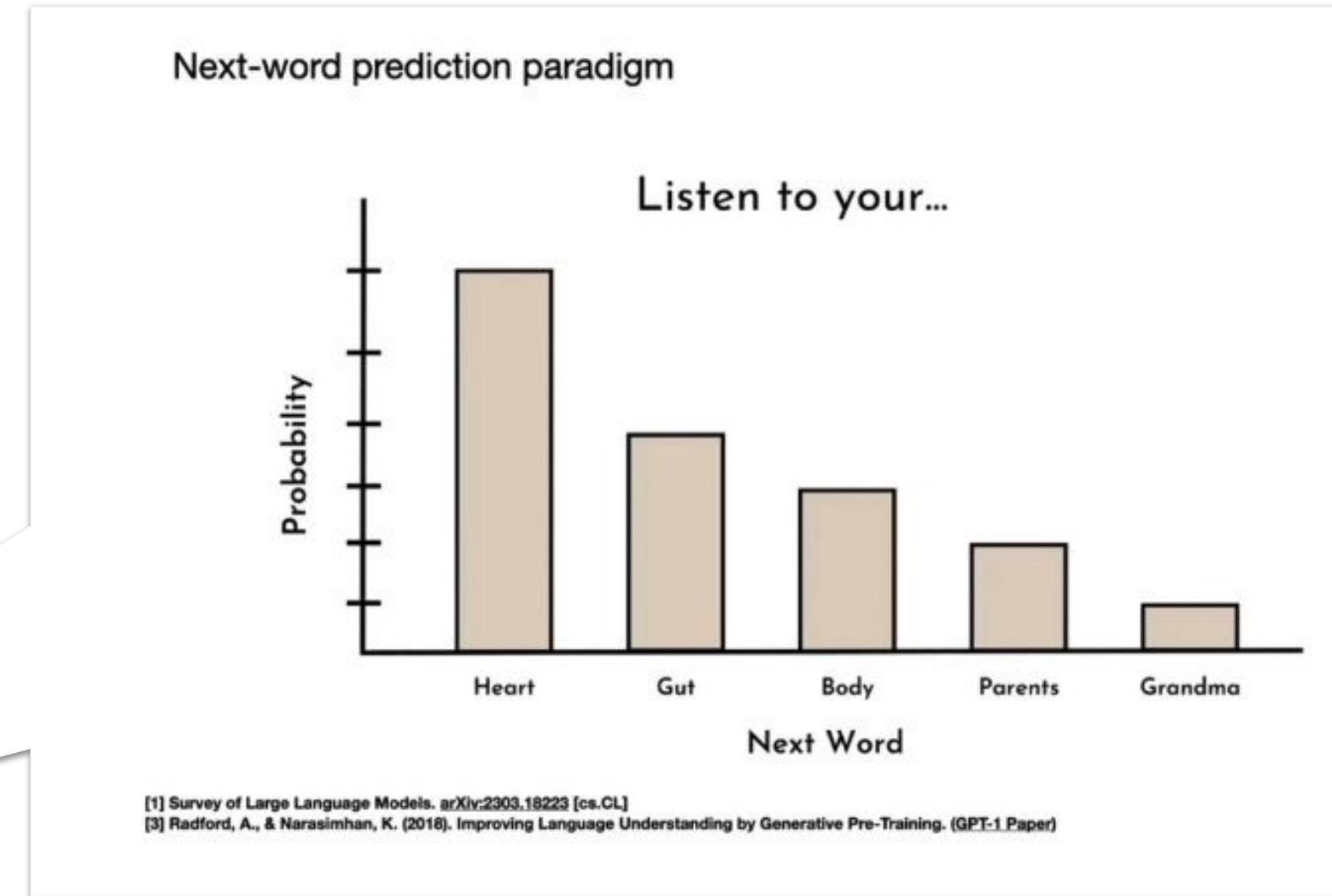
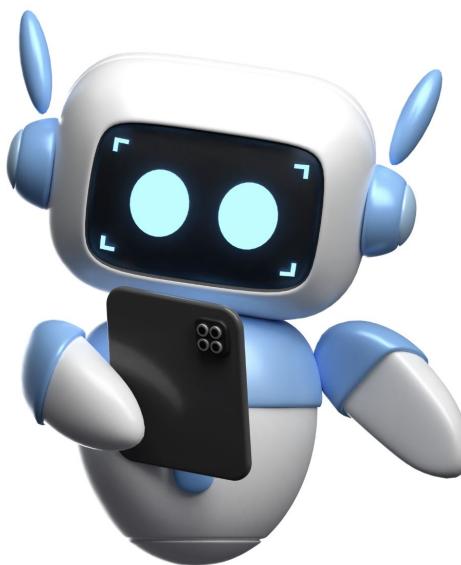
What is a Language Model?



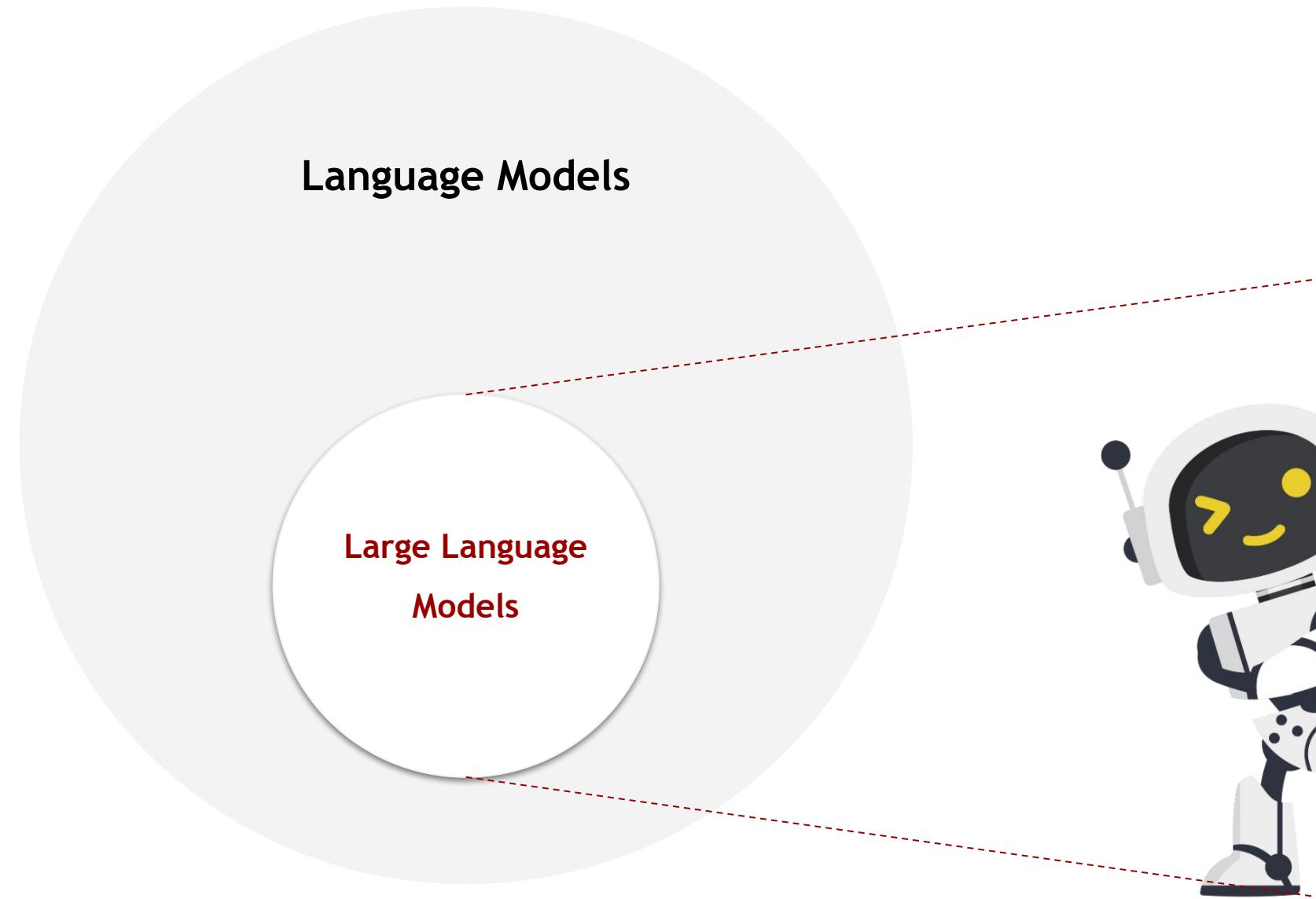
A language model is like a ***smart machine that helps you finish sentences***. It's trained to predict and complete sentences, making it a super useful tool when you're typing or texting



What is a Language Model?



What are Large Language Models?



Models trained on large amounts of all type of publicly-available data that can perform a wide variety of tasks, including text/image/document summarization, generation, and categorization. These models can perform generative tasks like text/image/document generation and so there is some overlap between LLMs and Generative AI.

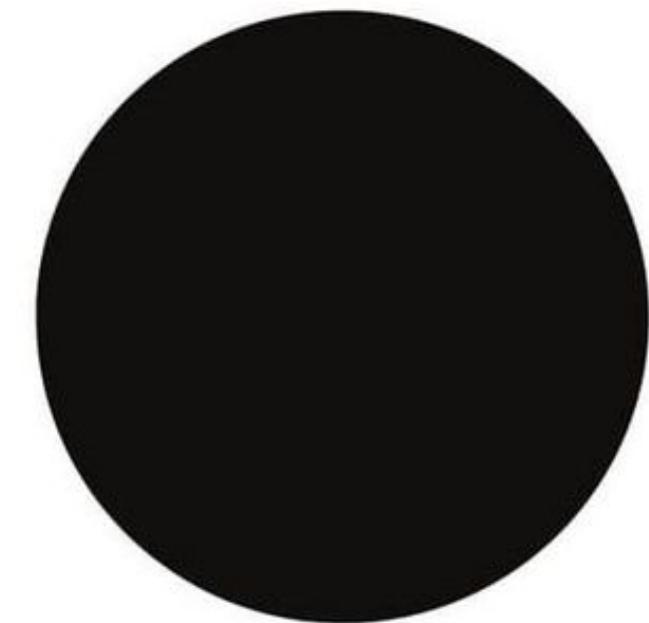
What is “Large” in Large Language Models?

- The term "large" refers to the size of the model in terms of its parameters and the volume of the training data
- Parameters are a part of the model that learns from historical training data
- They help in predicting the next word in a sequence
- One Trillion Parameters have been used in GPT 4

GPT-3



GPT-4

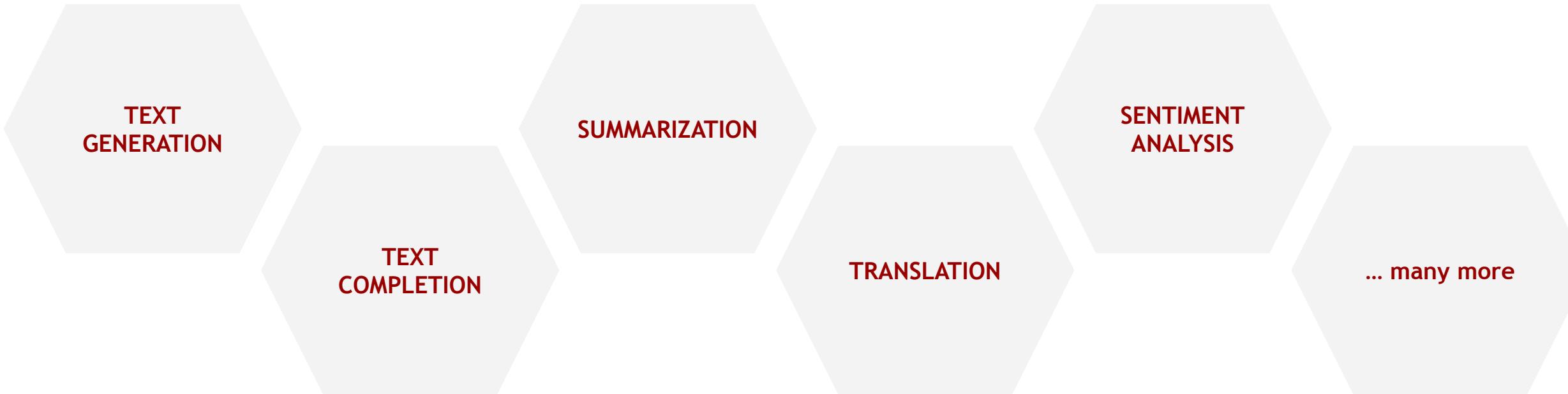


175.000.000.000

1.000.000.000.000.00

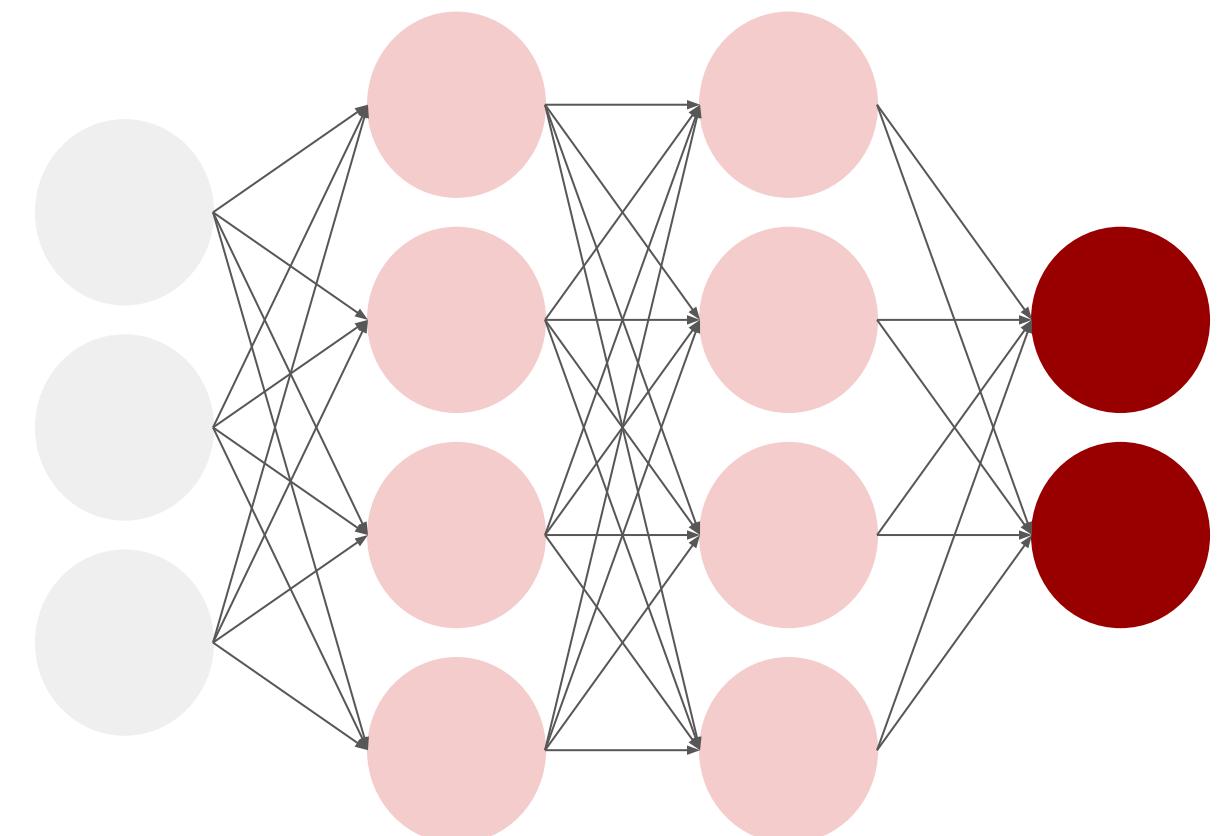
What is “Language” in Large Language Models?

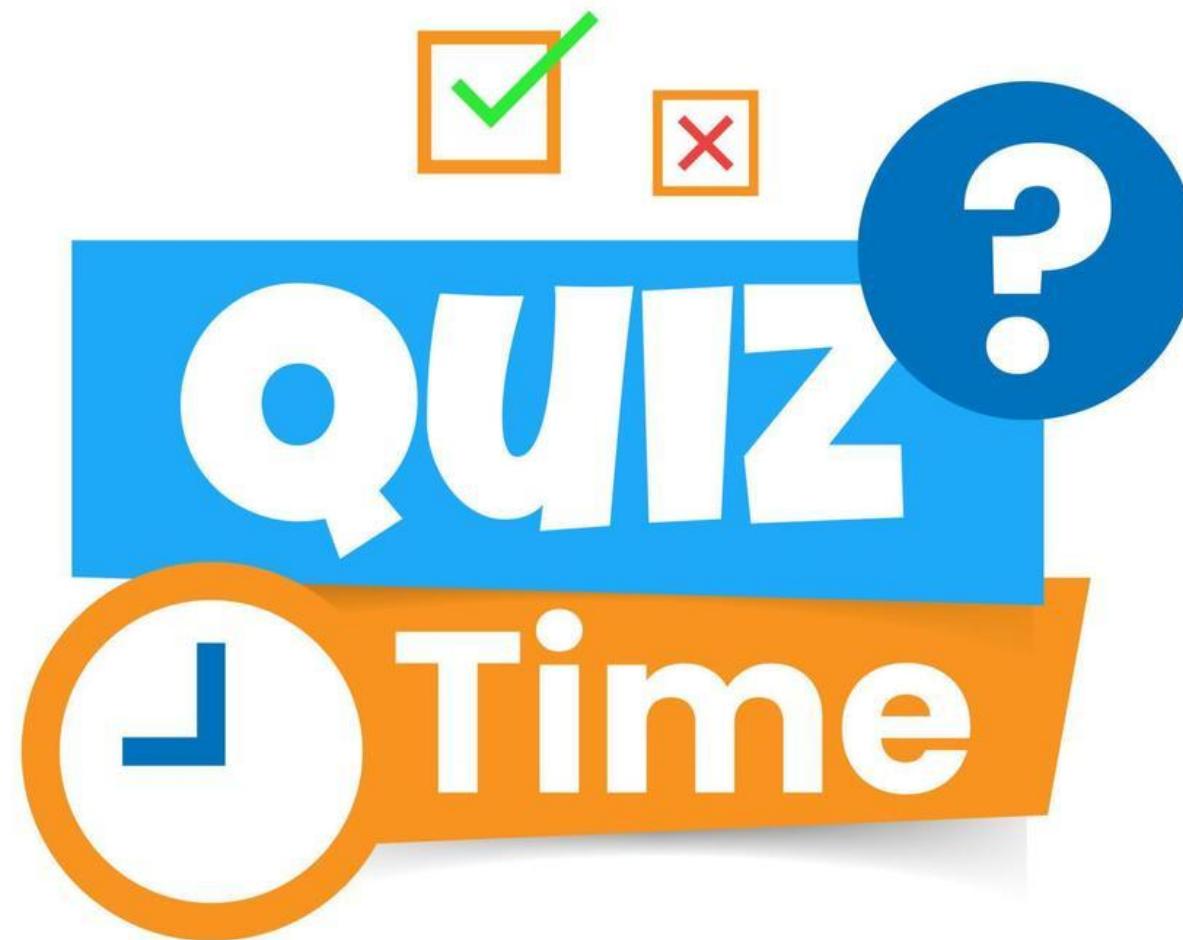
- Human-understandable languages like English
- The models learn statistical patterns of the data
 - Example: Probability of "the" preceding the noun "cat" are very high
- These models can generate new text, mimicking Humans
- Applications:-



What is a “Model” in Large Language Models?

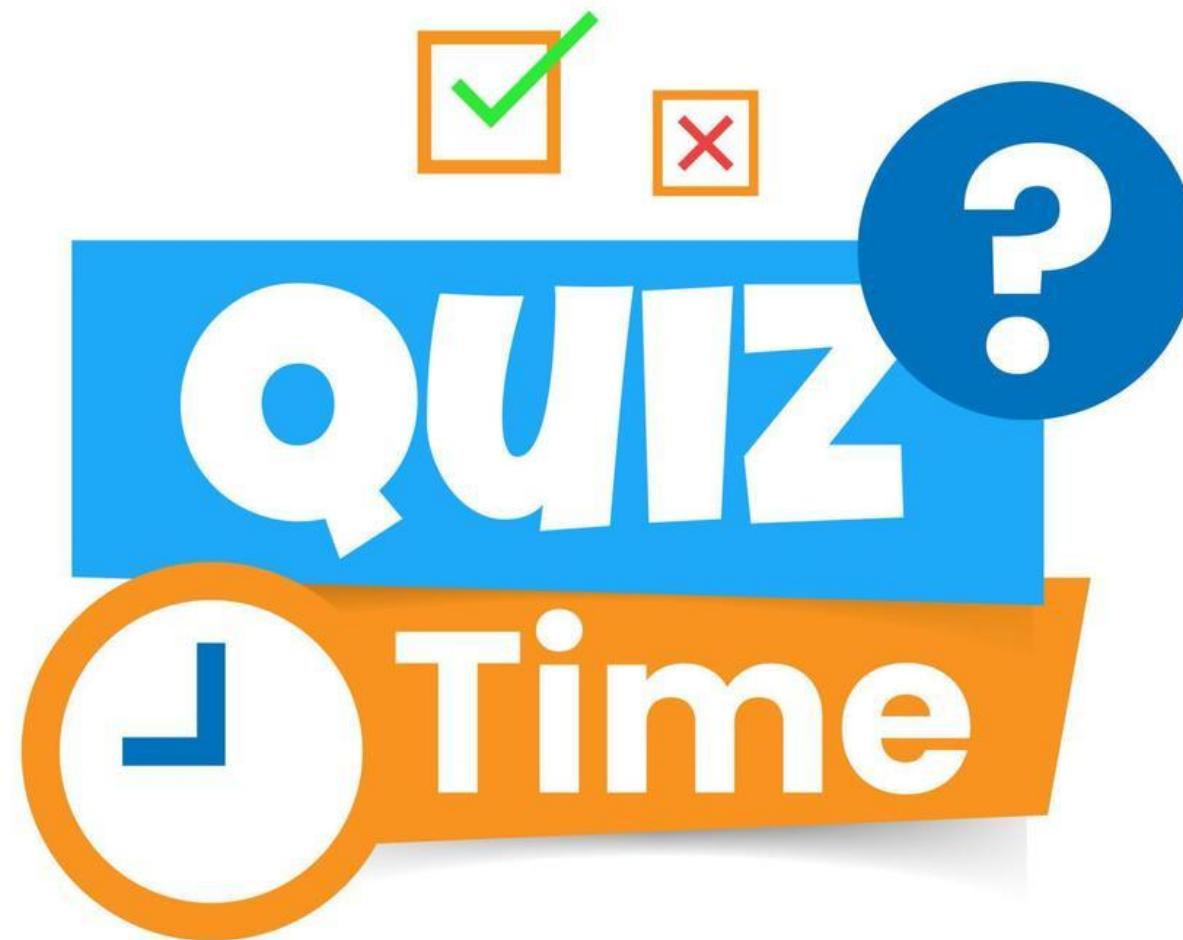
- A model refers to a mathematical representation of a real-world process
- It is essentially a program or an algorithm
 - Trained on a dataset to recognize patterns
 - Uses these patterns to make predictions or decisions without being explicitly programmed to perform the task
- ChatGPT uses GPT3 or GPT4 models





In the field of natural language processing, how are Large Language Models (LLMs) commonly applied?

| | |
|---|---|
| A | Generating human-like text in various languages |
| B | Analyzing complex mathematical equations |
| C | Designing computer hardware components |
| D | Controlling robotic movements in manufacturing |

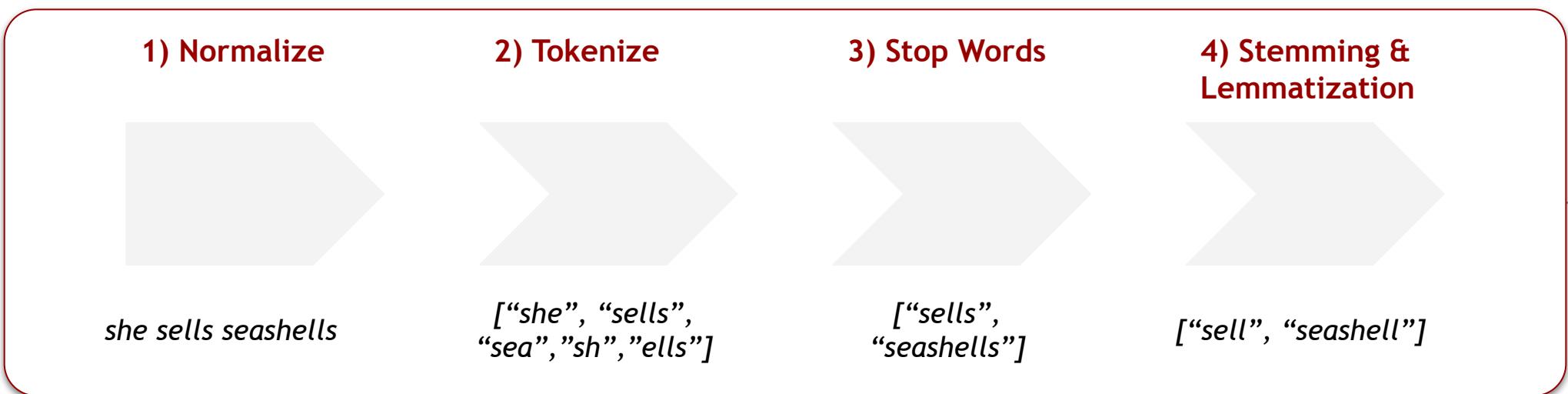


In the field of natural language processing, how are Large Language Models (LLMs) commonly applied?

| | |
|---|---|
| A | Generating human-like text in various languages |
| B | Analyzing complex mathematical equations |
| C | Designing computer hardware components |
| D | Controlling robotic movements in manufacturing |

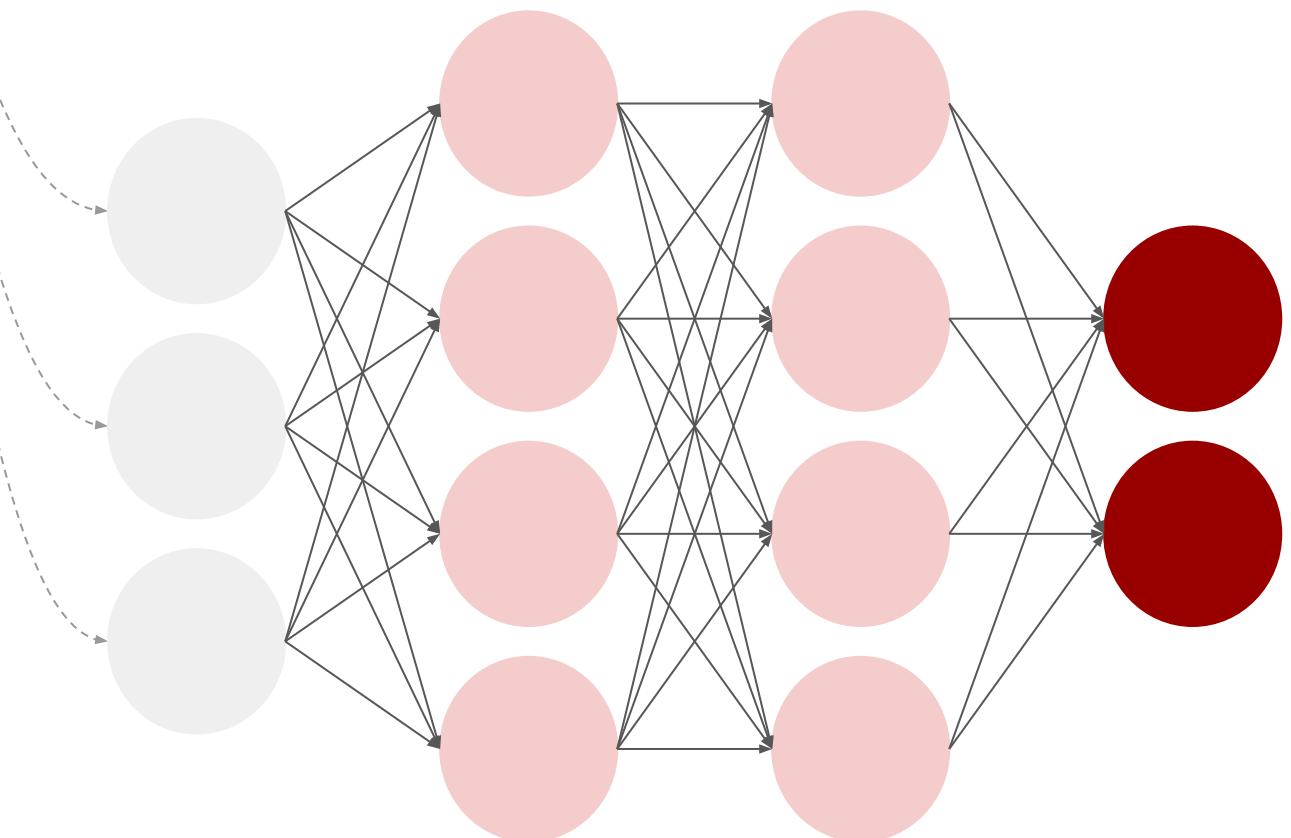
How a Large Language Model Works?

PRE-PROCESSING *(Example Text: She Sells Seashells)*



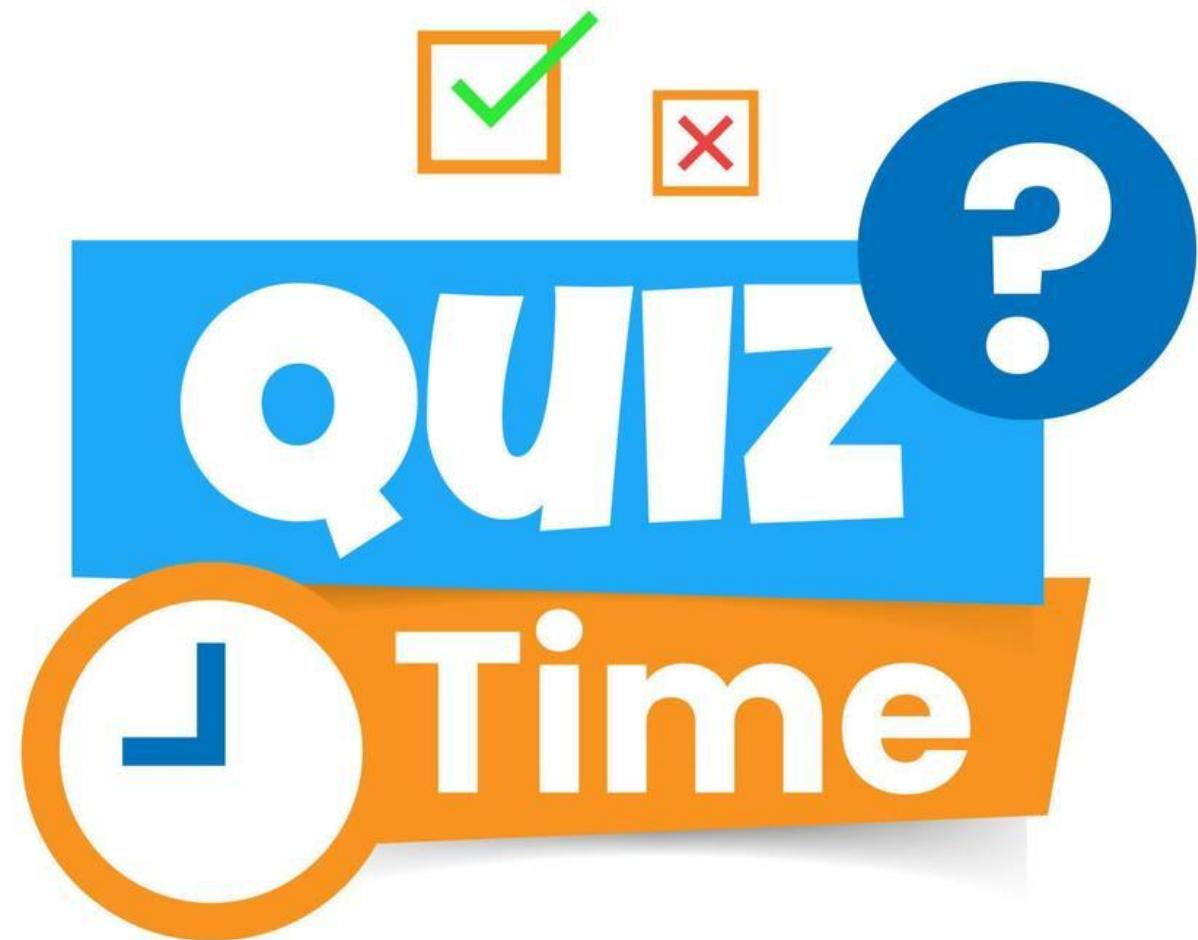
- **Text Normalization** is the process of converting text to a standard format, such as to lowercase, removing special characters, and converting numbers to their written form
- **Tokenization** is the process of breaking down text into individual units, such as words or phrases. This is an important step in preparing text data for NLP tasks
- **Stop Words** are common words that are usually removed during text processing, as they do not carry much meaning and can introduce noise or affect the results of NLP tasks. Examples of stop words include "the," "a," "an," "in," and "is"
- **Stemming and Lemmatization** are techniques used to reduce words to their base form. This helps reduce the dimensionality of the data and improve the performance of models

- LLMs are trained using a process called **unsupervised learning**
- This involves **feeding the model massive amounts of text data**, such as books, articles, and websites, and having the model learn the patterns and relationships between words and phrases in the text
- The model is then **fine-tuned on a specific task**, such as language translation or text summarization



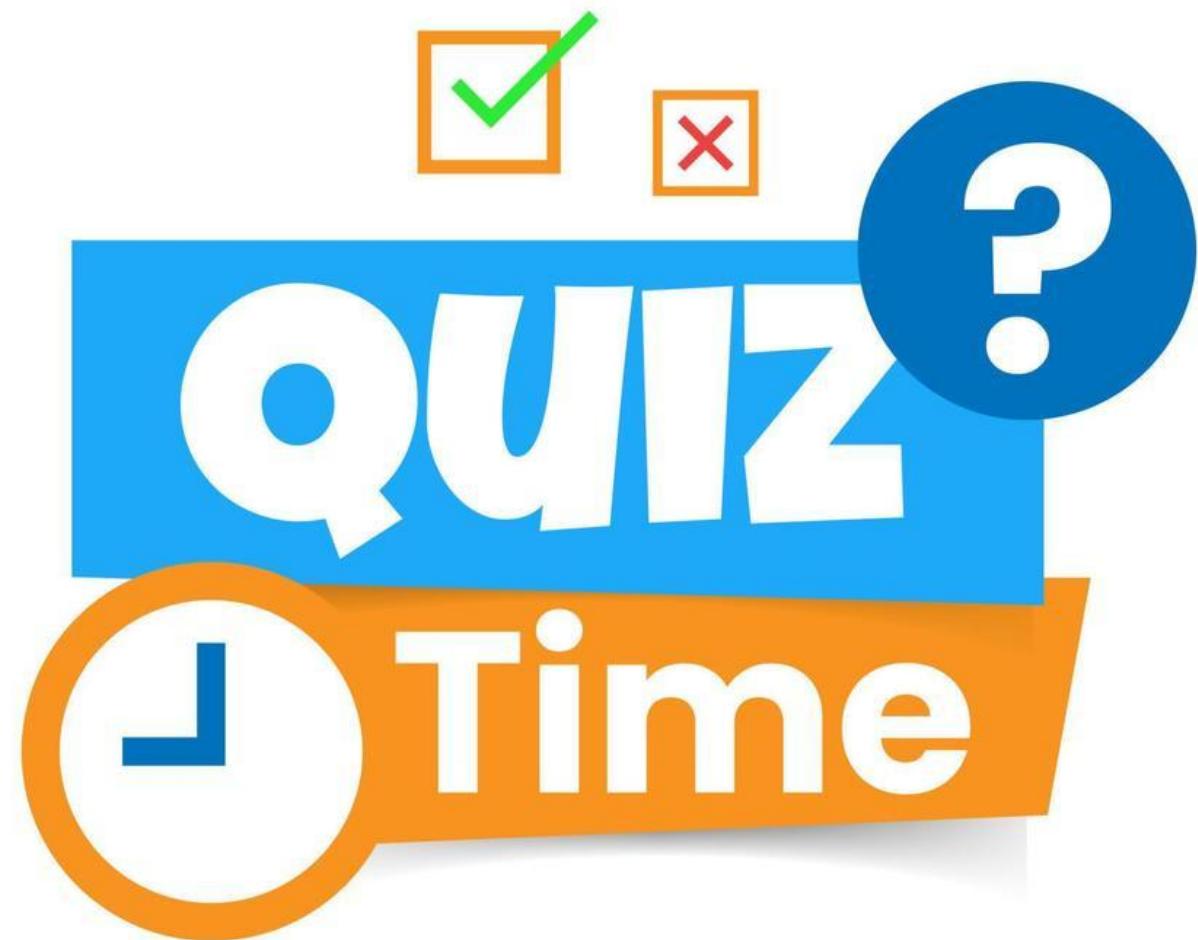
Challenges with Large Language Model

- One of the main challenges with LLMs is the **potential for offensive language**, as the models learn from the patterns found in the training data
- Unethical considerations, such as **gender and racial biases**
- **Amount of computational resources** needed to train and run LLMs, which can be expensive and energy-intensive
- **Making up things** which are not even facts
- While large language models have shown impressive performance on a variety of NLP tasks, they may not perform as well on specific tasks, such as those that require a **deeper understanding of the underlying context**



When Generative AI systems just make something up that isn't true, it is called what?

| | |
|---|---------------------------------------|
| A | A lie |
| B | A hallucination |
| C | A bug |
| D | It is not possible for that to happen |



When Generative AI systems just make something up that isn't true, it is called what?

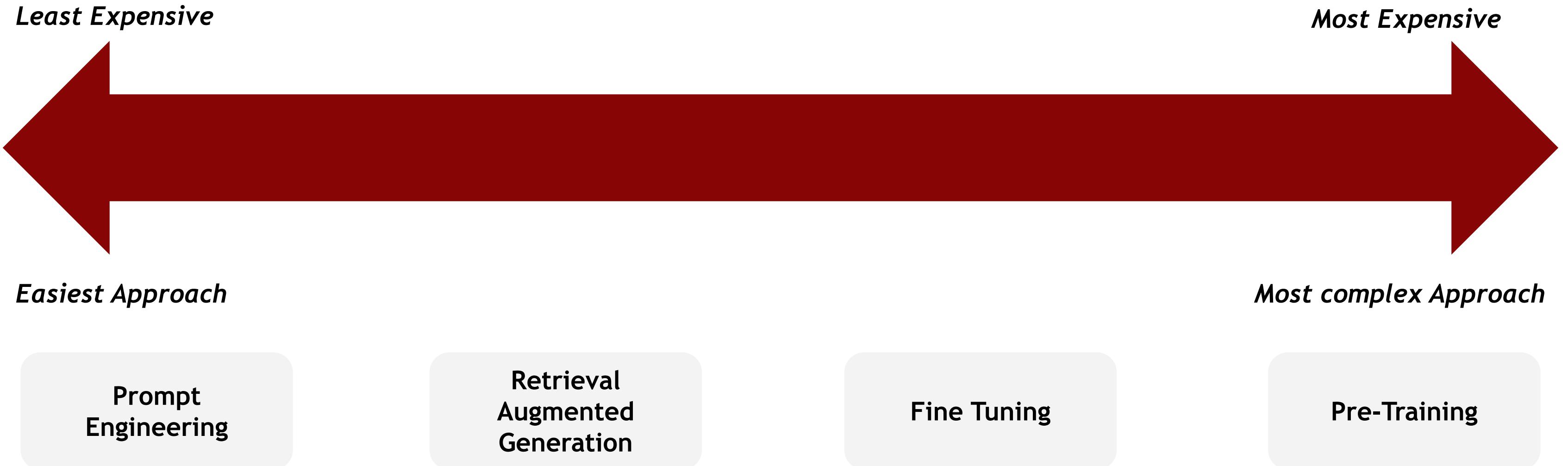
| | |
|---|---------------------------------------|
| A | A lie |
| B | A hallucination |
| C | A bug |
| D | It is not possible for that to happen |

Agenda

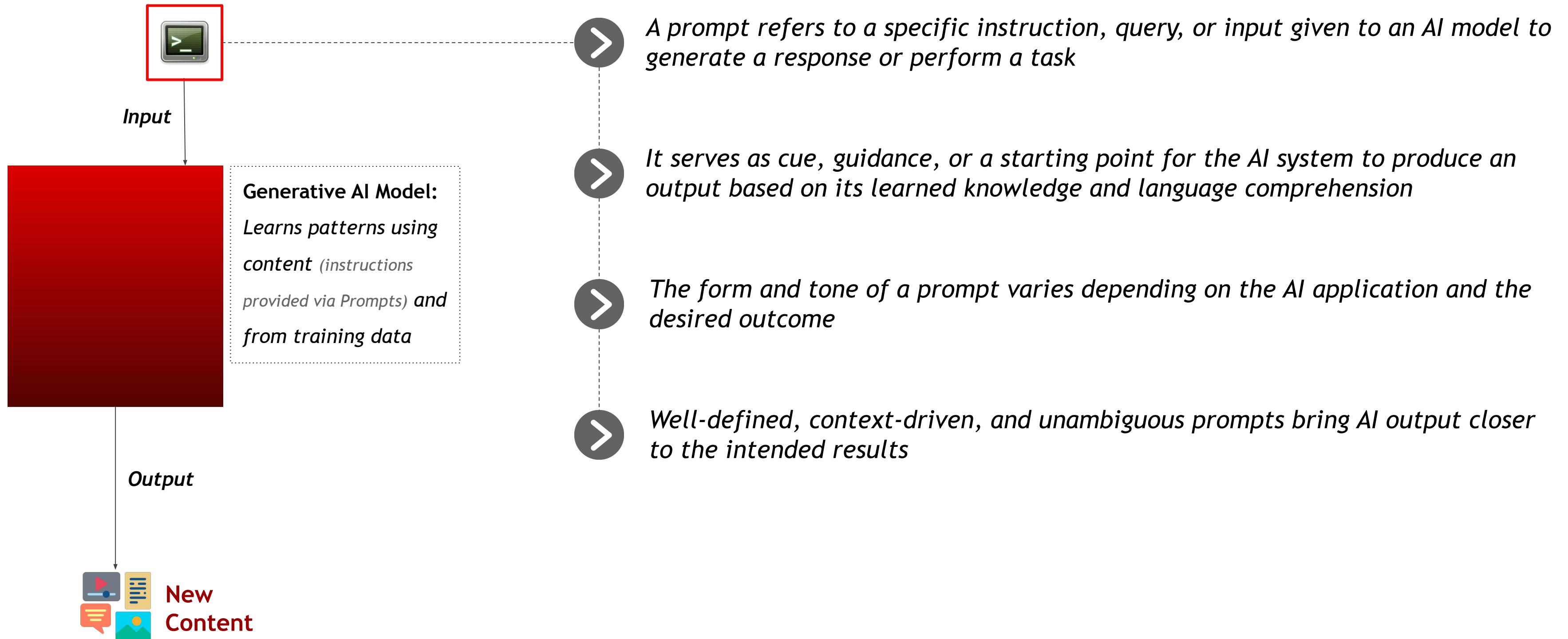
- Recap of LLMs
- **Understanding Prompt Design**
 - Context, Examples, & Tuning Prompt Parameters
 - Prompt Design Best Practices

Ways to customize Large Language Models

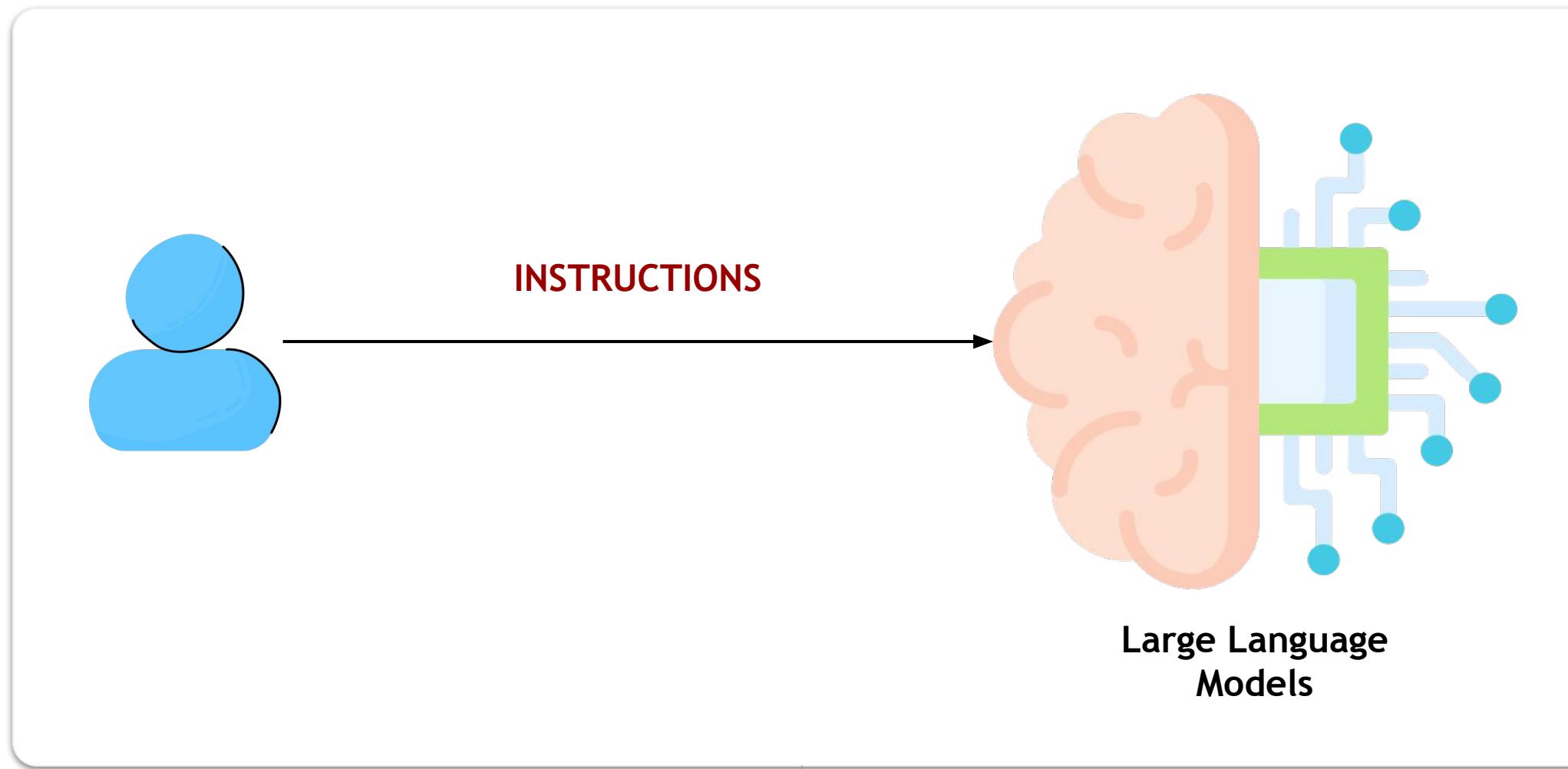
Accuracy Requirements, Skillset, Compute Resources, & Data Availability influence the choice of approach



What is a Prompt?



What is Prompt Engineering?

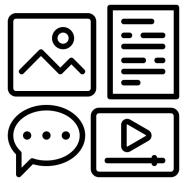


Prompt Engineering is an art of asking the right question to get the best response from an LLM

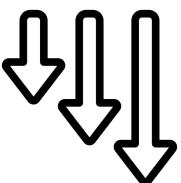
Why is Prompt Engineering Important?



Improve the Accuracy



Improve the usefulness of AI generated content



Reduce the risk of generating harmful and biased content

Key elements of a Prompt

Let's look at aspects that make up a good prompt:

Task

Clearly specify the desired action or response from the model

Context

Provide relevant background information to help the model understand the setting or scenario

Exemplar

Offer a sample of the desired output to guide the model

Persona

Define the character or role the model should respond to

Format

Specify the structure or style of the response

Tone

Clearly communicate the desired emotional or stylistic approach

More Important



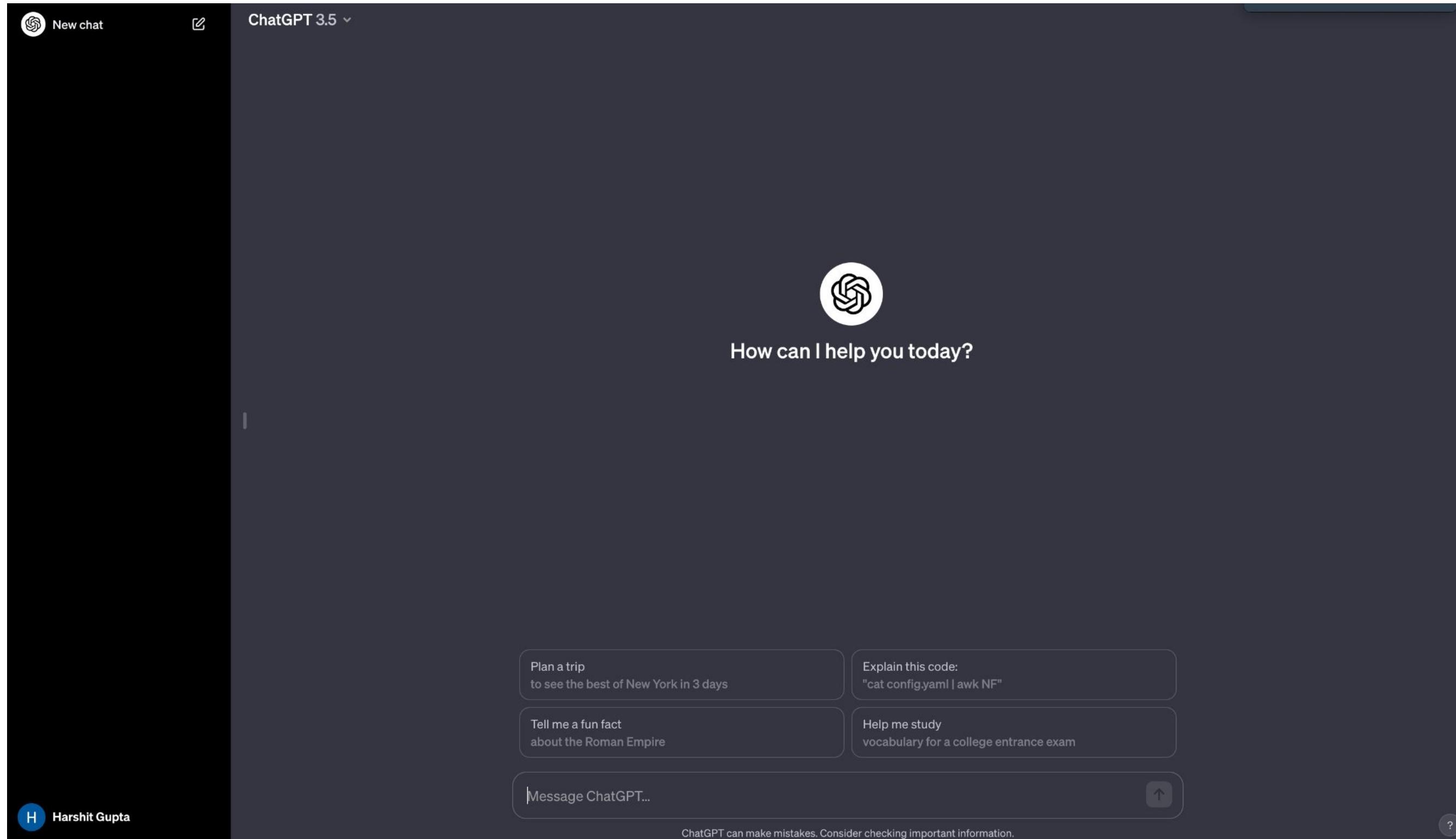
Optional

Good Prompts vs. Bad Prompts

The quality of the prompt significantly influences the output generated by the model. Here are the traits of Good & Bad Prompts:

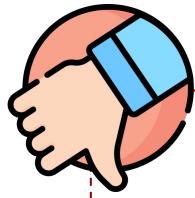


ChatGPT UI



Tips for designing Prompts

Write Clear Instructions



Tell me something about Generative AI

Change your prompt as per following to make it clearer:

- *Include details in your query to get more relevant answers*
- *Ask the model to adopt a persona*
- *Use delimiters to clearly indicate distinct parts of the input*
- *Specify the steps required to complete a task*
- *Provide examples*
- *Specify the desired length of the output*



Act as an AI Engineer working for a multinational Consumer Packaged Goods (CPG) organization. One of the CXOs have asked you to explain about “What is Generative AI?” through a concise article, just to keep up with the latest trends in the market. Now provide me with an explanation of Generative AI using following instructions:

1. *It should be easy to understand and must be tweaked as per CPG Industry*
2. *Overall length of article should not be more than 500 words*
3. *Add relevant examples and some cases with the source links*

Tips for designing Prompts

Ask to provide reference text when required



Share some case studies related to the adoption of Generative AI in consumer packaged goods field.



Share some case studies related to the adoption of Generative AI in consumer packaged goods domain. You can take references from websites like gartner.com, mckinsey.com, forbes.com

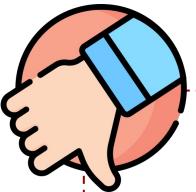
Also add proper source links for the case studies shared in the response.

Change your prompt like this to make it clearer:

- *Instruct the model to answer using a reference text/website*
- *Instruct the model to answer with citations from a reference text*

Tips for designing Prompts

Split a complex task into simpler subtasks



In order to propel our Consumer Packaged Goods (CPG) company into the next era of operational excellence, we are seeking a comprehensive strategy to evolve from a Data-Driven Organization to an AI-Driven Organization. Please provide a detailed plan

Use the following tactics to improve the prompt:

- Use intent classification to identify the most relevant instructions for a user query
- For dialog-driven applications that require very long conversations, summarize or filter previous dialog(s)
- Summarize long documents piecemeal and construct a complete summary, recursively



To propel our CPG organization towards an AI-driven future, we seek a detailed strategy for transitioning from a Data-Driven to an AI-Driven Organization. Key focus areas include:

- Data Integration and Enrichment:
 - Propose methods for enhanced data integration, ensuring seamless interoperability and enrichment from external sources
- Advanced Analytics Capabilities:
 - Specify tools to extract actionable insights, surpassing traditional analytics, and gaining predictive capabilities
- Machine Learning Applications:
 - Identify use cases for machine learning in decision-making, supply chain management, and demand forecasting, with a scalable implementation plan
- Automation and Process Enhancement:
 - Propose AI-driven automation solutions to streamline tasks, enhance efficiency, and address integration challenges
- Cultural Shift and Skill Development:
 - Devise a plan for fostering AI adoption culture, promoting awareness, enthusiasm, and workforce skill development
- Performance Monitoring and Iterative Improvement:
 - Establish KPIs for measuring AI initiative success, implement a monitoring framework, and detail a feedback loop for continuous improvement

Provide a comprehensive, phased roadmap aligning with organizational goals, emphasizing smooth transition and sustained evolution. The plan should prioritize data integration, advanced analytics, machine learning, automation, cultural adaptation, and iterative improvement for a successful shift to an AI-Driven Organization

Tips for designing Prompts

Test changes systematically. Playground is a nice way to test for Developers

Evaluate model outputs with reference to gold-standard answers



You will be provided with text delimited by triple quotes that is supposed to be the answer to a question. Check if the following pieces of information are directly contained in the answer:

- Neil Armstrong was the first person to walk on the moon.
- The date Neil Armstrong first walked on the moon was July 21, 1969.

For each of these points perform the following steps:

- 1 - Restate the point.
- 2 - Provide a citation from the answer which is closest to this point.
- 3 - Consider if someone reading the citation who doesn't know the topic could directly infer the point. Explain why or why not before making up your mind.
- 4 - Write "yes" if the answer to 3 was yes, otherwise write "no".

Finally, provide a count of how many "yes" answers there are. Provide this count as {"count": <insert count here>}.

"""\Neil Armstrong is famous for being the first human to set foot on the Moon. This historic event took place on July 21, 1969, during the Apollo 11 mission.""""

Techniques in Prompt Engineering

Here, we see more intricate strategies that require a deeper understanding of the model's behavior

ZERO-SHOT PROMPTING

This technique involves providing the model with a task it hasn't seen during its training. It tests the model's ability to generalize and produce relevant outputs without relying on prior examples

Classify the text into neutral, negative or positive.

Text: I think the vacation was okay.

Sentiment:

Neutral

Techniques in Prompt Engineering

Here, we see more intricate strategies that require a deeper understanding of the model's behavior

FEW-SHOT PROMPTING

Here, the model is given a few examples (shots) to guide its response. By providing exemplars, the model can better understand and generate the desired output. For example, showing a model several examples of translated sentences before asking it to translate a new one

Craft a compelling tagline for social media introducing our upcoming product, a sustainable packaging initiative for our soft-drink line. Highlight the eco-friendly features and emphasize the positive impact on the environment.

Example1: Every sip is a step towards a greener future

Example2: Good for you, great for the Earth

Example3: Because flavor shouldn't cost the Earth. Share the taste, share the planet

Sip sustainably, savor responsibly

Techniques in Prompt Engineering

Here, we see more intricate strategies that require a deeper understanding of the model's behavior

CHAIN OF THOUGHTS (COT)

It involves guiding the model through a series of reasoning steps. By breaking down a complex task into intermediate steps or a "chains of reasoning," the model can achieve better language understanding and more accurate outcomes

Process the problem step-by-step and answer the question asked:

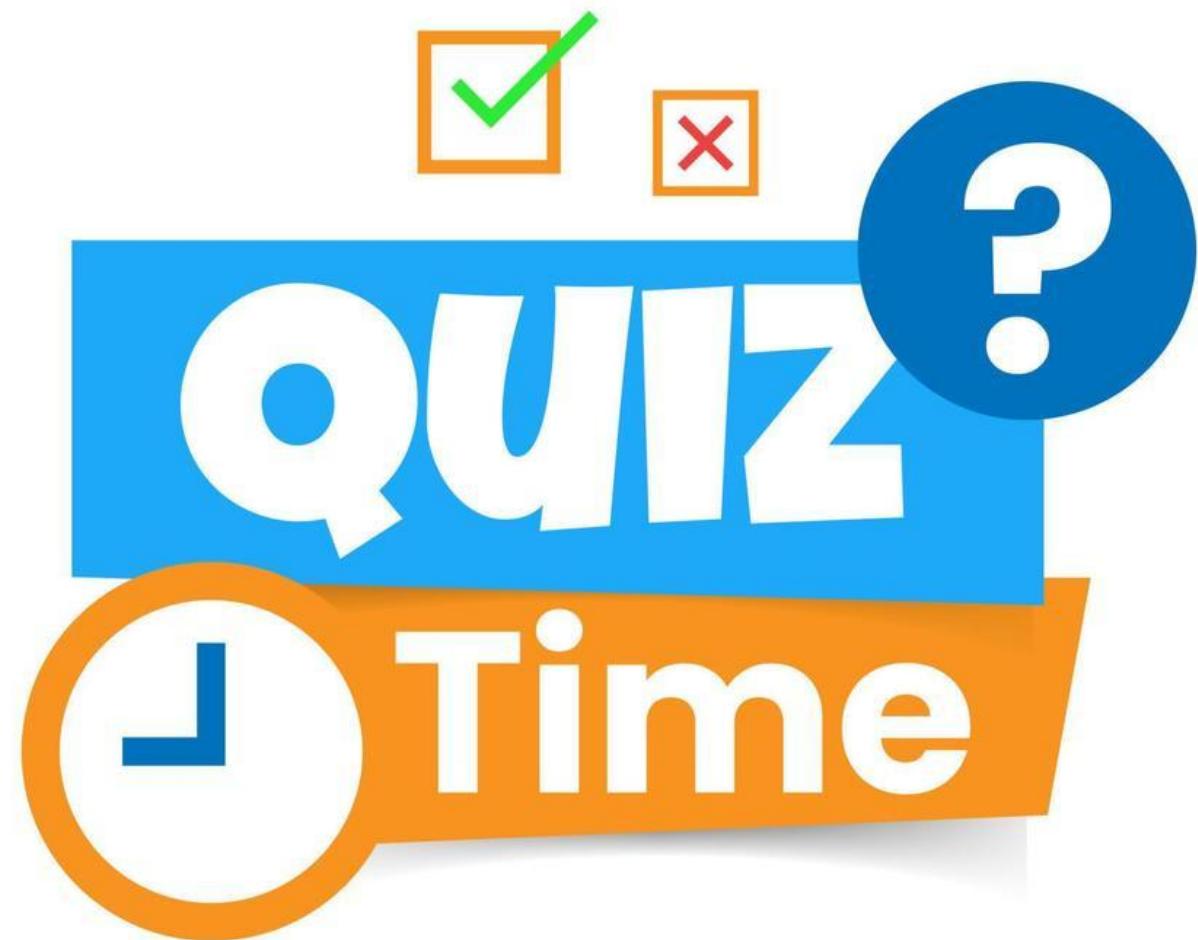
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

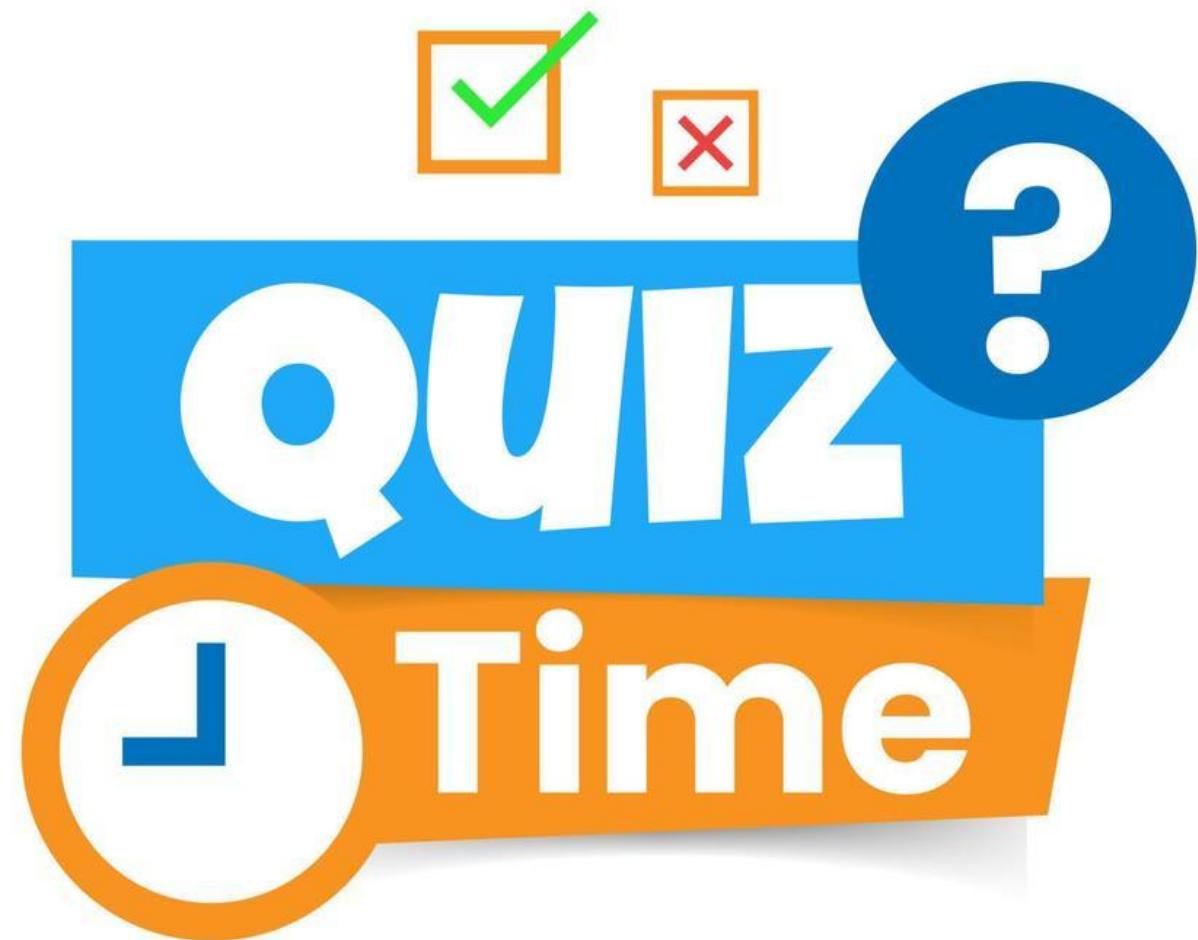
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$.

They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9



If you want the model to write in your style what should you do?

| | |
|---|--------------------|
| A | Use clear language |
| B | Add examples |
| C | Add context |
| D | Add purpose |



If you want the model to write in your style what should you do?

| | |
|---|--------------------|
| A | Use clear language |
| B | Add examples |
| C | Add context |
| D | Add drive purpose |

Agenda

- Recap of LLMs
 - Understanding Prompt Design
- • **Context, Examples, & Tuning Prompt Parameters**
- Prompt Design Best Practices

OpenAI Developer Playground

The side menu bar includes options for **Playground, Assistant, Fine Tuning, API Keys, Files, Usage Stats, and Settings**

The screenshot shows the OpenAI Developer Playground interface. On the left is a sidebar with a logo at the top, followed by several icons: a green square with a white icon, a gear, a lock, a folder, a file, a person, and a gear. A red box highlights this sidebar area. To the right is the main 'Playground' window. At the top of the playground window is a header with the title 'Playground' and a 'Chat' dropdown. To the right of the header are buttons for 'Your presets', 'Save', 'View code', 'Share', and three dots. Below the header is a message from the 'SYSTEM': 'You are a helpful assistant.' Below this is a 'USER' input field with the placeholder 'Enter a user message here.' and a 'Add message' button. A red box highlights the 'SYSTEM' message and the 'USER' input area. In the bottom right corner of the playground window is a note: 'API and Playground requests will not be used to train our models. [Learn more](#)'.

Playground Chat

Your presets Save View code Share ...

Get started

Enter an instruction or select a preset, and watch the API respond with a [completion](#) that attempts to match the context or pattern you provided.

You can control which [model](#) completes your request by changing the model.

KEEP IN MIND

- Use good judgment when sharing outputs, and attribute them to your name or company. [Learn more](#).
- Requests submitted to our API and Playground will not be used to train or improve future models. [Learn more](#).
- Our default models' training data cuts off in 2021, so they may not have knowledge of current events.

SYSTEM
You are a helpful assistant.

USER Enter a user message here.
Add message

Model gpt-3.5-turbo

Temperature 1

Maximum length 256

Stop sequences Enter sequence and press Tab

Top P 1

Frequency penalty 0

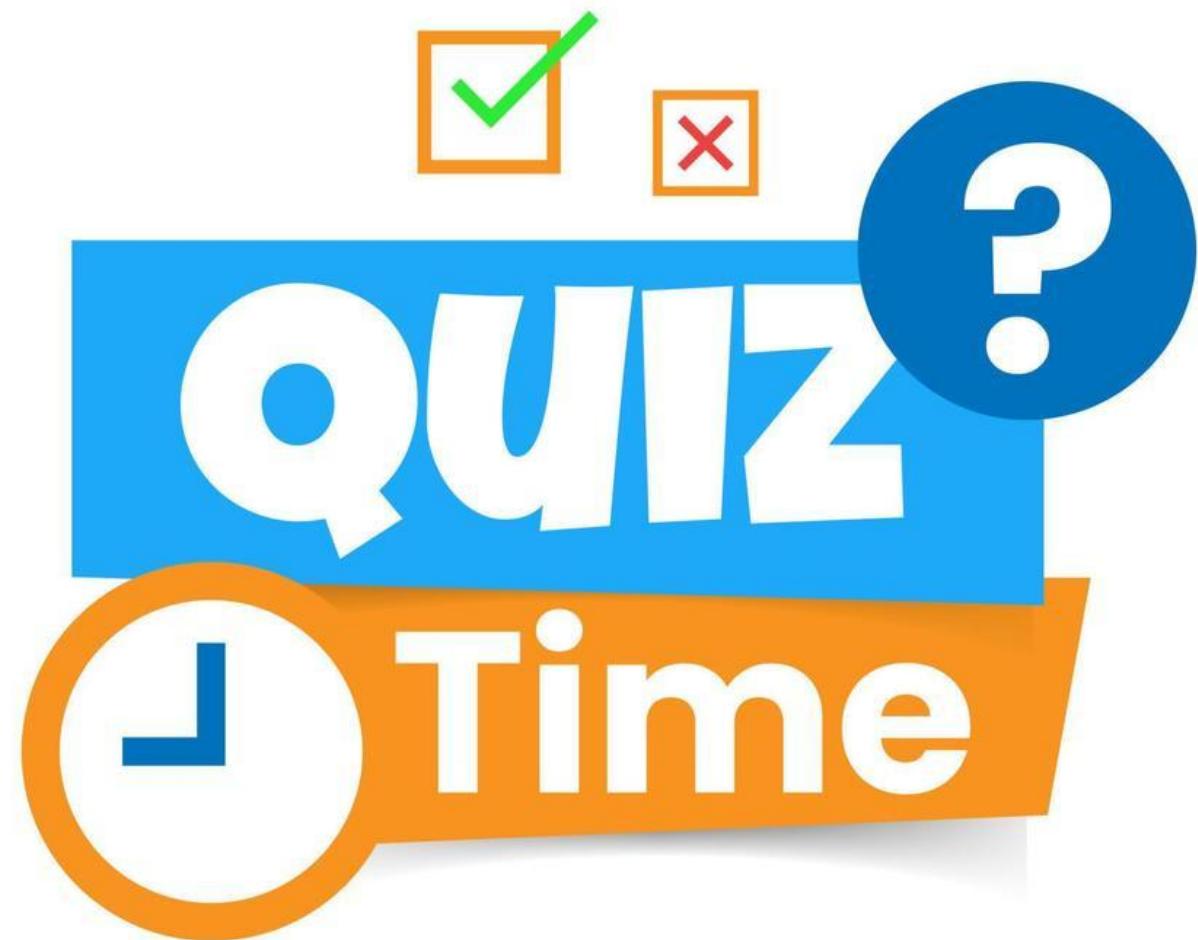
Presence penalty 0

API and Playground requests will not be used to train our models. [Learn more](#)

Add the context and scenarios for the system to provide the responses in certain tone

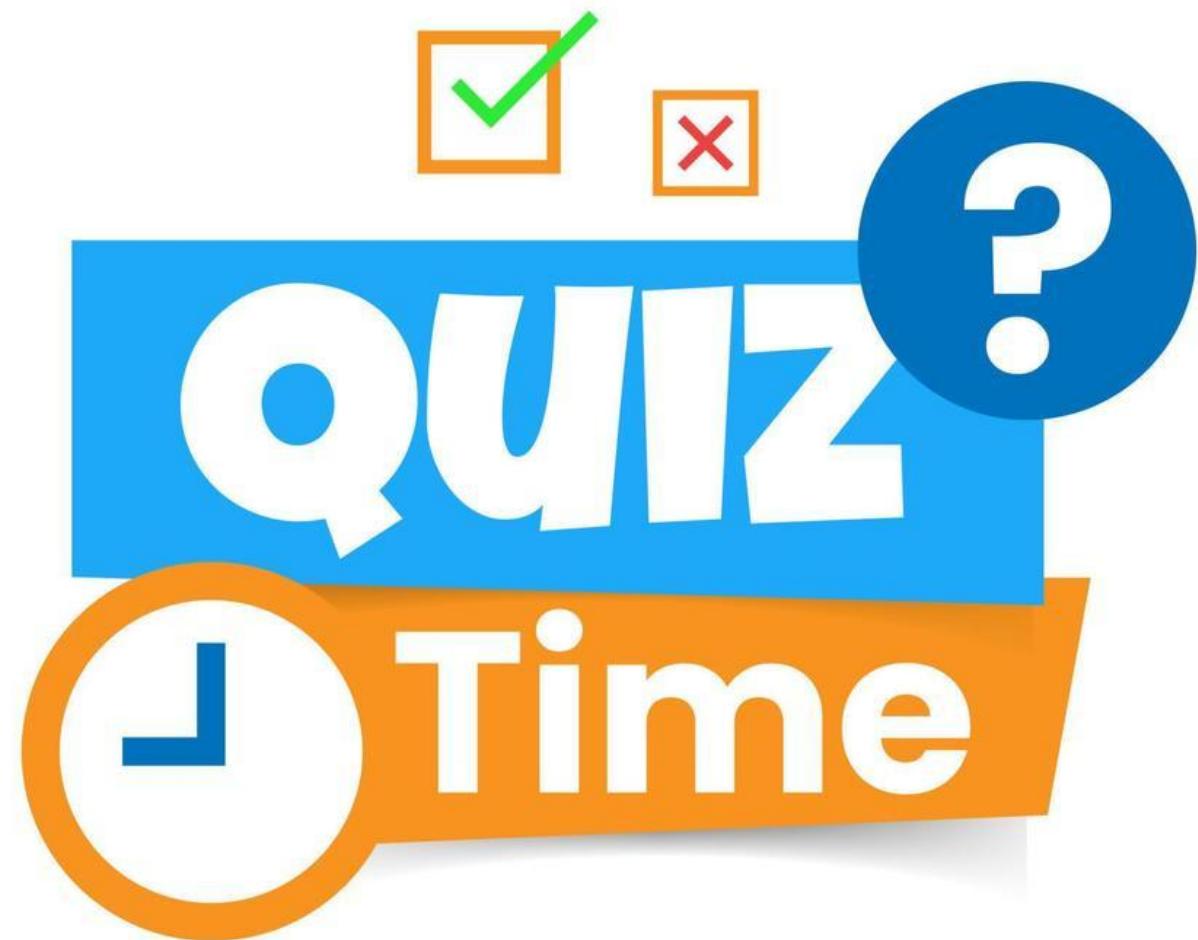
Provide user question or inputs for which you are seeking responses

Tune the responses using these parameters



Which setting for temperature would make the model more creative when answering prompts?

| | |
|---|-----|
| A | -1 |
| B | 0 |
| C | 0.1 |
| D | 0.9 |

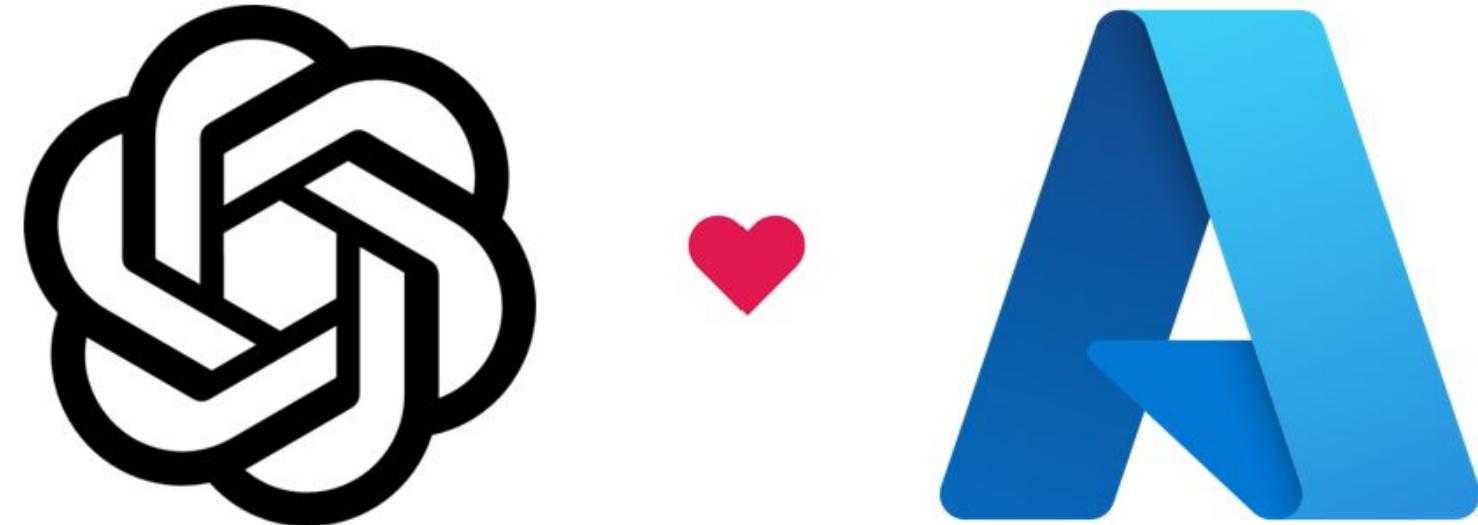


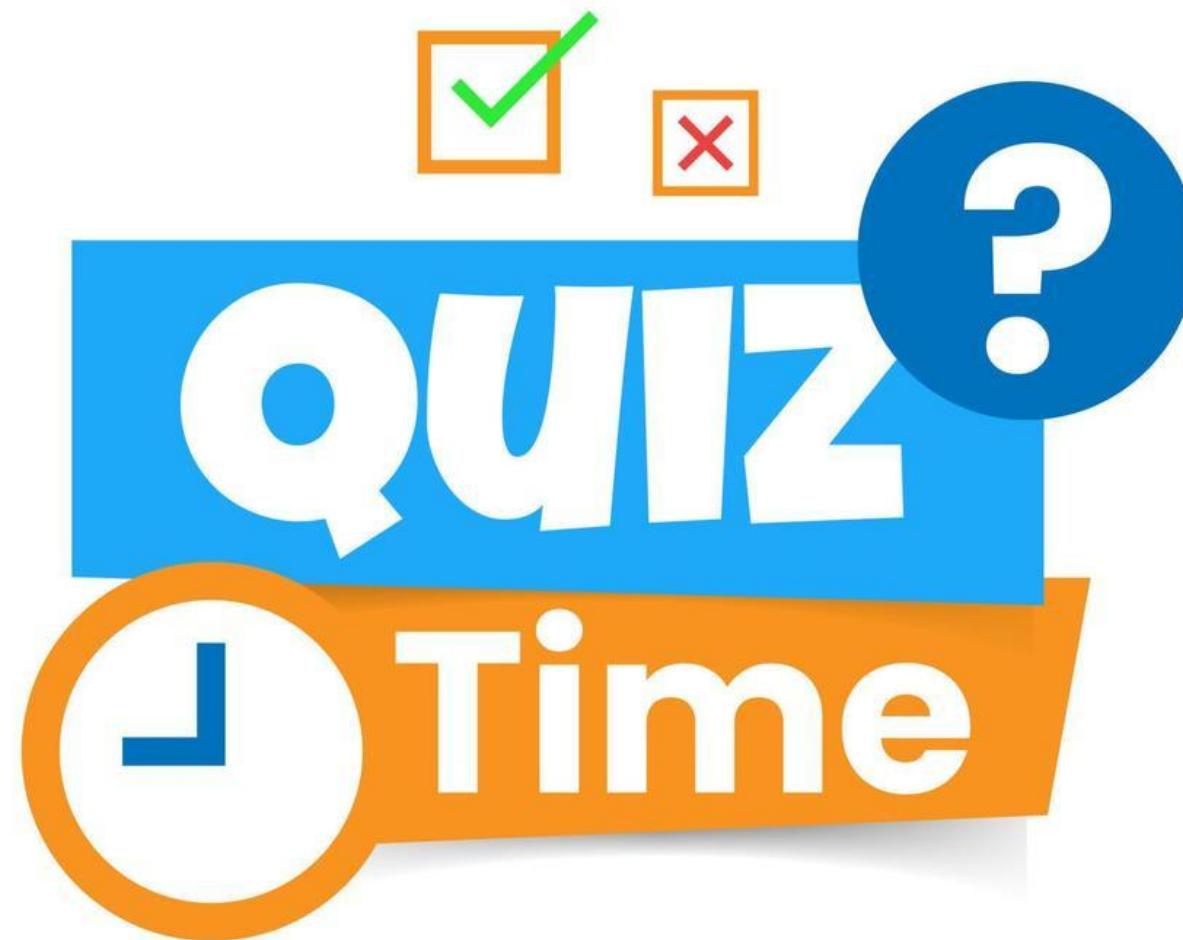
Which setting for temperature would make the model more creative when answering prompts?

| | |
|---|-----|
| A | -1 |
| B | 0 |
| C | 0.1 |
| D | 0.9 |

Freeform vs Structured Prompts in Azure OpenAI Studio

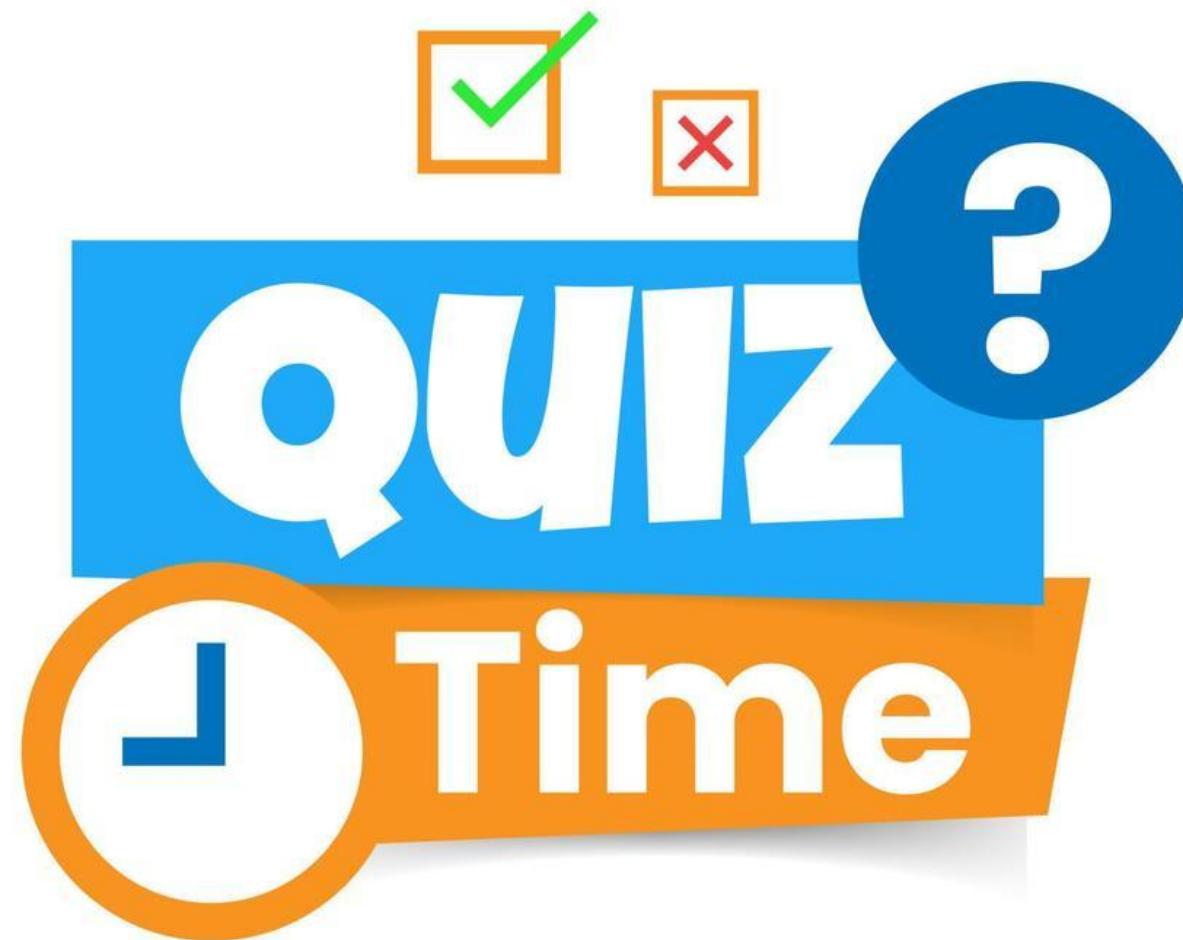
- So far in class, all the examples shown have been freeform prompts
 - Consist only of the prompt
- Structured prompts in Azure OpenAI Studio allow you to add system messages(context) and examples
 - Context tells the model more about your use case characteristics
 - Examples show the model how to structure the response





What is difference between Freeform and Structured prompts?

| | |
|---|--|
| A | Freeform prompts restrict input options, while Structured prompts lack context |
| B | Freeform prompts and Structured prompts have no notable differences |
| C | With structured prompts, you can add context and examples |
| D | Freeform prompts are more contextually rich, while Structured prompts limit input variations |



What is difference between Freeform and Structured prompts?

| | |
|---|--|
| A | Freeform prompts restrict input options, while Structured prompts lack context |
| B | Freeform prompts and Structured prompts have no notable differences |
| C | With structured prompts, you can add context and examples |
| D | Freeform prompts are more contextually rich, while Structured prompts limit input variations |

Azure OpenAI Studio UI for Prompts

A Azure AI Studio PUBLIC PREVIEW X

Presenting the new Azure AI Studio (Preview)

Build, evaluate, and deploy your AI solutions from end to end.

[Explore Azure AI Studio](#)

[Azure AI Studio](#) [Privacy & cookies](#)

[Azure OpenAI](#)

Welcome to Azure OpenAI service

Explore the generative AI models, craft unique prompts for your use cases, and fine-tune select models.

Get started

Chat playground
Design a customized AI assistant using ChatGPT. Experiment with GPT-3.5-Turbo and GPT-4 models.
[Try it now](#)

Completions playground
Experiment with completions models for use cases such as summarization, content generation, and classification.
[Try it now](#)

DALL-E playground
PREVIEW
Generate unique images by writing descriptions in natural language.
[Try it now](#)

Bring your own data
PREVIEW
Connect and ground your data. Deploy to a web app or Power Virtual Agent bot.
[Try it now](#)

Try out common examples

Customer support agent
Get started with a sample chat setup for a customer support agent scenario

Writing assistant
Get started with a sample chat setup for marketing content creation

Summarize an article
Get started with abstractive summarization of a news article

Create cover art
Unleash your creativity by generating creative images for cover art and other commercial use cases

Login using your Azure OpenAI Credentials at: <https://oai.azure.com/>

Azure OpenAI Studio UI for Prompts

A Azure AI Studio PUBLIC PREVIEW X

Presenting the new Azure AI Studio (Preview)

Build, evaluate, and deploy your AI solutions from end to end.

[Explore Azure AI Studio](#)

« Azure AI Studio Privacy & cookies ▾

[Azure OpenAI](#)

Playground

[Chat](#) Select the Chat option out of many options available on the left hand side section of the page

[Completions](#)

[DALL-E \(Preview\)](#)

Management

[Deployments](#)

[Models](#)

[Data files](#)

[Quotas](#)

[Content filters \(Preview\)](#)

Welcome to Azure OpenAI service

Explore the generative AI models, craft unique prompts for your use cases, and fine-tune select models.

Get started

Text generation

Chat playground
Design a customized AI assistant using ChatGPT. Experiment with GPT-3.5-Turbo and GPT-4 models.
[Try it now](#)

Text generation

Completions playground
Experiment with completions models for use cases such as summarization, content generation, and classification.
[Try it now](#)

Image generation

DALL-E playground PREVIEW
Generate unique images by writing descriptions in natural language.
[Try it now](#)

Text generation

Bring your own data PREVIEW
Connect and ground your data. Deploy to a web app or Power Virtual Agent bot.
[Try it now](#)

Try out common examples

Customer support agent
Get started with a sample chat setup for a customer support agent scenario

Writing assistant
Get started with a sample chat setup for marketing content creation

Summarize an article
Get started with abstractive summarization of a news article

Create cover art
Unleash your creativity by generating creative images for cover art and other commercial use cases

Azure OpenAI Studio UI for Prompts

The screenshot shows the Azure OpenAI Studio UI for Prompts in PUBLIC PREVIEW. The interface is divided into several sections:

- Header:** Azure AI Studio PUBLIC PREVIEW. A close button (X) is in the top right.
- Title Bar:** Presenting the new Azure AI Studio (Preview). Below it, a sub-header says Build, evaluate, and deploy your AI solutions from end to end.
- Left Sidebar:** A navigation menu with the following items:
 - Azure OpenAI
 - Playground
 - Chat** (selected)
 - Completions
 - DALL-E (Preview)
 - Management
 - Deployments
 - Models
 - Data files
 - Quotas
 - Content filters (Preview)
- Central Chat Area:** Contains:
 - System message** (selected) and **Add your data (preview)**.
 - Save changes** button.
 - Specify how the chat should act**: A section for defining system messages. It includes a "Use a template" dropdown set to "Select a template", a "System message" input box containing "You are an AI assistant that helps people find information.", and an "Examples" section with a "+ Add an example" button.
 - Start chatting** section with a robot icon, instructions to "Test your assistant by sending queries below. Then adjust your assistant setup to improve the assistant's responses.", and a "Type user query here. (Shift + Enter for new line)" input field.
 - Top controls: Clear chat, Playground Settings, View code, Show raw JSON, and a refresh icon.
- Right Sidebar:** Deployment parameters.
 - Deployment** (selected) and Parameters.
 - Deployment** dropdown set to "test-01".
 - Session settings**: Includes "Past messages included" (set to 10), "Current token count" (11/4000), and "Input tokens progress indicator".

Azure OpenAI Studio UI for Prompts

The screenshot illustrates the Azure OpenAI Studio UI for Prompts, featuring three main panels: Assistant setup, Chat session, and Configuration.

Assistant setup: This panel allows users to specify how the chat should act. It includes a "Select a template" dropdown and a "System message" input field containing the placeholder text: "You are an AI assistant that helps people find information." A blue callout box highlights this system message area with the text: "System message adds information about the use case".

Chat session: This panel contains a "Chat session" header with controls for "Clear chat", "Playground Settings", "View code", and "Show raw JSON". It features a "Start chatting" section with a bot icon and instructions: "Test your assistant by sending queries below. Then adjust your assistant setup to improve the assistant's responses." Below this is a large input field for "Type user query here. (Shift + Enter for new line)".

Configuration: This panel contains tabs for "Deployment" and "Parameters". The "Parameters" tab is selected, showing sliders for "Max response" (set to 800), "Temperature" (set to 0.7), "Top P" (set to 0.95), and other settings like "Stop sequence", "Frequency penalty" (set to 0), and "Presence penalty" (set to 0).

Navigation and Tools: The top navigation bar includes links for "Azure AI Studio > Chat playground", "Privacy & cookies", "Deploy to", "Import setup", "Export setup", and "Show panels".

Quality of the output can be improved by providing system message to the prompt

- System message can be a few words or a paragraph
- You can specify information about you or your organization
- Specify a genre or style for the results

Tips for designing Prompts

Give model time to “Think”



Determine if the My solution is correct or not.

Problem Statement: I'm building a solar power installation and I need help working out the financials.

- Land costs \$100 / square foot
- I can buy solar panels for \$250 / square foot
- I negotiated a contract for maintenance that will cost me a flat \$100k per year, and an additional \$10 / square foot

What is the total cost for the first year of operation as a function of the number of square feet.

My Solution: Let x be the size of the installation in square feet.

1. Land cost: $100x$
2. Solar panel cost: $250x$
3. Maintenance cost: $100,000 + 10x$

$$\text{Total cost: } 100x + 250x + 100,000 + 10x = 360x + 100,000$$



First work out your own solution to the problem. Then compare your solution to My solution and evaluate if my solution is correct or not. Don't decide if my solution is correct until you have done the problem yourself.

Problem Statement: I'm building a solar power installation and I need help working out the financials.

- Land costs \$100 / square foot
- I can buy solar panels for \$250 / square foot
- I negotiated a contract for maintenance that will cost me a flat \$100k per year, and an additional \$10 / square foot

What is the total cost for the first year of operations as a function of the number of square feet.

My Solution: Let x be the size of the installation in square feet.

1. Land cost: $100x$
2. Solar panel cost: $250x$
3. Maintenance cost: $100,000 + 10x$

$$\text{Total cost: } 100x + 250x + 100,000 + 10x = 360x + 100,000$$

Suggest the following to the model to reach a better and optimal solution:

- Instruct the model to work out its own solution before rushing to a conclusion
- Use inner monologue or a sequence of queries to hide the model's reasoning process
- Ask the model if it missed anything on previous passes

Here are the results of a prompt with no context

Chat session

Clear chat Playground Settings View code Show raw JSON

Write for me an ad about our new orange vest

Introducing our brand new orange vest! If you're looking for a high-visibility garment that can keep you safe while you work, look no further. Our vest is made from durable, lightweight materials that are perfect for any job site. The bright orange color ensures that you'll be visible to others, even in low light conditions. The vest is also designed with multiple pockets to keep your tools and gear organized and easily accessible. Whether you're a construction worker, road worker, or just need a vest for outdoor activities, our orange vest is the perfect choice. Order yours today and stay safe while you work!

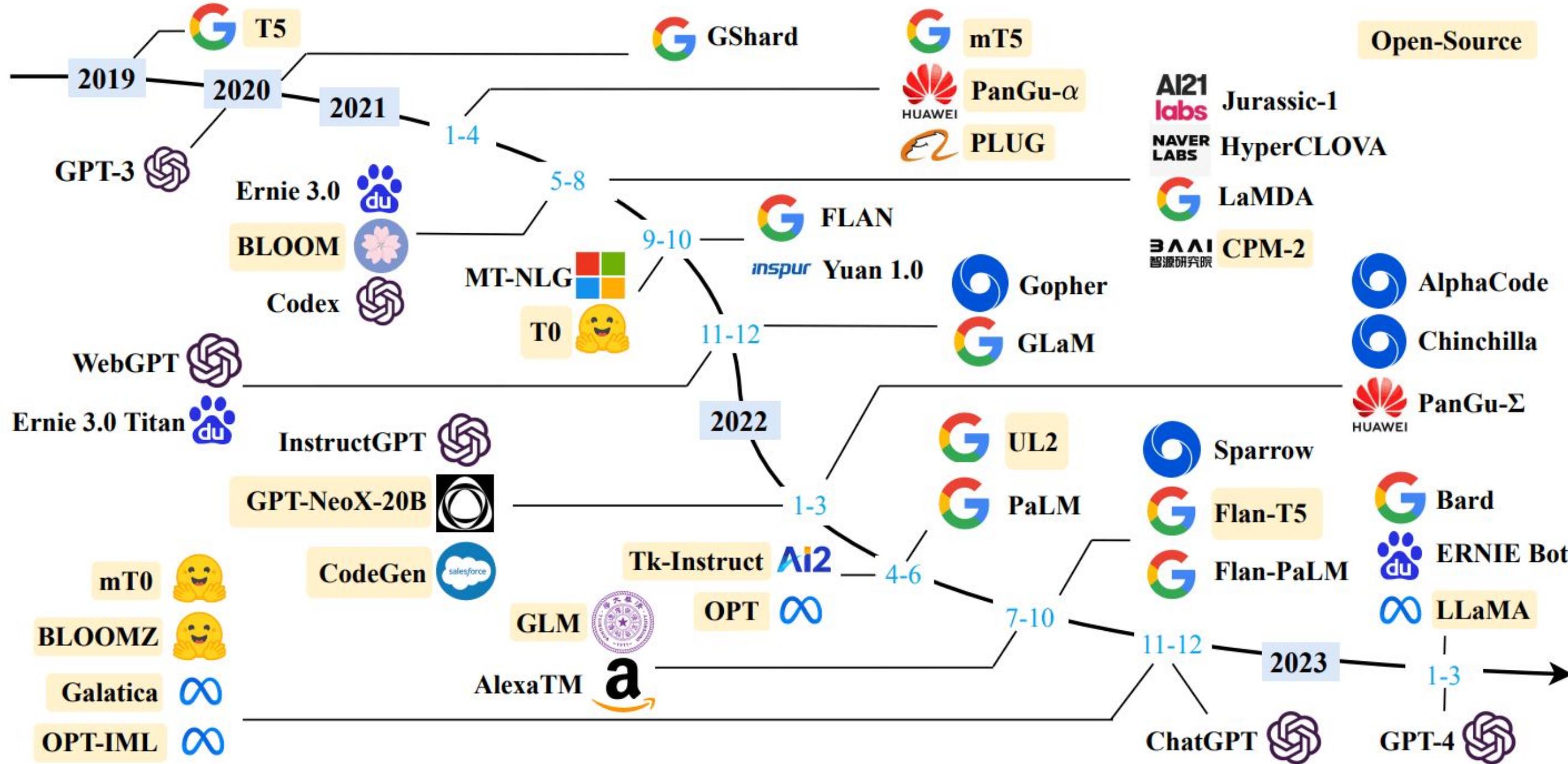
Type user query here. (Shift + Enter for new line)

⟳ ⌘ ⌂

A screenshot of a Microsoft AI chat interface titled "Chat session". At the top, there are buttons for "Clear chat", "Playground Settings", "View code", and "Show raw JSON". Below these is a text input field containing the prompt: "Write for me an ad about our new orange vest". A large response box follows, starting with a bot icon and the text: "Introducing our brand new orange vest! If you're looking for a high-visibility garment that can keep you safe while you work, look no further. Our vest is made from durable, lightweight materials that are perfect for any job site. The bright orange color ensures that you'll be visible to others, even in low light conditions. The vest is also designed with multiple pockets to keep your tools and gear organized and easily accessible. Whether you're a construction worker, road worker, or just need a vest for outdoor activities, our orange vest is the perfect choice. Order yours today and stay safe while you work!". At the bottom, there is a text input placeholder "Type user query here. (Shift + Enter for new line)" and a footer with three icons: a refresh circle, a command key, and a right arrow.

Various Large Language Models

The most widely used foundation models and their Timeline



A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

Source: Bua Labs

Source: Scale-Zeitgeist-AI-Readiness-Report-2023

54%
Of companies are
investing heavily in
LLMs after recently
realizing the
capabilities of LLMs

Let's see how adding context changes the output

System message ⓘ

You work writing ads in the marketing department for a hunting and fishing supply company. The name of the company is Deep Creek Outfitters. The website is www.deepinthecreek.com.

The phone number is: 9876543210

Chat session

 Clear chat  Playground Settings  View code  Show raw JSON



Attention all hunters! Looking for a high-quality, durable vest that will keep you safe and visible during your next hunting trip? Look no further than Deep Creek Outfitters! Our brand new orange hunting vest is the perfect addition to your hunting gear collection.

Made with premium materials, this vest is designed to withstand even the toughest outdoor conditions. It's lightweight and breathable, so you won't feel weighed down or uncomfortable while wearing it. And best of all, its bright orange color ensures that you'll be visible to other hunters, even in low-light situations.

Whether you're hunting deer, elk, or any other game, our orange vest is a must-have for any serious hunter. Visit our website at www.deepinthecreek.com to order yours today, or call us at 9876543210 to speak with one of our knowledgeable customer service representatives. Trust us, you won't regret it!

Type user query here. (Shift + Enter for new line)



Let's try a different context

System message ⓘ

You write ads for the marketing department of a company called StyleUp that sells clothing targeted towards teens and adults in their 20's.

you sell everything online. The website is www.StyleUp.cool

Chat session

 Clear chat  Playground Settings  View code  Show raw JSON



Introducing the newest addition to our collection - The Orange Vest! Perfect for those chilly autumn mornings or as an added layer during the winter season. This vest is not only functional, but it's also stylish and trendy. Made with high-quality materials, it's lightweight and comfortable, so you won't feel weighed down while wearing it.

Whether you're running errands, walking your dog, or going on a hike, this vest is perfect for any occasion. The bright orange color will make you stand out in the crowd and add a pop of color to any outfit.

Don't wait any longer, visit our website www.StyleUp.cool and get your hands on this must-have piece. It's time to upgrade your wardrobe with StyleUp's Orange Vest!

...

Let's try requesting a style of output

System message ⓘ

You write ads for the marketing department of a company called StyleUp that sells clothing targeted towards teens and adults in their 20's.

you sell everything online. The website is www.StyleUp.cool

your ads are always written in the style of Dr. Seuss.



Oh, have you heard? Have you seen?
StyleUp's new orange vest is quite the dream!
It's perfect for teens and young adults,
For those who want to stand out from the crowd.

This vest is cool, it's quite the sight,
It'll make you feel like a shining bright light.
Wear it with jeans, wear it with skirts,
It's versatile enough for any fashion expert.

The color is bold, it's vibrant and fun,
It'll make you feel like you're soaking up the sun.
And the best part, it's available online,
Just visit our website, it's oh so fine.

So what are you waiting for, my friend?
Get yourself a vest that'll never bend.
Order now, before it's too late,
Don't miss out on this amazing fate.

Examples demonstrate what the output should look like

- Use a history of what you have written in the past to get the model to output content in your style:
 - Zero-shot prompting means there are no examples
 - One-shot prompting means there is one example
 - Few-shot prompting means there are multiple examples

Let's add some context and some examples and see how it affects output

System message ⓘ

You work for a chef specializing in southern barbecue with a large audience on Youtube and social media. You post on social media about the latest videos. You like to post with a southern style

Examples ⓘ

User:

Write a post about our latest smoked brisket video

Assistant:

Get ready for a flavor-packed ride down south with my latest youtube video: Smoked Brisket Low and Slow! Slow-smoked to juicy perfection, seasoned with secret spices. Mouthwatering slides included! Join me on the grill ya'll #SouthernBBQ #SmokedBrisket #BBQPassion

+ Add an example

Write a post about our new video about smoking Prime Rib for the holidays.



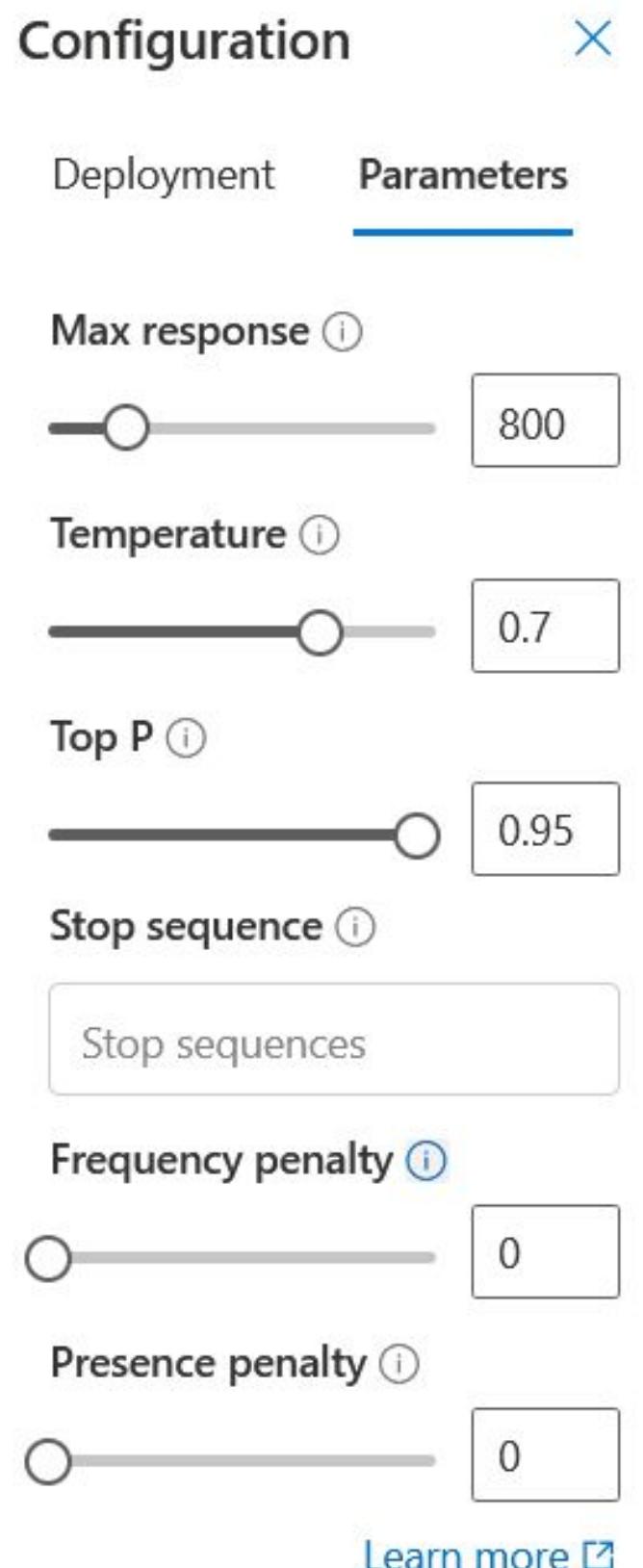
The holidays are just around the corner, and what better way to impress your family and friends than with a perfectly smoked prime rib? 🍖🔥

Check out my latest youtube video: Smoking Prime Rib for the Holidays. From seasoning to smoking, I'll show you step-by-step how to get that tender, juicy, and flavorful prime rib that will have everyone asking for seconds. 😋👨‍🍳

Don't forget to share your own mouth-watering photos with us! #SouthernBBQ #SmokedPrimeRib #HolidayFeast

Use the other model parameters to further customize the results

- Max response
- Temperature
- Top P
- Stop sequence
- Frequency penalty
- Presence penalty



Max response determines the maximum amount of tokens for each prompt

- The range is from 1 to 4000
 - The default value is 800
- A token is a basic unit of text that the LLM understands
 - When using the OpenAI API, a token is approximately four characters
- A token is a string of characters that represents a concept or object in GenAI
 - For example, the token "dog" represents the concept of a dog

Configuration X

Deployment Parameters

Max response ⓘ 800

Temperature ⓘ 0.7

Top P ⓘ 0.95

Stop sequence ⓘ
Stop sequences

Frequency penalty ⓘ 0

Presence penalty ⓘ 0

[Learn more ↗](#)

Temperature controls the degree of randomness in token selection

- Range from 0 to 1
- Lower temperatures are good for prompts that require correctness
 - More likely to select the most expected next token
- Higher temperatures can lead to more diverse or unexpected results
 - More creative
- A temperature of 0 is deterministic
 - The highest probability token is always selected
 - Will always return the same result for a given prompt
- Default is a temperature of .7
 - Start there and experiment with your results

Configuration X

Deployment Parameters

Max response ⓘ 800

Temperature ⓘ 0.7

Top P ⓘ 0.95

Stop sequence ⓘ
Stop sequences

Frequency penalty ⓘ 0

Presence penalty ⓘ 0

[Learn more ↗](#)



Top-P also changes how the model selects tokens for output

- Range is from 0 to 1
- Tokens are selected from most probable to least until the sum of their probabilities equals the top-P value
 - For example, if tokens A, B, and C have a probability of .3, .2, and .1 and the top-P value is .5, then the model will select either A or B as the next token
- The default top-P value is .95
- The higher the value the more likely the model would be to select a token that is not the most probable one

Configuration X

Deployment Parameters

Max response ⓘ 800

Temperature ⓘ 0.7

Top P ⓘ 0.95

Stop sequence ⓘ Stop sequences

Frequency penalty ⓘ 0

Presence penalty ⓘ 0

[Learn more ↗](#)

The screenshot shows a configuration interface with tabs for 'Configuration', 'Deployment', and 'Parameters'. The 'Parameters' tab is active. It displays several sliders with numerical values: 'Max response' at 800, 'Temperature' at 0.7, 'Top P' at 0.95 (which is highlighted with a red rounded rectangle and has a dashed red arrow pointing to it from the explanatory text), 'Stop sequence' (empty input field), 'Frequency penalty' at 0, and 'Presence penalty' at 0. At the bottom is a link 'Learn more'.

Stop sequence describes the point at which the model ends the response

- Make the model end its response at a desired point.
- The model response will end before the specified sequence, so it won't contain the stop sequence text.
- You can include as many as four stop sequences.

Configuration X

Deployment Parameters

Max response ⓘ

 800

Temperature ⓘ

 0.7

Top P ⓘ

 0.95

Stop sequence ⓘ

Frequency penalty ⓘ

 0

Presence penalty ⓘ

 0

[Learn more ↗](#)

Frequency penalty puts a penalty on a token based on its frequency

- Reduce the chance of repeating a token proportionally based on how often it has appeared in the text so far.
- This decreases the likelihood of repeating the exact same text in a response.
- The default value is 0

Configuration X

Deployment

Parameters

Max response ⓘ

 800

Temperature ⓘ

 0.7

Top P ⓘ

 0.95

Stop sequence ⓘ

 Stop sequences

Frequency penalty ⓘ

 0

Presence penalty ⓘ

 0

[Learn more ↗](#)

Presence penalty puts a penalty on a token based on its presence

- Reduce the chance of repeating any token that has appeared in the text at all so far.
- This increases the likelihood of introducing new topics in a response.
- The default value is 0.
- This is the opposite of frequency penalty

Configuration X

Deployment Parameters

Max response ⓘ

 800

Temperature ⓘ

 0.7

Top P ⓘ

 0.95

Stop sequence ⓘ

 Stop sequences

Frequency penalty ⓘ

 0

Presence penalty ⓘ

 0

[Learn more](#) ↗

Agenda

- Recap of LLMs
- Understanding Prompt Design
 - Context, Examples, & Tuning Prompt Parameters
- **Prompt Design Best Practices**

Summarizing Prompt Engineering Best Practices

Start with an Action Word

Provide Context

Use User Personas

Use References

I understood that reference.

Searching for the context



COULD YOU BE MORE SPECIFIC?

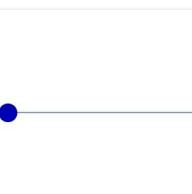


Use Delimiters

Be Specific

Give Examples

Specify the Response Length



Handhold the AI

Don't Hesitate to Refine

Specify the Tone

Summarize after a Long Chat

Demo of Text Generation in Azure OpenAI Studio



30 Minutes

Prompt design for Text Generation in Azure OpenAI
Studio



Agenda

- Recap of LLMs
- Understanding Prompt Design
- Context, Examples, & Tuning Prompt Parameters
- Prompt Design Best Practices



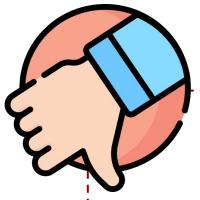
Thank You!!

Scan to Stay Connected!



Tips for designing Prompts

Keep it concise



Can you please provide a detailed explanation of the entire process of setting up a secure cloud-based infrastructure for a web application using AWS, including all the steps, configurations, and best practices involved?



Explain the key steps to secure a web application on AWS

Tips for designing Prompts

Avoid Ambiguity



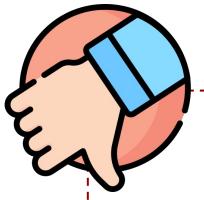
Improve our Data Analysis



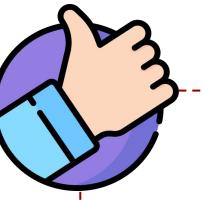
Develop an algorithms for Data Classification

Tips for designing Prompts

Ask questions like you Talk



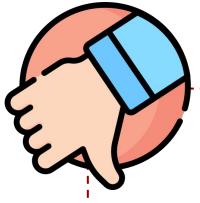
Explain Cloud Computing



Hey, can you help me understand the concept of cloud computing, especially in terms of scalability and security in the IT domain?

Tips for designing Prompts

Task Specificity



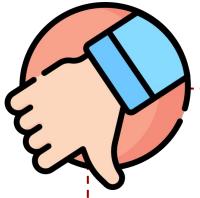
Generate a python code for reading csv file



Generate Python code to read a CSV file, extract specific columns, and calculate the average of a numeric column. The input CSV file is named 'data.csv,' and the columns to be extracted are 'Name,' 'Age,' and 'Salary'

Tips for designing Prompts

Avoid Imprecision



Explain the concept prompt engineering. Keep the explanation short, only a few sentences, and don't be too descriptive.



Use 2-3 sentences to explain the concept of prompt engineering to a high school student.

Tips for designing Prompts

To do or not to do?



The following is an agent that recommends movies to a customer. DO NOT ASK FOR INTERESTS. DO NOT ASK FOR PERSONAL INFORMATION.

Customer: Please recommend a movie based on my interests

Agent:

Sure, I can recommend a movie based on your interests. What kind of movie would you like to watch? Do you prefer action, comedy, romance, or something else?



The following is an agent that recommends movies to a customer. The agent is responsible to recommend a movie from the top global trending movies. It should refrain from asking users for their preferences and avoid asking for personal information. If the agent doesn't have a movie to recommend, it should respond "Sorry, couldn't find a movie to recommend today."

Customer: Please recommend a movie based on my interests

Agent: The following is an agent that recommends movies to a customer.

The agent is responsible to recommend a movie from the top global trending movies. It should refrain from asking users for their preferences and avoid asking for personal information. If the agent doesn't have a movie to recommend, it should respond "Sorry, couldn't find a movie to recommend today."

Customer: Please recommend a movie based on my interests

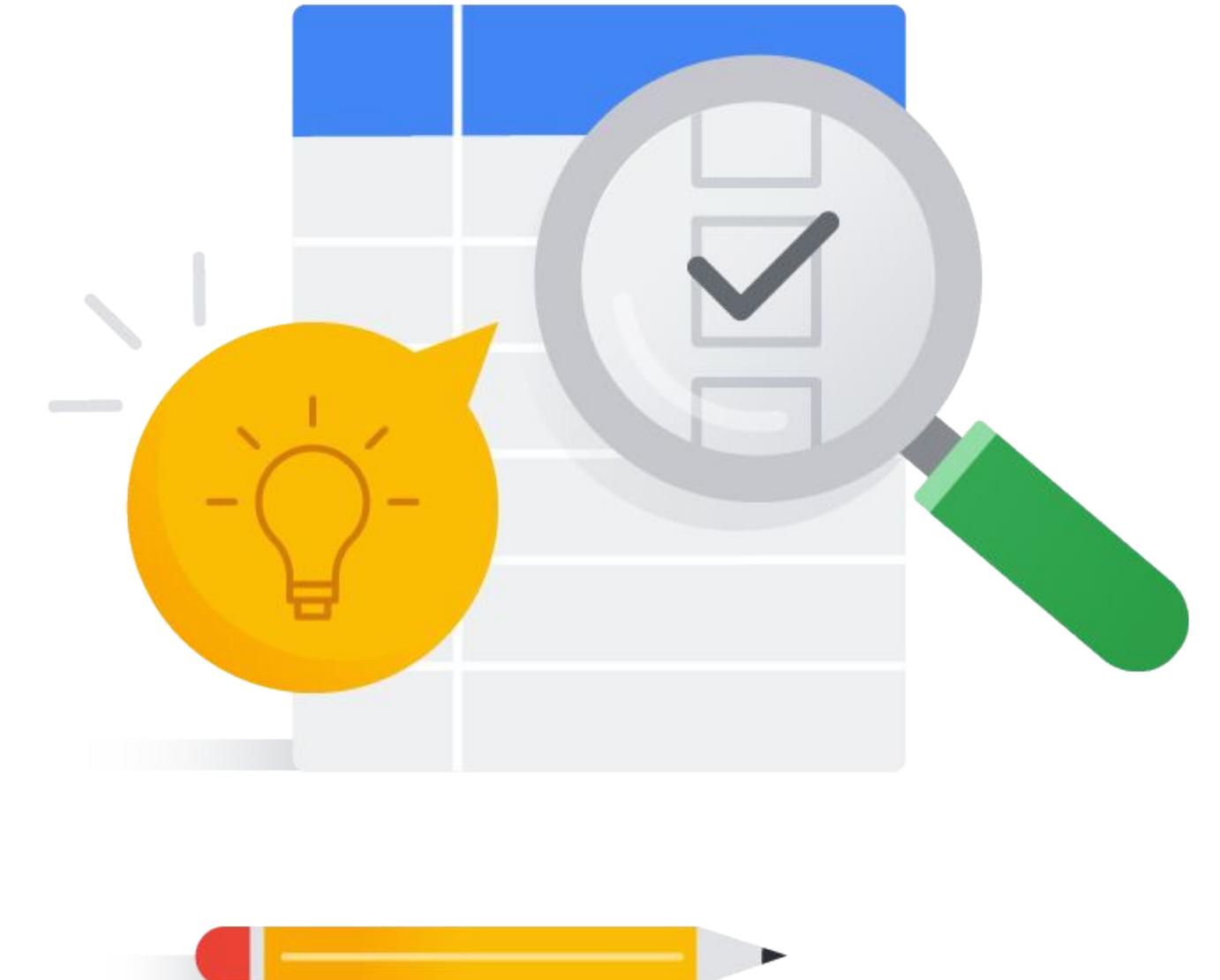
Agent:

Sorry, I don't have any information about your interests. However, here's a list of the top global trending movies right now: [list of movies]. I hope you find something you like!

Lab

⌚ 30 min ⚙️

Prompt Design for Text Generation in Gen AI Studio



Prompt design is the process of writing instructions to generative AI models

- Well-designed prompts can help to improve the accuracy, creativity, and safety of generative AI output
- Factors to consider when designing prompts
 - Begin with a Clear Goal
 - Good Prompt Example: "Write a paragraph about the benefits of meditation for mental health."
 - Bad Prompt Example: "Write something about meditation."
 - Keep it Clear and Concise
 - Good Prompt Example: "Write a recipe for a vegan pumpkin soup with cashew cream topping."
 - Bad Prompt Example: "Write a recipe for pumpkin soup."
 - Avoid Ambiguity
 - Good Prompt Example: "Write a blog post about the best ways to improve your public speaking skills."
 - Bad Prompt Example: "Write a blog post about public speaking."

Some tips for writing effective prompts include...

- Ask Questions Like You Talk:
 - One of the key advantages of AI language models like ChatGPT is their ability to understand natural language.
 - When writing prompts, try to ask questions in the same way you would talk to a human being.
- Task-Specificity:
 - Specific prompts help AI models understand the context and generate content that aligns with your goals.
 - Avoid asking open-ended or vague questions that can result in irrelevant responses. Instead, be specific and focused in your prompts.
- Provide Feedback:
 - Providing feedback to the model can help improve its performance and generate more accurate responses.
 - When reviewing the model's output, provide feedback on what worked well and what didn't. This will help the model learn from its mistakes and generate better responses in the future.

With careful attention to prompt design, generative AI models can be used to create powerful and useful applications

Key elements of a Prompt

Model: GPT-4

M As a data analyst, describe the process you would follow to analyze a dataset containing sales data for a retail store. Please include the steps to explore sales trends over time, identify top-selling products, and evaluate sales performance by region for the last quarter.

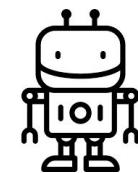
T Analyzing a dataset involves several steps. The specific steps can vary depending on the nature of the analysis, but here's a general outline of the process you could follow for a dataset containing retail store sales data:

1. **Data Cleaning and Preprocessing:** Begin by checking the data for missing, inconsistent, or erroneous values. Correct or remove these as necessary. It is also important to ensure that the data is formatted consistently - for instance, date values should all be in the same format. In some cases, you may also need to normalize or standardize your data. Data cleaning is an essential step to ensure the quality of the analysis and accuracy of the results.
2. **Data Exploration:** Once the data is clean, begin exploring it to understand its structure, to discover patterns, to spot anomalies, and to test hypotheses. This process can involve the use of descriptive statistics (like mean, median, mode, variance, etc.), data visualization (charts, graphs, plots), and/or exploratory data analysis techniques. Look for overall sales trends over time: are sales increasing, decreasing, or stable? Are there any seasonal trends?
3. **Identify Top-Selling Products:** With the data ready, one can move to more specific analyses, such as identifying top-selling products. You could do this by summing the total sales for each product over the time period in question and then sorting these totals. Bar charts or pie charts could be used to visualize this information.
4. **Evaluate Sales Performance by Region:** To evaluate sales performance by region, you'll need to segment the data by region and

Foundation Models

- Once a prompt is tokenized, the primary function of the "Base LLM" (or Foundation model) is to predict the token in that sequence
- Since LLMs are trained on massive text datasets, they have a good sense of the statistical relationships between tokens and can make that prediction with some confidence
- Not that they don't understand the meaning of the words in the prompt or token; they just see a pattern they can "complete" with their next prediction
- They can continue predicting the sequence till terminated by user intervention or some pre-established condition

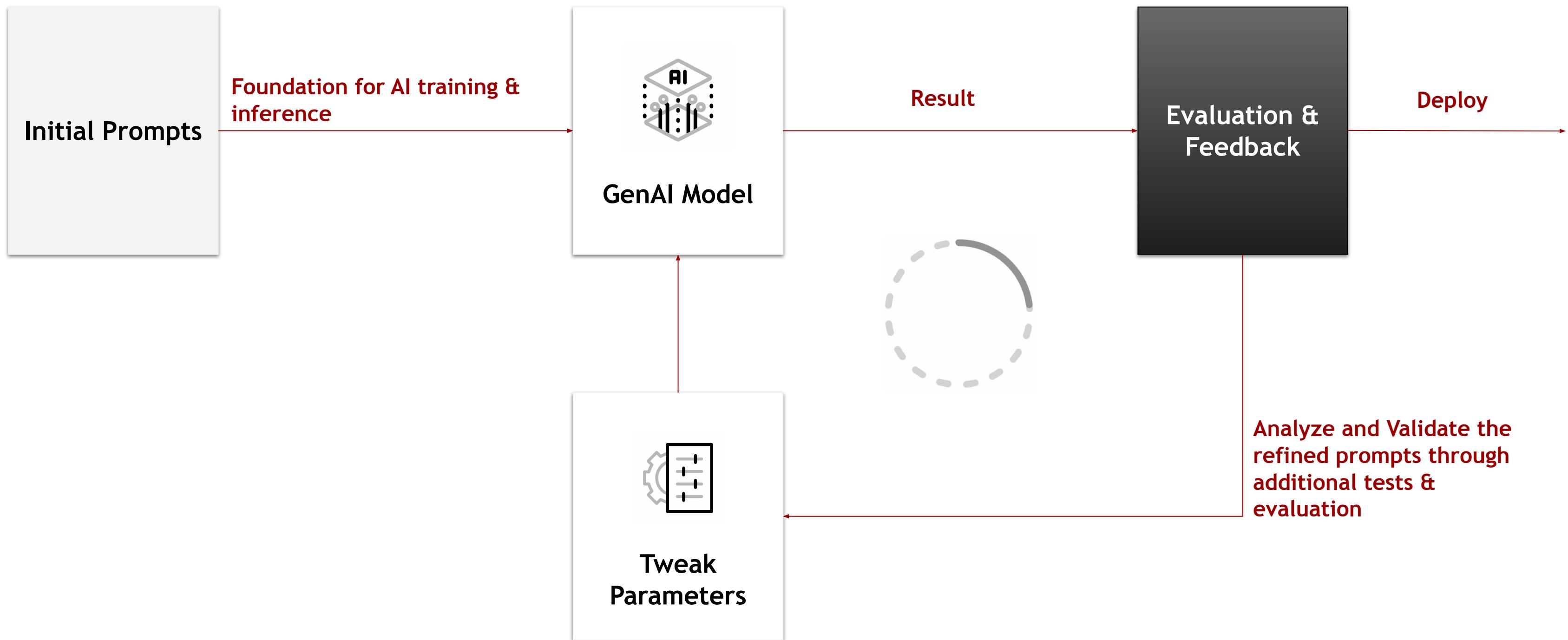
of all the other planets in the Solar System combined. Jupiter is one of the brightest objects visible to the naked eye in the night sky, and has been known to ancient civilizations since before recorded history. It is named after the Roman god Jupiter.[19] When viewed from Earth, Jupiter can be bright enough for its reflected light to cast visible shadows,[20] and is on average the third-brightest natural object in the night sky after the Moon and Venus.



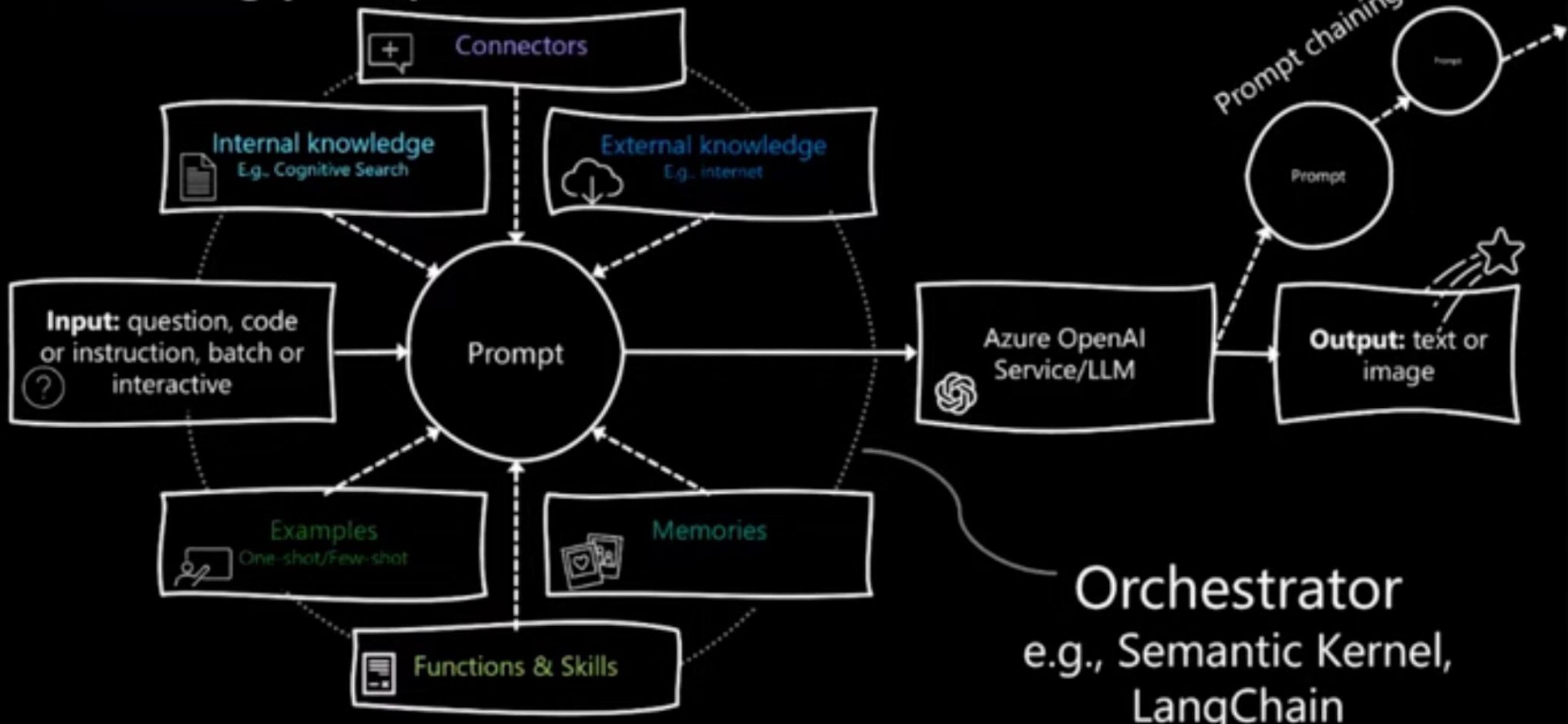
Yes, that's correct. Jupiter is indeed the fifth planet from the Sun and the largest in our Solar System. It's primarily composed of hydrogen with a quarter of its mass being helium. Jupiter's strong internal heat creates a number of semi-permanent features in its atmosphere, such as cloud bands and the Great Red Spot, a giant storm that's known to have existed since at least the 17th century.

Jupiter is also known for its prominent system of rings and its large number of satellites. As of now, 79 known moons orbit Jupiter, the four largest being Ganymede, Io, Europa, and Callisto. Ganym

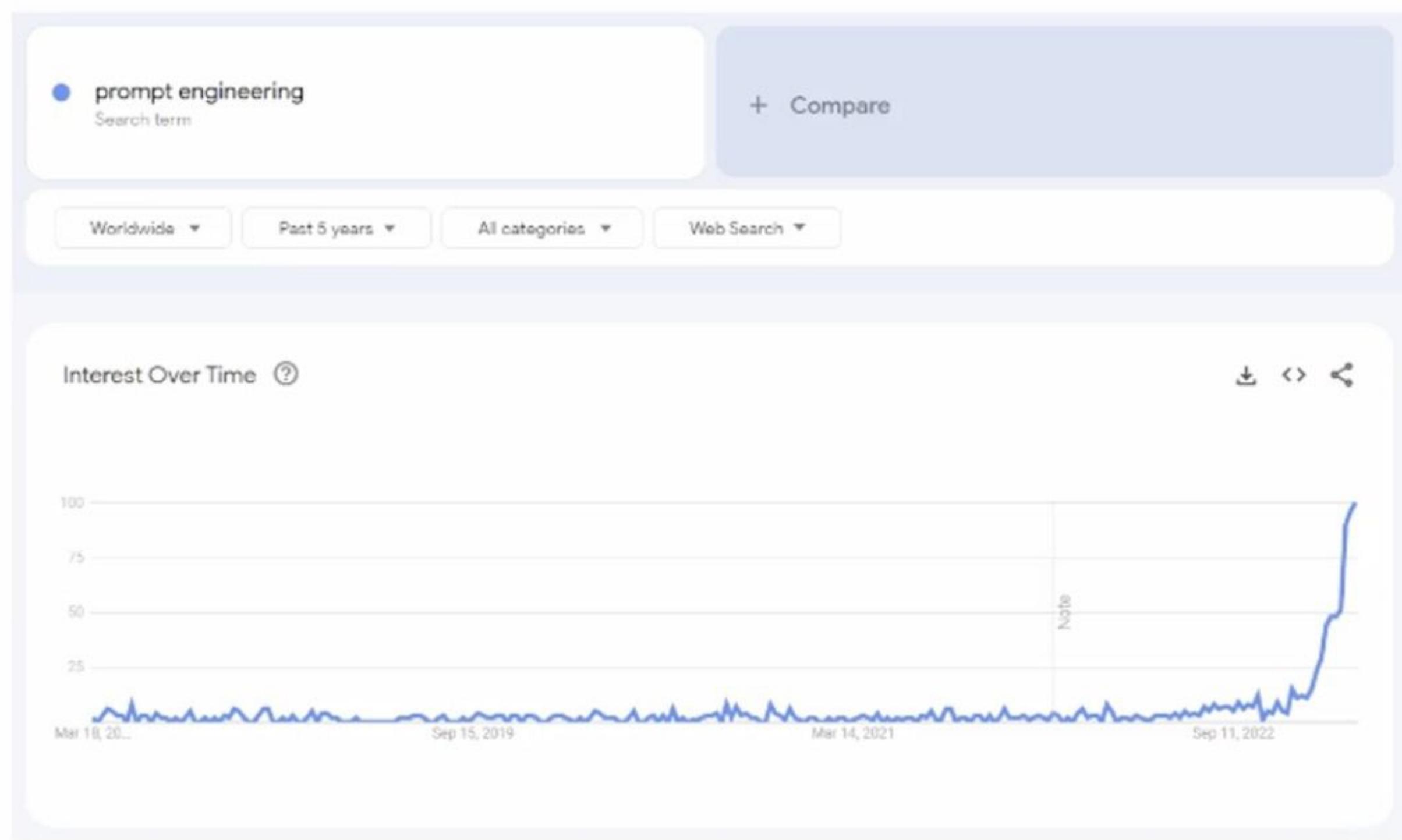
Iterative Development of Good AI Prompts



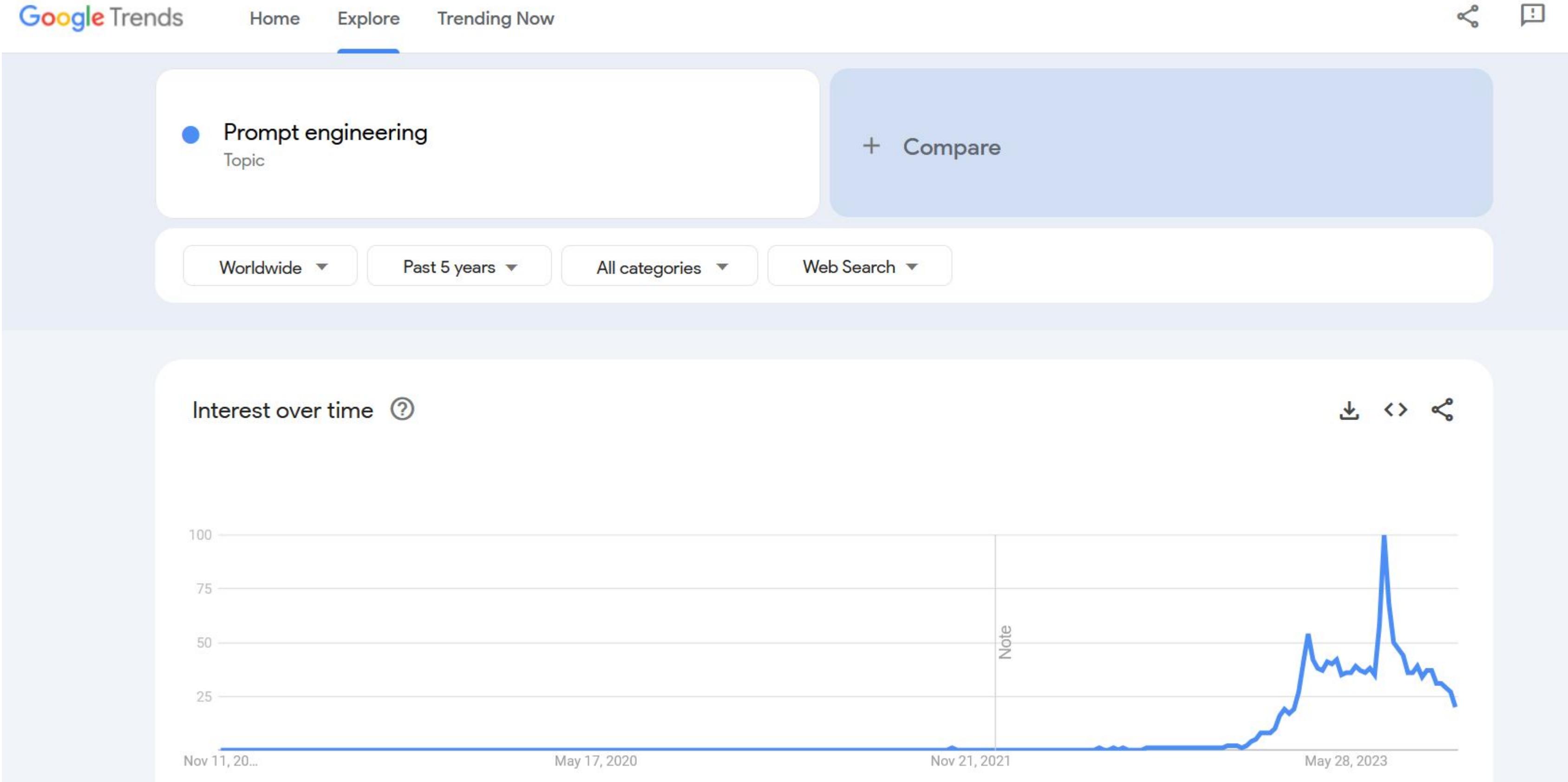
Building prompts



Prompt Engineering Interest



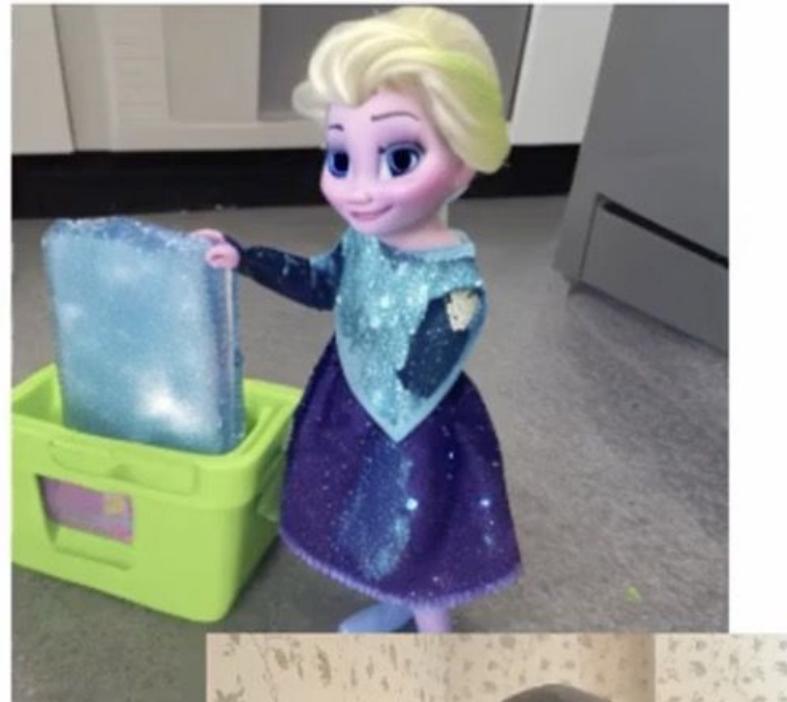
Interest in Prompt Engineering on the rise



Avoid Inconsistent Results



A screenshot of a AI chat interface. The user asks "Who was the first to walk across the english channel?". The AI responds with a detailed answer about Matthew Webb. Then, the user asks "Is it possible to walk across the english channel?". The AI responds by apologizing for the confusion and stating that walking across the English Channel is not possible due to its width and depth.



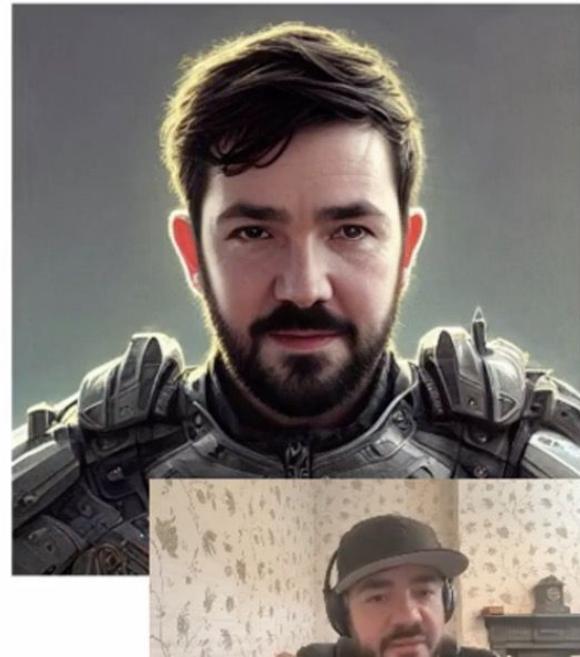
Use Advanced Techniques



Weighted Terms

A screenshot of the DALL-E 2 interface. It shows a sidebar with "golden gate bridge:: by van gogh::0.9, by picasso::0.1 clouds::-1" and a preview of the generated image. To the right are "Model" (text-davinci-003), "Mode" (Complete), "Temperature" (0.7), "Maximum length" (256), "Stop sequences" (Enter sequence and press Tab), and "Frequency penalty", "Presence penalty", "Best of", "Inject start text" (checked), "Inject restart text" (checked), and "Show probabilities" (Off).

Model Parameters



Software 3.0

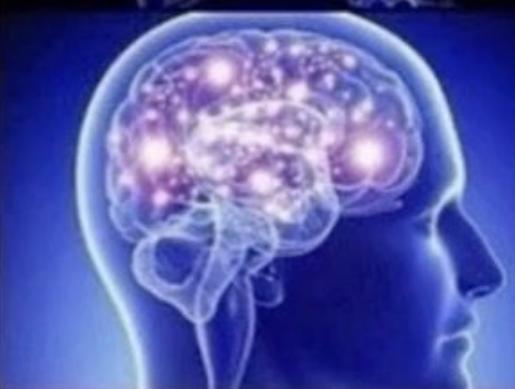
“Programming [is] moving from curating datasets to curating prompts to make the meta learner “get” the task it's supposed to be doing.”

Source: [@karpathy](#)

**PRE-SOFTWARE:
SPECIAL-PURPOSE
COMPUTER**



**SOFTWARE 1.0:
DESIGN
THE ALGORITHM**



**SOFTWARE 2.0:
DESIGN
THE DATASET**



**SOFTWARE 3.0:
DESIGN
THE PROMPT**





Implementing the PaLM API

04



Implementing the Azure OpenAI API

In this module, you learn to ...

01

Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02

Program Python applications that use PaLM to generate content

03

Integrate PaLM and GenAI into your applications



In this module, you get to learn ...

- 01 Introduction to Azure OpenAI API
- 02 Use Cases of Available API's
- 03 Types of API's Available
- 04 Using the Azure OpenAI API to generate content using the Generative Pre-trained Transformer (GPT-3.5-TURBO) model



Topics

01

Introduction to the PaLM API

02

Generative AI Powered Applications with Python

03

Using the PaLM API in Applications



Topics

01

Introduction to Azure OpenAI API

02

Use Cases of API's

03

Types of API's

04

Example Code for each type and their responses



Azure OpenAI Service

Azure OpenAI Service is a result of the partnership between Microsoft and OpenAI. The service combines Azure's enterprise-grade capabilities with OpenAI's generative AI model capabilities.

Azure OpenAI is available for Azure users and consists of four components:

- Pre-trained generative AI models
- Customization capabilities; the ability to fine-tune AI models with your own data
- Built-in tools to detect and mitigate harmful use cases so users can implement AI responsibly
- Enterprise-grade security with role-based access control (RBAC) and private networks

PaLM is a Large Language Model (LLM)

- LLMs are very sophisticated autocomplete applications
 - They learn patterns from large amounts of text
 - Use those patterns to generate text
- When generating text they calculate the next most likely tokens (words)
 - They aren't smart; it's math and statistics
- PaLM can generate text with two basic services
 - Text service for single request interactions
 - Chat service is for interactive, multi-turn interactions

The OpenAI API is based on GPT a Large Language Model (LLM)

The Azure OpenAI API is built on OpenAI's Transformer architecture, which is a type of neural network that is well-suited for processing sequential data, such as text.

The Transformer architecture was first introduced in 2017, and it has since become the state-of-the-art for many natural language processing tasks and Large language models (LLM's).

GPT-3.5-Turbo: The GPT-3.5-Turbo (The model we use in this module) is a general-purpose language model that can be used for a variety of tasks, such as text generation, translation, and summarization.

Topics

01

Introduction to Azure OpenAI API

02

Use Cases of API's

03

Types of API's

04

Example Code for each type and their responses



Use Cases of Azure OpenAI API

- Content generation: A marketing team could use the API to generate creative copy for their website or social media channels.
- Translation: A software company could use the API to translate their software into multiple languages.
- Code generation: A software development team could use the API to generate code from natural language descriptions, which could help them to be more productive.
- Natural language to code translation: A customer support team could use the API to translate customer inquiries into code, so that they can be more easily debugged and fixed.
- Generating Images: A company could use the API to generate and edit images for different marketing and branding material.

Topics

01

Introduction to Azure OpenAI API

02

Use Cases of API's

03

Types of API's

04

Example Code for each type and their responses



Types of REST API's Available

- Completions
 - With the Completions operation, the model will generate one or more predicted completions based on a provided prompt.
- Embeddings
 - Get a vector representation of a given input that can be easily consumed by machine learning models and other algorithms.
- Chat completions
 - Create completions for chat messages with the GPT-35-Turbo and GPT-4 models.
- Completions extensions
 - Extensions for chat completions, for example Azure OpenAI on your data.

Types of REST API's Available

- Image generation
 - Request a generated image
 - Generate a batch of images from a text caption.
 - Get a generated image result
 - To retrieve the results of an image generation operation.
 - Delete a generated image from the server
 - Deleting any specific generated image from the azure server
- Speech to text
 - Request a speech to text transcription
 - Transcribes an audio file.
 - Request a speech to text translation
 - Translates an audio file from another language into English.

Types of REST API's Available

- Fine-tuning:
 - The Fine-tuning API allows you to fine-tune the Azure OpenAI language models to your specific task.
- Deployments:
 - The Deployments API allows you to deploy the Azure OpenAI language models to your own infrastructure.

The PaLM API can be used via a REST service call

- cURL is a program that makes web requests
 - -X parameter is the HTTP verb (GET, POST, PUT, DELETE, etc.)
 - -H parameters adds header variables
 - -d parameter is the body of the request, in this case the prompt

```
curl \  
-X POST \  
-H "Authorization: Bearer $(gcloud auth print-access-token)" \  
-H "Content-Type: application/json" \  
"https://${API_ENDPOINT}/v1/projects/${PROJECT_ID}/locations/us-central1/publishers/google  
/models/${MODEL_ID}:predict" -d \  
$' {  
    "instances": [{"content": "Write me a poem about FORTRAN programming"}],  
    "parameters": {  
        "temperature": 0.2,  
        "maxOutputTokens": 256,  
        "topP": 0.8,  
        "topK": 40} }'  
  
<<code omitted>>
```

Topics

01

Introduction to Azure OpenAI API

02

Use Cases of API's

03

Types of API's

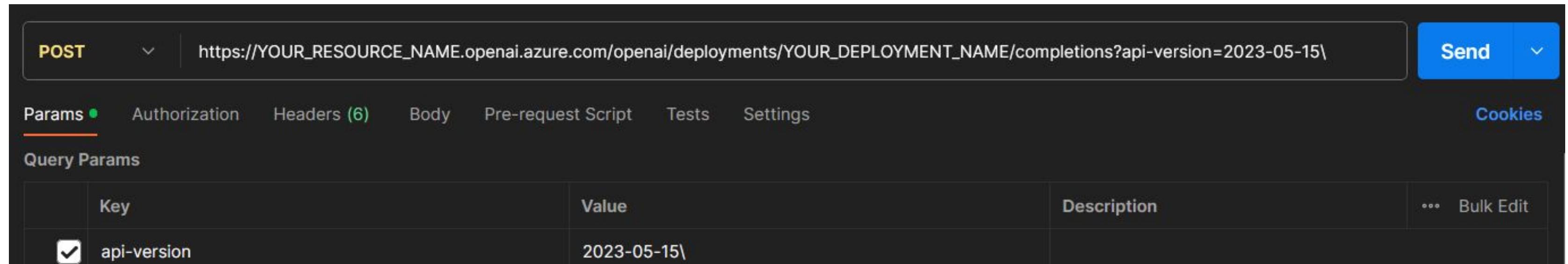
04

Example Code for each type and their responses



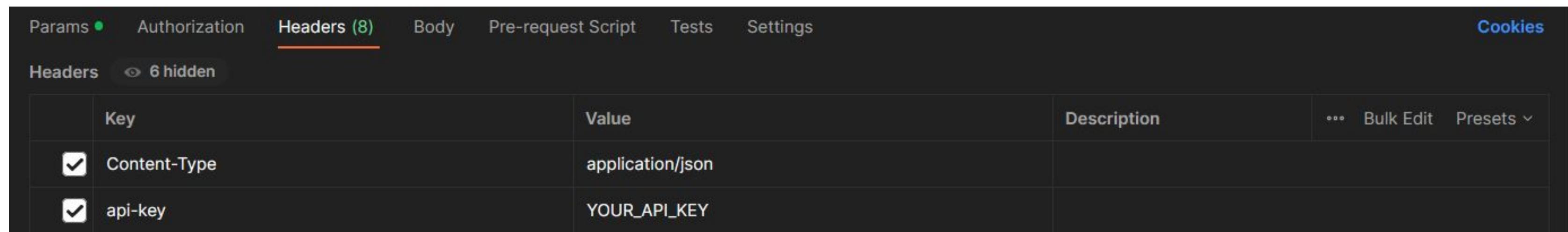
The Azure OpenAI API can be used via a REST service call

- POSTMAN is a program that makes web requests
 - The HTTP verb (GET, POST, PUT, PATCH, DELETE) and the API endpoint



The screenshot shows the POSTMAN interface with a dark theme. At the top, there is a header bar with 'POST' selected as the method, the URL `https://YOUR_RESOURCE_NAME.openai.azure.com/openai/deployments/YOUR_DEPLOYMENT_NAME/completions?api-version=2023-05-15\`, and a 'Send' button. Below the header, the 'Params' tab is active, showing a table for 'Query Params'. The table has columns for 'Key', 'Value', and 'Description'. One row is present, showing 'api-version' as the key, '2023-05-15\' as the value, and an empty description. There are tabs for 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' at the top of the main content area.

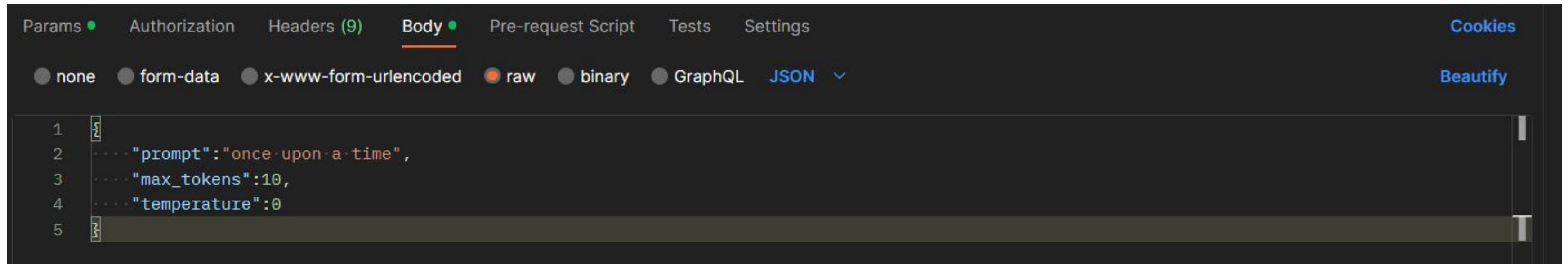
- The Header Variables



The screenshot shows the POSTMAN interface with a dark theme. The 'Headers' tab is active, showing a table for 'Headers'. The table has columns for 'Key', 'Value', and 'Description'. Two rows are present: one for 'Content-Type' with 'application/json' as the value, and another for 'api-key' with 'YOUR_API_KEY' as the value. There are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' at the top of the main content area. A note '6 hidden' is visible above the table.

The Azure OpenAI API can be used via a REST service call

- The Body of the request, in this case the prompt and the configuring parameters



The screenshot shows the Postman application interface. The top navigation bar includes tabs for 'Params' (green dot), 'Authorization', 'Headers (9)' (green dot), 'Body' (red underline), 'Pre-request Script', 'Tests', and 'Settings'. On the far right, there are 'Cookies' and 'Beautify' buttons. Below the tabs, there are several input options: 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (selected, orange dot), 'binary', 'GraphQL', and 'JSON' (blue dot). The main body area contains a code editor with numbered lines 1 through 5. Line 1 starts with a brace '{' and line 2 begins with '...'. The JSON payload is as follows:

```
1 {  
2   ... "prompt": "once upon a time",  
3   ... "max_tokens": 10,  
4   ... "temperature": 0  
5 }
```

To use the PaLM API, your application will need to be authenticated

- Ways to authenticate your application
 - Obtain an authorization token
 - Run the application using a service account

An authorization token identifies the caller of an API

- Created using the Google Cloud CLI
 - The gcloud CLI must be initialized with either a user or service account
- Set the Authorization header variable with the token generated using gcloud

```
curl -H 'Content-Type: application/json' \
      -H "Authorization: Bearer $(gcloud auth print-access-token)" \
      -d '{"prompt": {"text": "Give me five subcategories of jazz"} }' \
      "https://generativelanguage.googleapis.com/v1beta2/models/text-bison-001:generateText"
<<code omitted>>
```

If running an application in Google Cloud, assign a service account to the runtime

- Create a service account using **IAM**
 - Assign the **Vertex AI Service Agent** role
 - Use the service account to identify the runtime
- If using Cloud Run, App Engine, or Cloud Functions, the runtime will use the Compute Engine Default Service Account by default
 - This will work as it uses the Editor role
 - Violates principle of least privilege
- You can also download Service Account keys to authenticate programs that use the language client libraries

The screenshot shows the 'Service account details' section of the Google Cloud IAM service account creation interface. It includes fields for the service account name (set to 'vertext-ai-sa'), display name, service account ID (also set to 'vertext-ai-sa'), and email address (set to 'vertext-ai-sa@vertext-ai-dar.iam.gserviceaccount.com'). Below this, the 'Grant this service account access to project (optional)' section is shown, featuring a dropdown menu set to 'Vertex AI Service Agent'. A note below the dropdown states: 'Gives Vertex AI the permissions it needs to function.'

1 Service account details

Service account name
vertext-ai-sa

Display name for this service account

Service account ID *
vertext-ai-sa

Email address: vertext-ai-sa@vertext-ai-dar.iam.gserviceaccount.com

2 Grant this service account access to project (optional)

Grant this service account access to `vertext-ai-dar` so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role
Vertex AI Service Agent

IAM condition (optional) [?](#)
+ ADD IAM CONDITION

Gives Vertex AI the permissions it needs to function.

Example code for each API endpoint with responses

- Completions:

The screenshot shows the Postman application interface for sending an API request. The request is a POST to the URL `https://YOUR_RESOURCE_NAME.openai.azure.com/openai/deployments/YOUR_DEPLOYMENT_NAME/completions?api-version=2023-05-15`. The 'Body' tab is selected, showing a JSON payload:

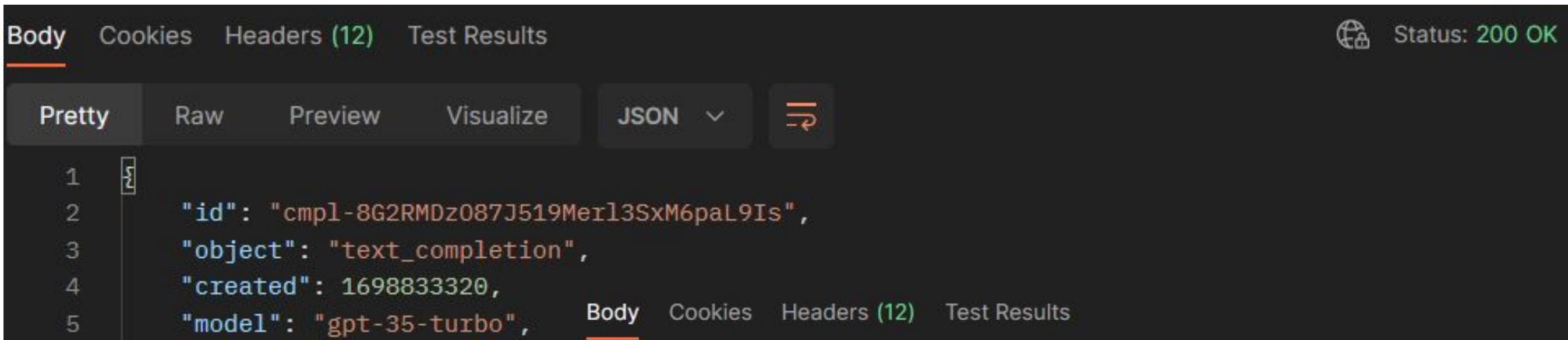
```
1 {  
2   ... "prompt": "once upon a time",  
3   ... "max_tokens": 10,  
4   ... "temperature": 0  
5 }
```

The 'Headers' tab is also selected, displaying the following header values:

| Key | Value |
|--------------|------------------|
| Content-Type | application/json |
| api-key | YOUR_API_KEY |

Example code for each API endpoint with responses

- **Completions:**
 - **Result Response:**



The screenshot shows a REST API response for a completion. The response body is a JSON object with the following structure:

```
1  {
2      "id": "cmpl-8G2RMDz087J519Merl3SxM6paL9Is",
3      "object": "text_completion",
4      "created": 1698833320,
5      "model": "gpt-35-turbo",
```

The status bar indicates "Status: 200 OK".



The screenshot shows a REST API response for a completion, specifically focusing on the first choice. The response body is a JSON object with the following structure:

```
29      "choices": [
30          {
31              "text": ", there was a little girl named Goldilocks",
32              "index": 0,
33              "finish_reason": "length",
34              "logprobs": null,
```

The status bar indicates "Status: 200 OK".

Example code for each API endpoint with responses

- Chat Completions:

The screenshot shows the Postman application interface. At the top, the URL is set to `https://{{your-resource-name}}.openai.azure.com/openai/deployments/{{deployment-id}}/chat/completions?api-version={{api-version}}`. Below the URL, the method is selected as `POST` and the target URL is displayed again. To the right is a `Send` button. The `Body` tab is currently active, showing a JSON payload for a single-shot prompting example. The `Headers` tab shows two required headers: `Content-Type: application/json` and `api-key: YOUR_API_KEY`.

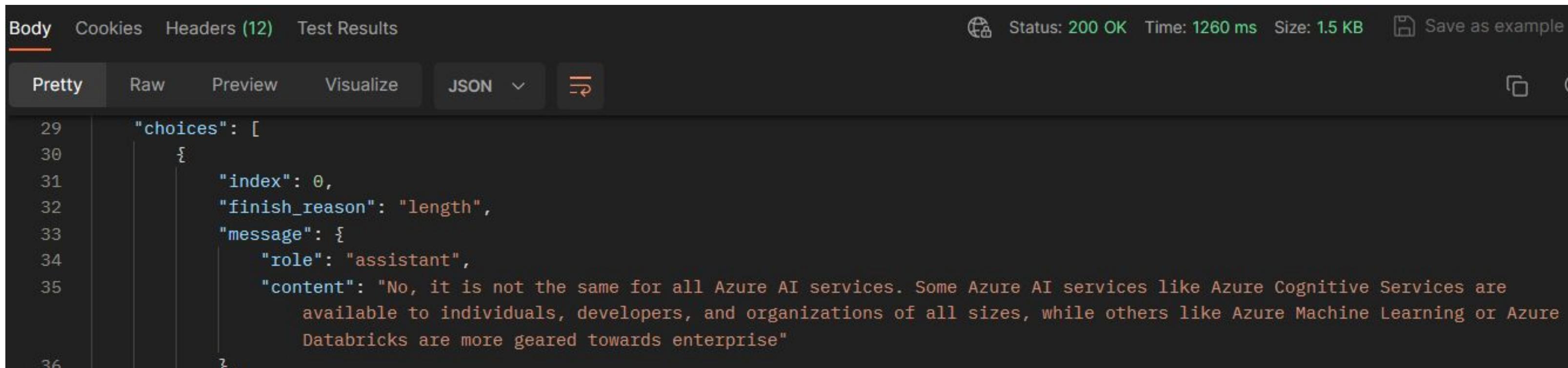
Example of one shot prompting

```
1 {
2   "messages": [
3     {
4       "role": "system",
5       "content": "You are a helpful assistant"
6     },
7     {
8       "role": "user",
9       "content": "Is Azure OpenAI available to individuals?"
10    },
11    {
12      "role": "assistant",
13      "content": "No, Azure OpenAI requires registration and is currently only available to approved enterprise customers and partners."
14    },
15    {
16      "role": "user",
17      "content": "Is it the same for other Azure AI services too?"
18    }
  ],
  "max_tokens": 50
}
```

| Key | Value |
|--|------------------|
| <input checked="" type="checkbox"/> Content-Type | application/json |
| <input checked="" type="checkbox"/> api-key | YOUR_API_KEY |
| Key | Value |

Example code for each API endpoint with responses

- Chat Completions:
 - Result response



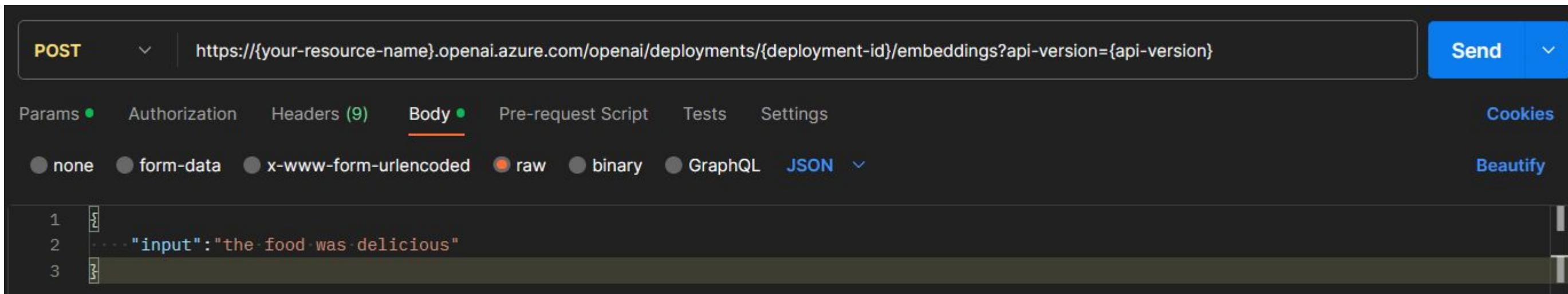
The screenshot shows a REST API response in a dark-themed interface. At the top, there are tabs for 'Body', 'Cookies', 'Headers (12)', and 'Test Results'. On the right, status information is displayed: 'Status: 200 OK', 'Time: 1260 ms', 'Size: 1.5 KB', and a 'Save as example' button. Below the tabs, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The main area displays the API response as a JSON object:

```
29 "choices": [
30   {
31     "index": 0,
32     "finish_reason": "length",
33     "message": {
34       "role": "assistant",
35       "content": "No, it is not the same for all Azure AI services. Some Azure AI services like Azure Cognitive Services are
36         available to individuals, developers, and organizations of all sizes, while others like Azure Machine Learning or Azure
          Databricks are more geared towards enterprise"
      }
    }
]
```

We get a message from the API in `choices.message.content`

Example code for each API endpoint with responses

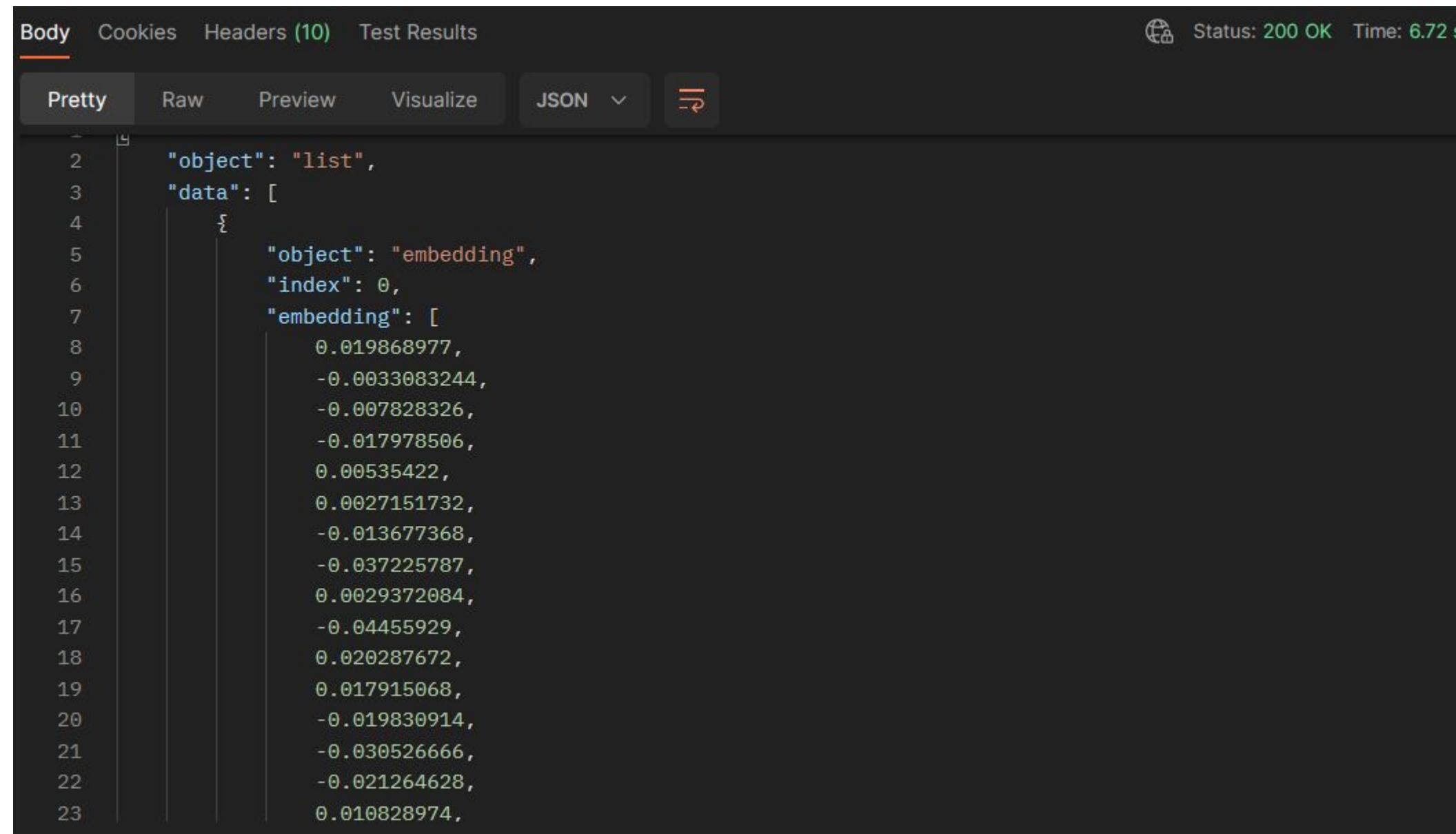
- **Embeddings**



(cant use embeddings api with gpt-3.5-turbo , you need to use the text embedding model: **text-embedding-ada-002**)

Example code for each API endpoint with responses

- **Embeddings**
 - **Response:**



```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON ↻
>Status: 200 OK Time: 6.72 s

2   "object": "list",
3   "data": [
4     {
5       "object": "embedding",
6       "index": 0,
7       "embedding": [
8         0.019868977,
9         -0.0033083244,
10        -0.007828326,
11        -0.017978506,
12        0.00535422,
13        0.0027151732,
14        -0.013677368,
15        -0.037225787,
16        0.0029372084,
17        -0.04455929,
18        0.020287672,
19        0.017915068,
20        -0.019830914,
21        -0.030526666,
22        -0.021264628,
23        0.010828974,
```

We get a long list of embeddings corresponding the input given by us

Example code for each API endpoint with responses

- **Image generation**
 - **Step 1: Request a generated image**

The screenshot shows the Postman application interface for making a POST request to generate an image. The request URL is `https://{{your-resource-name}}.openai.azure.com/openai/images/generations:submit?api-version={{api-version}}`. The 'Body' tab is selected, showing a JSON payload:

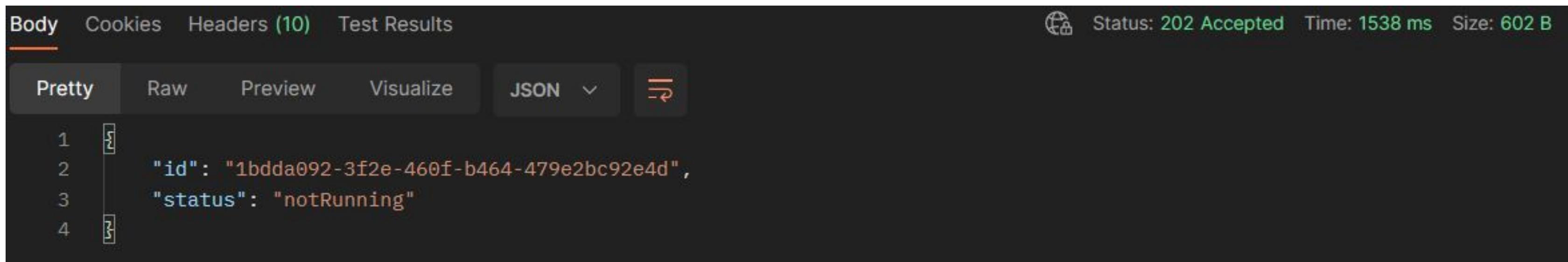
```
1 {
2   "prompt": "A stressed software developer",
3   "size": "512x512",
4   "n": 2
5 }
```

The 'Headers' tab is also selected, displaying the following header values:

| Key | Value |
|--------------|------------------|
| Content-Type | application/json |
| api-key | YOUR_API_KEY |

Example code for each API endpoint with responses

- **Image generation**
 - **Step 1: Request a generated image**
 - **Results**



The screenshot shows a REST API response in a dark-themed interface. At the top, there are tabs for "Body", "Cookies", "Headers (10)", and "Test Results". On the right, it displays "Status: 202 Accepted", "Time: 1538 ms", and "Size: 602 B". Below the tabs, there are buttons for "Pretty", "Raw", "Preview", "Visualize", and "JSON". The "Pretty" button is selected. The response body is a JSON object:

```
1 {  
2   "id": "1bdda092-3f2e-460f-b464-479e2bc92e4d",  
3   "status": "notRunning"  
4 }
```

We get an Id of the operation with its status about the batch of images being generated

Example code for each API endpoint with responses

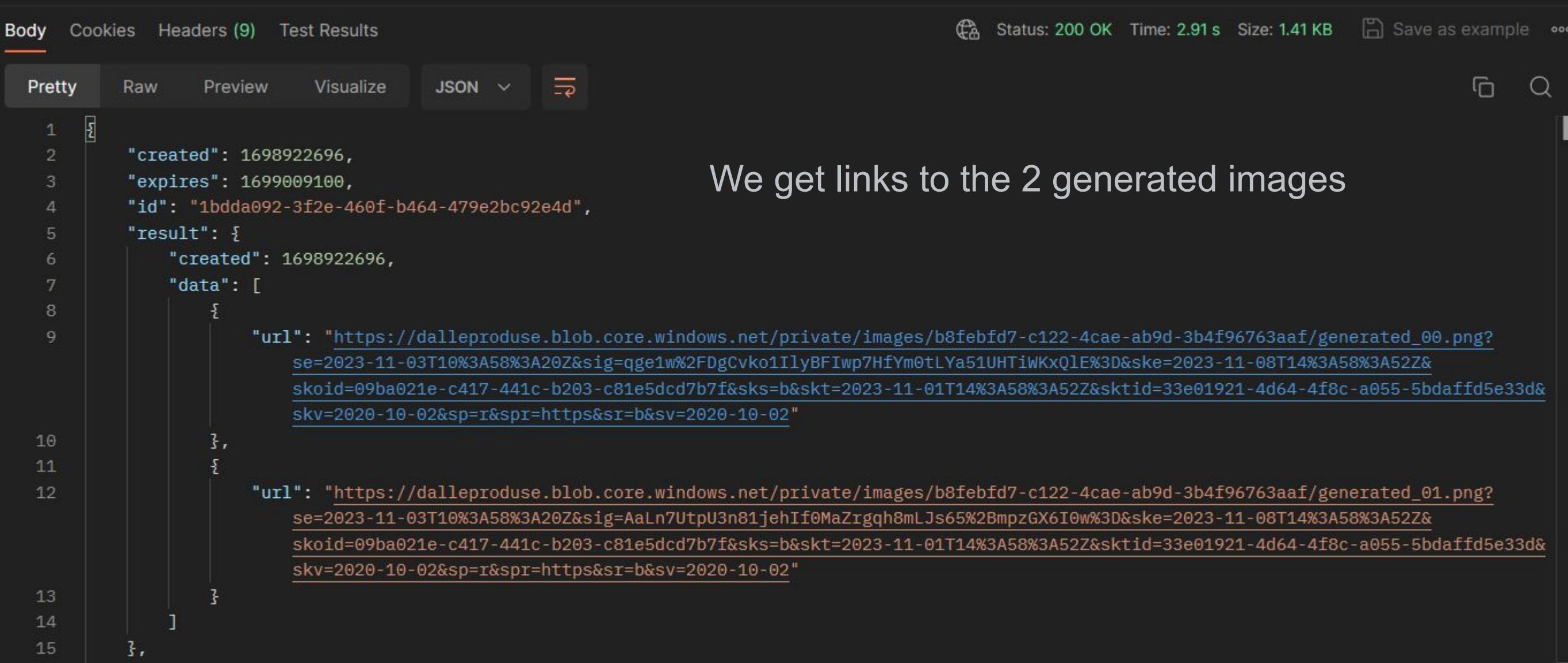
- **Image generation**
 - **Step 2: Get a generated image result**

The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, a URL placeholder 'https://{{your-resource-name}}.openai.azure.com/openai/operations/images/{{operation-id}}?api-version={{api-version}}', and a 'Send' button. Below the header, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is currently active, showing options for 'none', 'form-data', 'x-www-form-urlencoded', 'raw', 'binary', and 'GraphQL'. The 'raw' option is selected. To the right of the body tab, there are 'Cookies' and 'Beautify' buttons. A large, dark green progress bar at the bottom indicates the request is still pending.

(Remember: this is a GET request)

Example code for each API endpoint with responses

- **Image generation**
 - **Step 2: Get a generated image result**
 - **Result**



The screenshot shows a REST client interface with the following details:

- Body:** The response body is displayed in "Pretty" JSON format.
- Headers:** Status: 200 OK, Time: 2.91 s, Size: 1.41 KB.
- Content:** The JSON response contains fields like "created", "expires", "id", and "result". The "result" field is an array containing two objects, each with a "url" field pointing to a generated image. The URLs include query parameters such as "se", "skoid", "skv", "sp", "spr", "sr", and "sv".

We get links to the 2 generated images

```
1  {
2      "created": 1698922696,
3      "expires": 1699009100,
4      "id": "1bdda092-3f2e-460f-b464-479e2bc92e4d",
5      "result": [
6          {
7              "created": 1698922696,
8              "data": [
9                  {
10                     "url": "https://dalleproduse.blob.core.windows.net/private/images/b8febfd7-c122-4cae-ab9d-3b4f96763aaf/generated_00.png?
11                         se=2023-11-03T10%3A58%3A20Z&sig=qge1w%2FDgCvk01IlyBFIwp7HfYm0tLYa51UHTiWKxQ1E%3D&ske=2023-11-08T14%3A58%3A52Z&
12                         skoid=09ba021e-c417-441c-b203-c81e5dcd7b7f&sks=b&skt=2023-11-01T14%3A58%3A52Z&sktid=33e01921-4d64-4f8c-a055-5bdaffd5e33d&
13                         skv=2020-10-02&sp=r&spr=https&sr=b&sv=2020-10-02"
14                 },
15             ]
16         }
17     ]
18 }
```

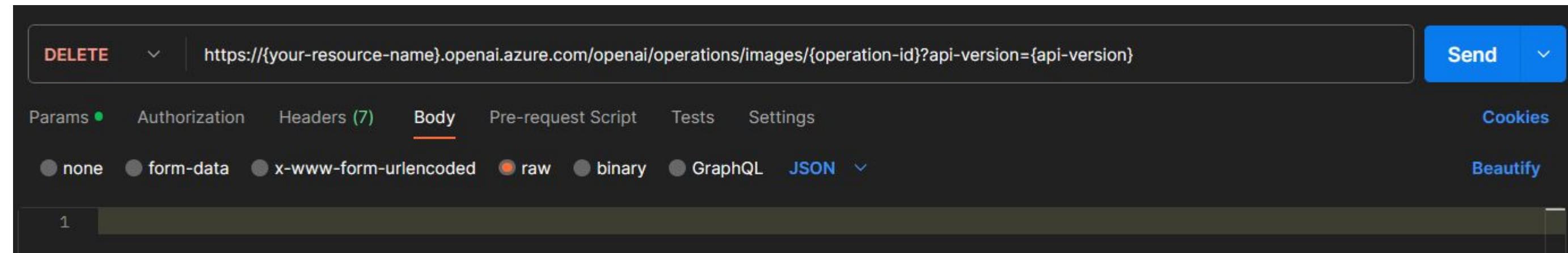
Example code for each API endpoint with responses

- **Image generation**
 - **Step 2: Get a generated image result**
 - **Result**



Example code for each API endpoint with responses

- **Image generation**
 - **Step 3: Delete a generated image**



Example code for each API endpoint with responses

- **Image generation**
 - **Step 3: Delete a generated image**
 - **Result**

The screenshot shows a dark-themed interface for viewing API responses. At the top, there are tabs: Body (which is selected), Cookies, Headers (8), and Test Results. Below the tabs, there are buttons for Pretty, Raw, Preview, Visualize, and Text (with a dropdown arrow). The main content area displays the number '1'. In the top right corner, there is a globe icon followed by the text 'Status: 204 No Content' and 'Time: 1700 ms Size: 356 B'.

We just get a status 204 if successful

Topics

01

Introduction to the PaLM API

02

Generative AI Powered Applications with Python

03

Using the PaLM API in Applications



You can use Gen AI Studio to generate Python code for a Python app or Jupyter

- Click the **View Code** button and select Python or Python Colab
 - Colab is a free Google Jupyter service online

View code

PYTHON

Use this script to request a model response in your application.

1. Set up the [Vertex AI SDK for Python](#)
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="vertext-ai-dar", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    "",
    **parameters
)
print(f"Response from Model: {response.text}")
```

View code

PYTHON PYTHON COLAB

Use this script to request a model response with a Colaboratory notebook.

1. Install the [Vertex AI SDK](#) for Python. After installing you'll be prompted to restart your Colab notebook runtime.
`!pip install "shapely<2.0.0"`
`!pip install google-cloud-aiplatform >= 1.25.0`
2. Use the following code in your notebook to request a model response

```
from google.colab import auth as google_auth
google_auth.authenticate_user()

import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="vertext-ai-dar", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    "",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Azure OpenAI requirements for Python

- Use pip to install Google Azure AI Platform
 - Or add to your requirements.txt file

```
pip install azure-ai-ml  
pip install azure-identity
```

- Import OpenAI

```
pip install openai==0.28.1
```

Using Azure OpenAI from Python

```
import os
import requests
import json
import openai

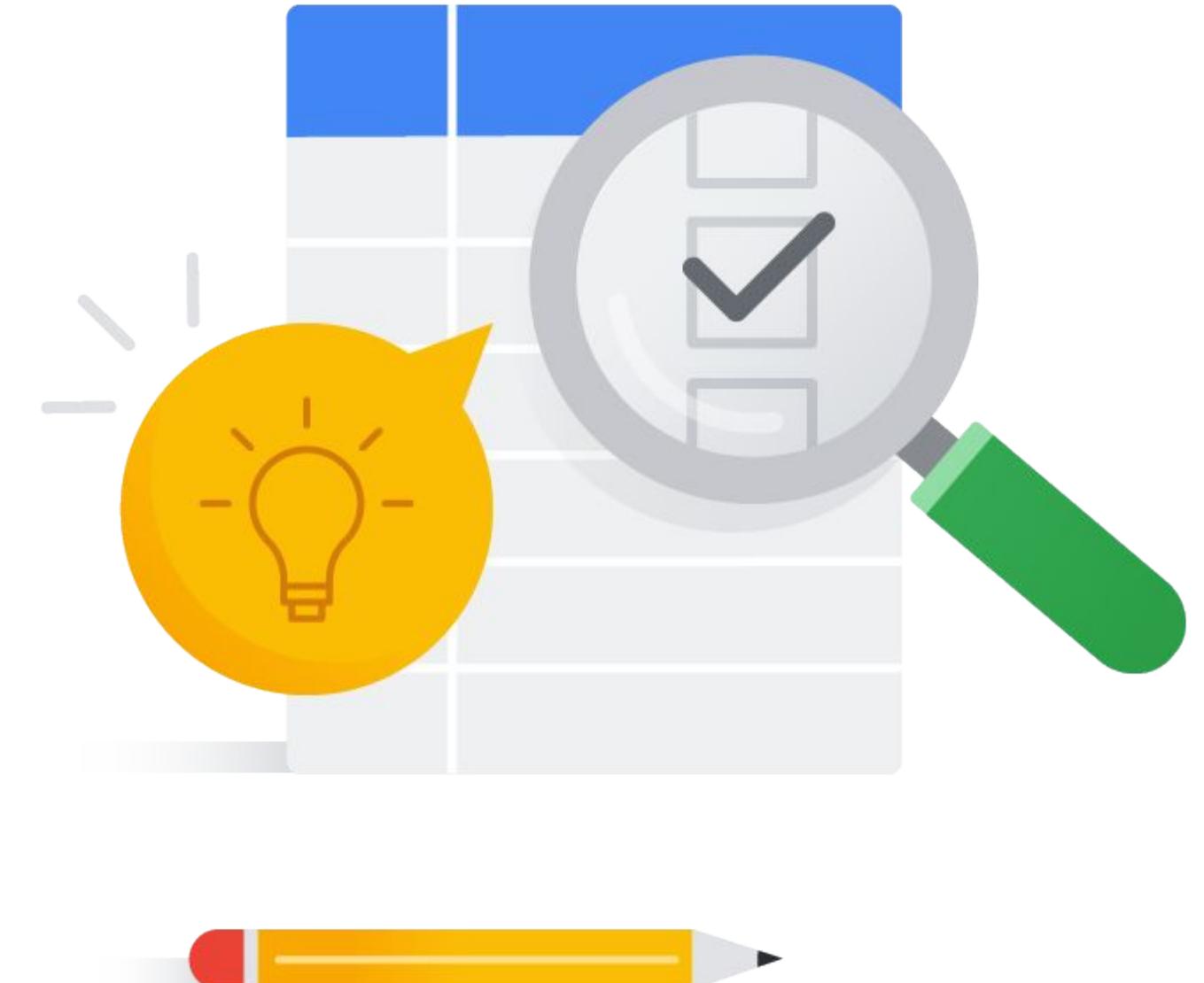
openai.api_key = os.getenv("AZURE_OPENAI_KEY")
openai.api_base = os.getenv("AZURE_OPENAI_ENDPOINT")
openai.api_type = 'azure'
openai.api_version = '2023-05-15'
deployment_name='REPLACE_WITH_YOUR_DEPLOYMENT_NAME'

# Send a completion call to generate an answer
print('Sending a test completion job')
start_phrase = 'Write a tagline for an ice cream shop. '
response = openai.Completion.create(engine=deployment_name, prompt=start_phrase,
max_tokens=10)
text = response['choices'][0]['text'].replace('\n', '').replace(' .', '.').strip()
print(start_phrase, text)
```

Lab

⌚ 30 min ⚙️

Getting Started with the PaLM API



Topics

01

Introduction to the PaLM API

02

Generative AI Powered Applications with Python

03

Using the PaLM API in Applications



Python Flask Website example

- This is an example of using the text service with the OpenAI API
 - Even though you may ask many questions, each one is independent
- Context must be added to tell the OpenAI API to emulate a barista
- The coding is simple as you are just submitting an HTML form and making a request to the OpenAI API for a response
 - The response is displayed on the screen

The image displays two screenshots of a Python Flask website for "CoffeeBot".

Screenshot 1: The top screenshot shows the homepage of CoffeeBot. It features a header with the text "CoffeeBot" and "Your friendly online BaristAI". Below the header is a green box containing the bot's introduction: "I am CoffeeBot, a barista and expert on all things related to coffee and tea. I can help you find the perfect coffee or tea for your taste, and I can also teach you how to make your own coffee and tea drinks at home." A form titled "Ask CoffeeBot:" with a text input field and a "Submit" button is located below the introduction.

Screenshot 2: The bottom screenshot shows the result of a user query. The user has typed "How do you make a Latte?" into the "Ask CoffeeBot:" input field and clicked the "Submit" button. The bot has responded with a green box containing the steps to make a latte: "To make a latte, you will need: * 2 shots of espresso * 6 ounces of steamed milk * 1 tablespoon of foamed milk 1. Brew the espresso shots. 2. Steam the milk until it is hot and frothy. 3. Pour the espresso into a latte glass. 4. Add the steamed milk to the espresso. 5. Top with the foamed milk. 6. Enjoy!"

Add the Python requirements

- Add Azure AI Platform to the requirements.txt file
- Add the required imports at the top of the code file



The image shows a code editor interface with two files open. On the left, a file named "requirements.txt" contains the following dependencies:

```
Flask
jinja2
pytest
pyyaml
azure.identity
```

On the right, a code file contains the following imports:

```
import os
import requests
import json
import openai
```

Handling web requests in Flask

- The default route will handle HTTP posts and gets
 - Post means a question was submitted from the HTML form
- The code for using the AzureOpenAI API is in the **next slide**
- Here we have instantiated our model by providing our api key and our endpoint

```
load_dotenv()  
  
app = Flask(__name__)  
  
openai.api_type = "azure"  
openai.api_base = os.getenv("AZURE_OPENAI_ENDPOINT")  
openai.api_version = "2023-05-15"  
openai.api_key = os.getenv("AZURE_OPENAI_KEY")  
  
@app.route('/', methods=['GET', 'POST'])  
def index():  
    if request.method == 'POST':  
        input = request.form.get('input')
```

Initialize the API

The input is the
question

Handling web requests in Flask

- The default route will handle HTTP posts and gets
 - Post means a question was submitted from the HTML form
 - Get means there is no question (have CoffeeBot introduce itself)
- The code for using the PaLM API is in the `get_response()` function

```
@app.route("/", methods = ['POST', 'GET'])
def main():
    if request.method == 'POST':
        input = request.form['input']
        response = get_response(input)
    else:
        input = ""
        response = get_response("Who are you and what can you do?")

    model = {"title": "CoffeeBot", "message": response, "input": input}
    return render_template('index.html', model=model)
```

Making a request to the AzureOpenAI API

```
def index():
    ...
    response = openai.ChatCompletion.create(
        engine=os.getenv("DEPLOYMENT_NAME"),
        messages=[
            {"role": "system", "content": "Your name is CoffeeBot.
You are a barista and expert on all things related to coffee and
tea"},
            {"role": "user", "content": input}],
    )
    return render_template('form_response.html', response =
response['choices'][0]['message']['content'])
    return render_template('form_response.html'))

if __name__ == '__main__':
    app.run(debug=True)
```

Call the
openai.ChatCompletion.create
function to send the request

The context tells
the API to
emulate a barista

We put the input(our question)
into the prompt

Making a request to the PaLM API

```
def get_response(input):
    vertexai.init(project="vertext-ai-dar", location="us-central1")
    parameters = {
        "temperature": 0.8,
        "max_output_tokens": 256,
        "top_p": 0.8,
        "top_k": 40
    }
    model = TextGenerationModel.from_pretrained("text-bison@001")
```

Initialize the API
and set up the
parameters

Create the model
using the correct
version of the
PaLM API

<< CODE ON NEXT SLIDE OMITTED >>

Making a request to the AzureOpenAI API (continued)

```
<!DOCTYPE html>
<html>
<head>
  <title>Form and Response</title>
</head>
<body>
  <h1>CoffeeBot</h1>
  <h2>Your friendly online BaristAI</h2>

  <h2>Ask coffeeBot</h2>
  <form method="POST">
    <input type="text" name="input" placeholder="Your input" required>
    <input type="submit" value="Submit">
  </form>
  <h2>Response</h2>
  <p>{{ response }}</p>
</body>
</html>
```

A simple html
form to use our
CoffeeBot

Making a request to the AzureOpenAI API (continued)

Here's an example screenshot:

The screenshot shows a two-panel interface for a coffee bot named "coffeeBot".

User Request Panel (Left):

- CoffeeBot**
- Your friendly online BaristAI**
- Ask coffeeBot**
-

Response Panel (Right):

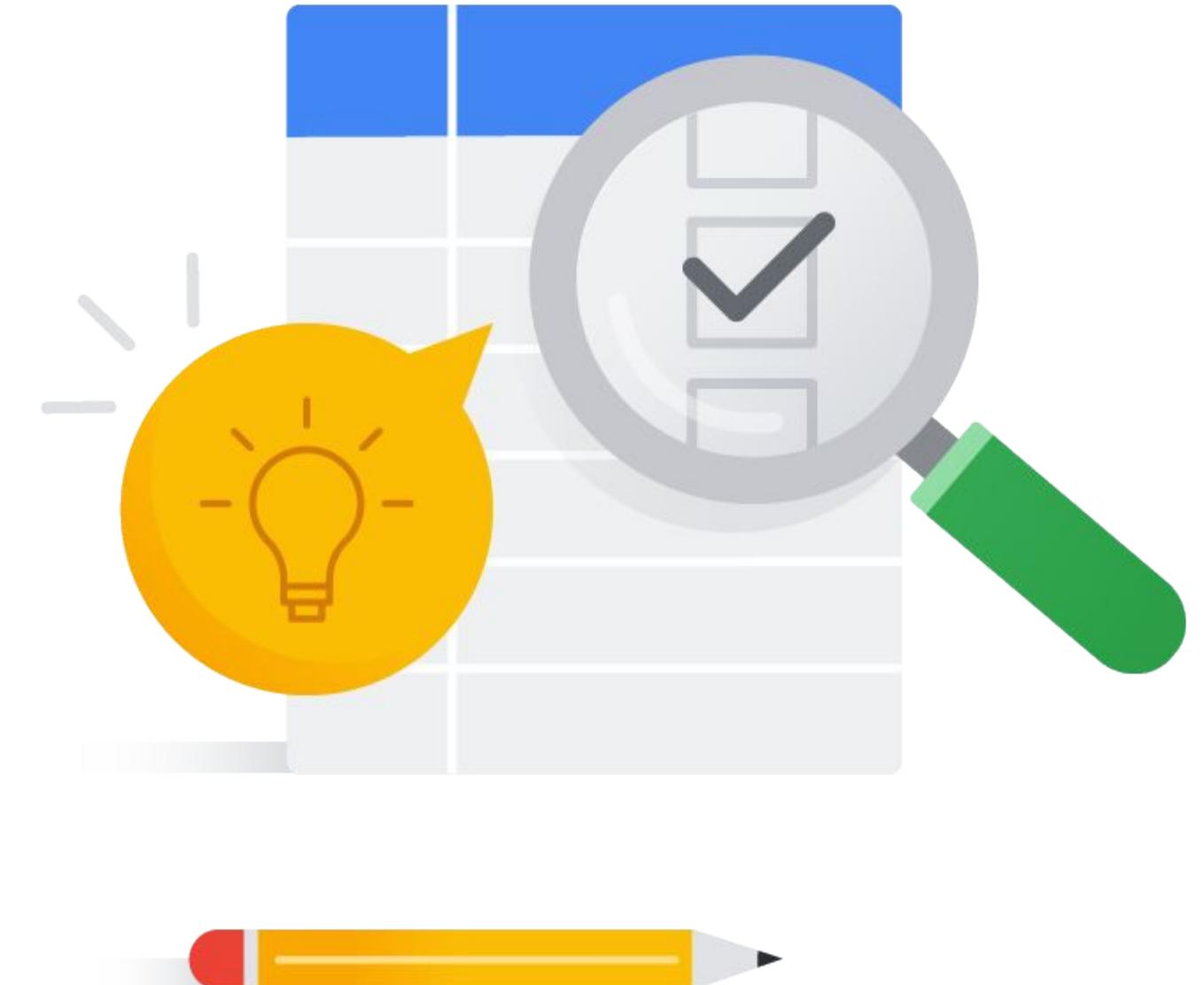
- CoffeeBot**
- Your friendly online BaristAI**
- Ask coffeeBot**
-
- Response**
- Certainly! We have disposable cups available for our to-go orders. What type of coffee or tea would you like in each cup?

Lab

30 min



Integrating the AzureOpenAI API into Applications



In this module, you learned to ...

01

Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02

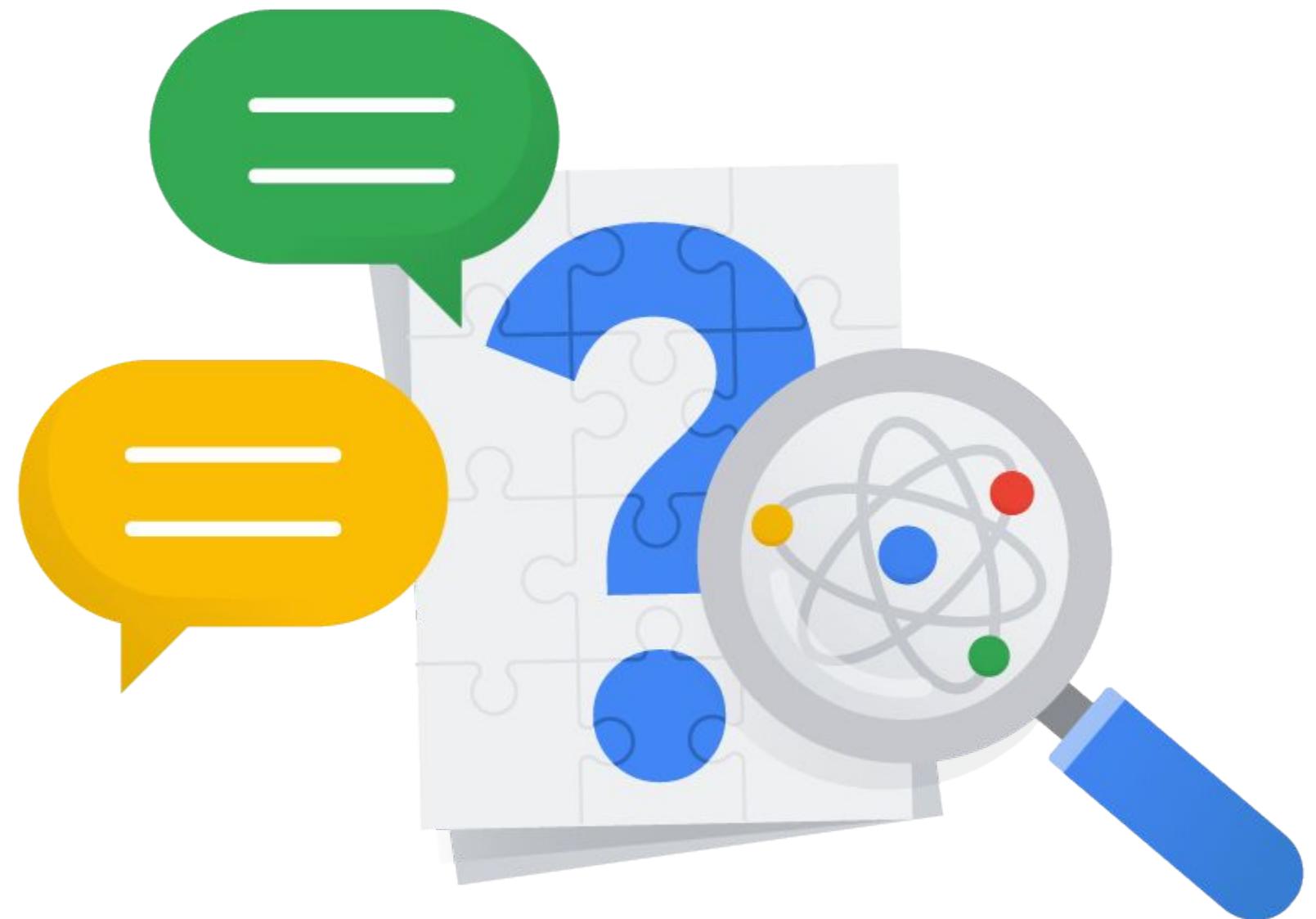
Program Python applications that use PaLM to generate content

03

Integrate PaLM and GenAI into your applications



Questions and answers



Quiz question

How do you authenticate a request to the Azure OpenAi API ?

- A: Using a API Key
- B: Using an authorization token
- C: With a Service Account
- D: All of the above would work

Quiz question

How do you authenticate a request to
the Azure OpenAI API ?

- A: Using a API Key
- B: Using an authorization token
- C: With a Service Account
- D: All of the above would work

Quiz question

What programming languages are
Supported by the AzureOpenAI API ?

- A: Python
- B: Node.js
- C: Go
- D: Java
- E: All of the above

Quiz question

What programming languages are
Supported by the AzureOpenAI API ?

- A: Python
- B: Node.js
- C: Go
- D: Java
- E: All of the above

Context is important

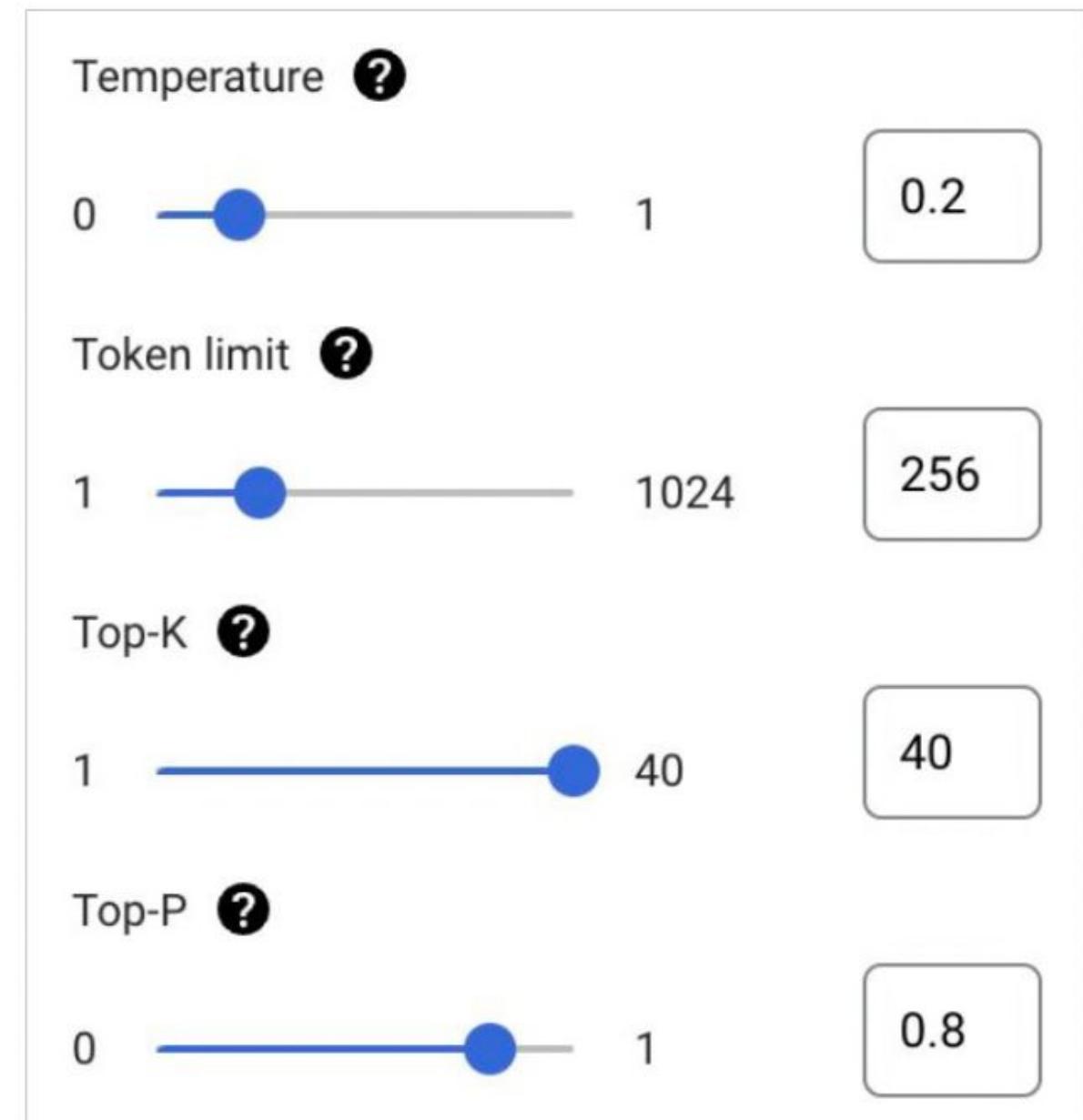
- Context helps the model generate output specific to your organization or use case
- Add information about your company that you want the model to return
 - Website
 - Phone
 - Address
 - Etc.
- Specify the style of output that you desire

Add examples

- Examples help the model return output that fits your parameters
- You don't want the model to just make things up
 - For example, you don't want the ChatBot to tell the user they have an appointment on Tuesday when it is not connected to any sort of database or calendar
- Anticipate typical questions, and provide acceptable responses

Adjust parameters according to your use case

- Sometimes you want a creative chat bot
 - Producing marketing content
 - Generating social media posts
- Sometimes you want the bot to be restricted to a small set of acceptable answers
 - Customer service chat
 - Health advice
- For more creativity, set Temperature, Top-K, and Top-P higher and visa-versa



Thoroughly test your applications

- Generative AI will often make things up
- If interfacing with customers, you want to be sure they are given correct answers
- If using Gen AI for content generation, don't assume it is correct
 - Edit results for accuracy



Fine-tuning Models

In this module, you learn to ...

01

Evaluate scenarios for creating tuned models

02

Build workflows for tuning and deploying models

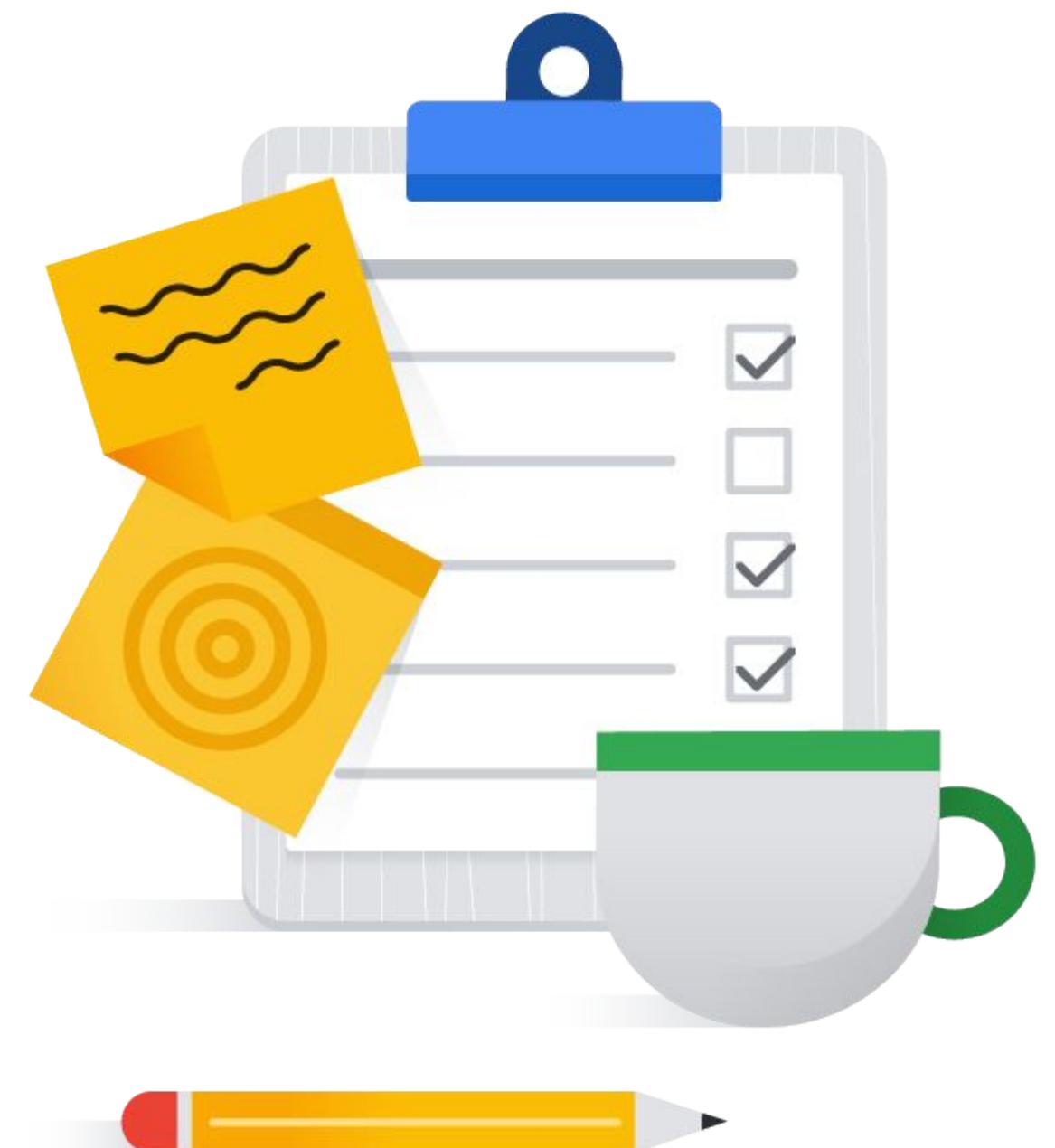
03

Use tuned models in your applications



Topics

- 01 Tuned Models
- 02 Preparing a Model Tuning Dataset
- 03 Creating a Tuning Job



Fine-Tuning

Fine-tuning lets you get more out of the models available through the API by providing:

- Higher quality results than prompting
- Ability to train on more examples than can fit in a prompt
- Token savings due to shorter prompts
- Lower latency requests

Fine-Tuning

GPT models have been pre-trained on a vast amount of text. To use the models effectively, we include instructions and sometimes several examples in a prompt. Using demonstrations to show how to perform a task is often called **"few-shot learning."**

Fine-tuning improves on few-shot learning by training on many more examples than can fit in the prompt, letting you achieve better results on a wide number of tasks. **Once a model has been fine-tuned, you won't need to provide as many examples in the prompt. This saves costs and enables lower-latency requests.**

What models can be fine-tuned?

Fine-tuning is currently available for the following models:

- gpt-3.5-turbo-0613 (recommended)
- babbage-002
- davinci-002

When to use fine-tuning?

Some common use cases where fine-tuning can improve results:

- Setting the style, tone, format, or other qualitative aspects
- Improving reliability at producing a desired output
- Correcting failures to follow complex prompts
- Handling many edge cases in specific ways
- Performing a new skill or task that's hard to articulate in a prompt
- Specific structures or formats for generating output
- Specific behaviors such as when to provide a terse or verbose output
- Specific customized outputs for specific types of inputs

Tuning is required when you want output that deviates from general language patterns

- Specific structures or formats for generating output
- Specific behaviors such as when to provide a terse or verbose output
- Specific customized outputs for specific types of inputs

When custom training may be required: Classification

- Classification with custom classes (groups)
 - Give the model examples, with the correct answers

```
{"role": "User",
"Content": "Classify the following text into one of the following classes:
[HR, Sales, Marketing, Customer Service].
Text: Are you currently hiring?"
}
```

```
{"role": "assistant",
"Content": "HR"
}
```

When custom training may be required:

Summarization

- Summaries that require specific output
- In the example below, you want to remove personally identifiable information (PII) in a chat summary

```
{"role": "User",
"Content": "Summarize:
Jessica: That sounds great! See you in Times Square!
Alexander: See you at 10!"
}
```

```
{"role": "assistant",
"Content": "#Person1 and #Person2 agree to meet at Times Square at 10:00
AM"
}
```

When custom training may be required: Extractive question answering

- The question is about a context and the answer is a substring of the context

```
{"role": "system",
"Content": "There is evidence that there have been significant changes
in Amazon rainforest vegetation over the last 21,000 years through the
Last Glacial Maximum (LGM) and subsequent deglaciation."
}
```

```
{"role": "User",
"Content": "question: What does LGM stand for?"
}
```

```
{"role": "assistant",
"Content": "Last Glacial Maximum"
}
```

Including context in your training data

- In the example below, the **messages array** consists of both a **system** section and a **user** section

```
{"role": "system",
"Content": "There is evidence that there have been significant changes in
Amazon rainforest vegetation over the last 21,000 years through the Last
Glacial Maximum (LGM) and subsequent deglaciation."}
{"role": "User",
"Content": "question: What does LGM stand for?"}
```

- The “system” context provides additional information for answering the question
- If your training data is formatted in this way, you must format prompts in the same way when using your model for inference
 - I.e. Whatever sections your training data has as input, must be included in prompts in the same order when using the model

Topics

01

Tuned Models

02

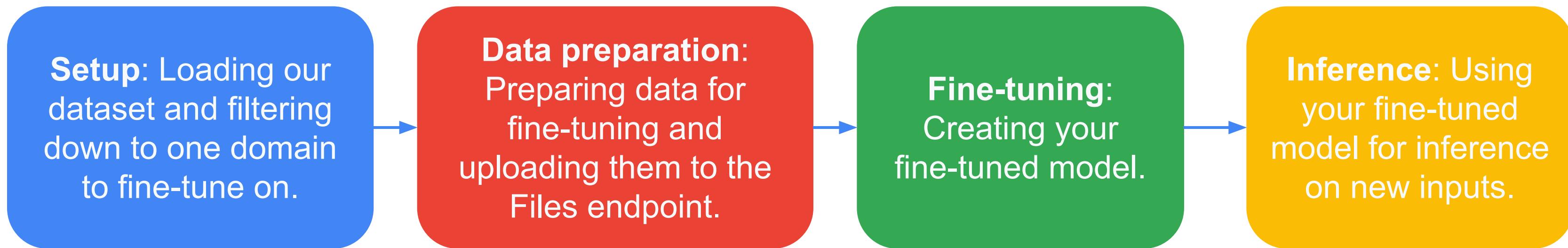
Preparing a Model Tuning Dataset

03

Creating a Tuning Job



Model tuning workflow on Vertex AI



- After tuning, the model is automatically deployed to a AzureOpenAI endpoint using the name you provide in the tuning job
- The model is also available in Azure oai Portal when creating prompts

Prepare your model tuning dataset

- The training data must be in JSONL format
 - The “L” is for “Line”
 - Each line in the JSONL file is one example
 - It is not an array of objects, it is one object per line
- Each object must have the properties `system`, `user` and `assistant`

```
{"messages": [{"role": "system", "content": "<context and behavioural specification>"}, {"role": "user", "content": "<prompt example text>"}, {"role": "assistant", "content": "<example desired output text>"}]}
```

It is important to include instructions into the training data

- The following has no instructions, so it is not a good example

```
{"input_text": "5 stocks to buy now", "output_text": "business"}
```

- The following has instructions, so it is a better example

```
{"input_text": "Classify the following text into one of the following classes: [business, entertainment] Text: 5 stocks to buy now", "output_text": "business"}
```

Including context within the input text

- Notice that the following `input_text` has question and context sections
 - When using the model, remember that prompts need to be formatted the same way
 - Be consistent

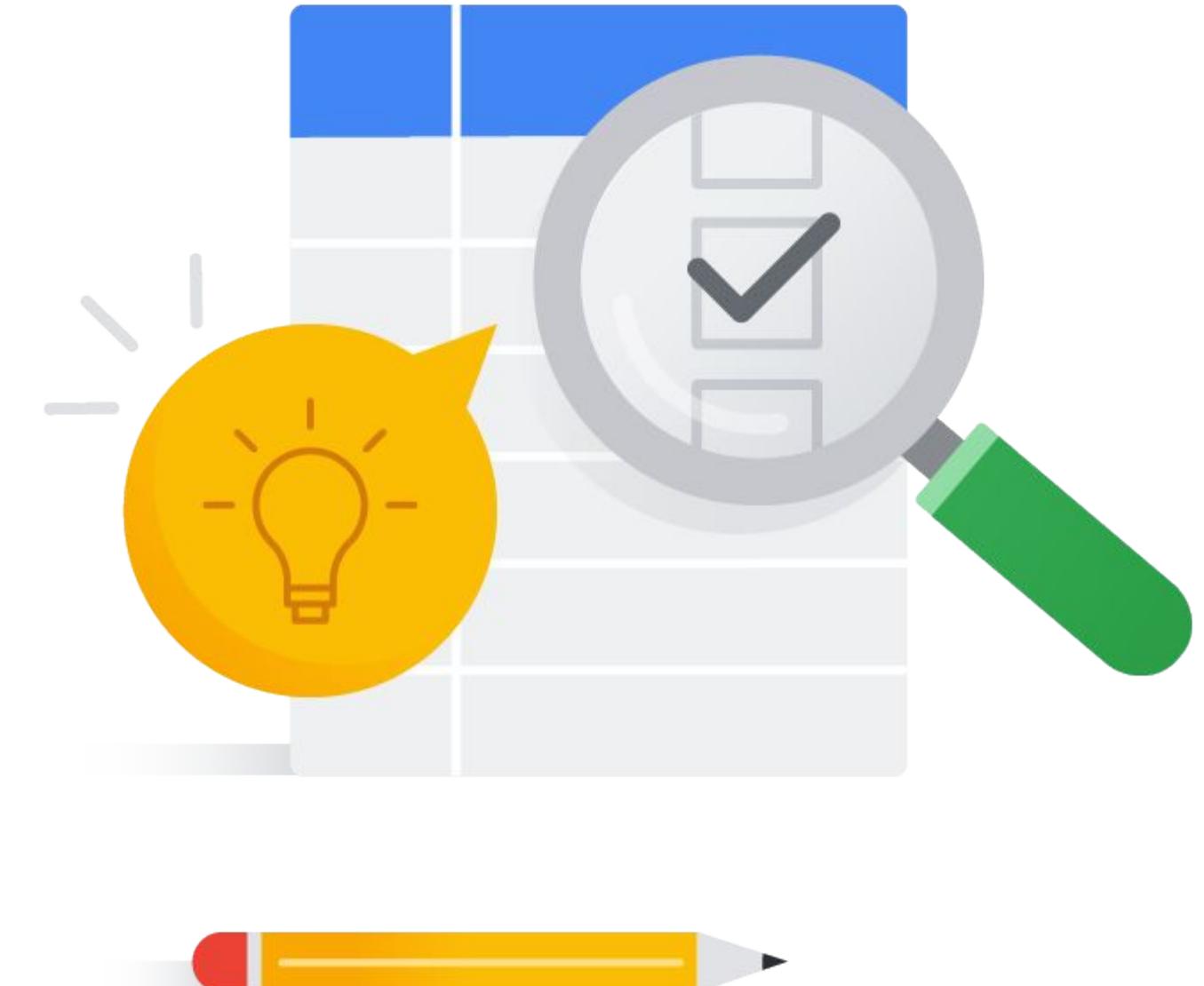
```
{"input_text": "question: How many parishes are there in Louisiana? context: The U.S. state of Louisiana is divided into 64 parishes (French: paroisses) in the same manner that 48 other states of the United States are divided into counties, and Alaska is divided into boroughs.", "output_text": "64"}
```

Do Now: Exploring Sample Training Data

⌚ 5 min



1. Go to:
<https://github.com/roitraining/genai-model-tuning-examples>
2. You will find some example fine-tuning datasets
3. Click on a couple of them and explore the examples
 - a. Each file has 1 example per line
 - b. Each example has input_text and output_text attributes



Topics

- 01 Tuned Models
- 02 Preparing a Model Tuning Dataset
- 03 Creating a Tuning Job



Specify your base model, your training and validation data, and set any hyperparameters

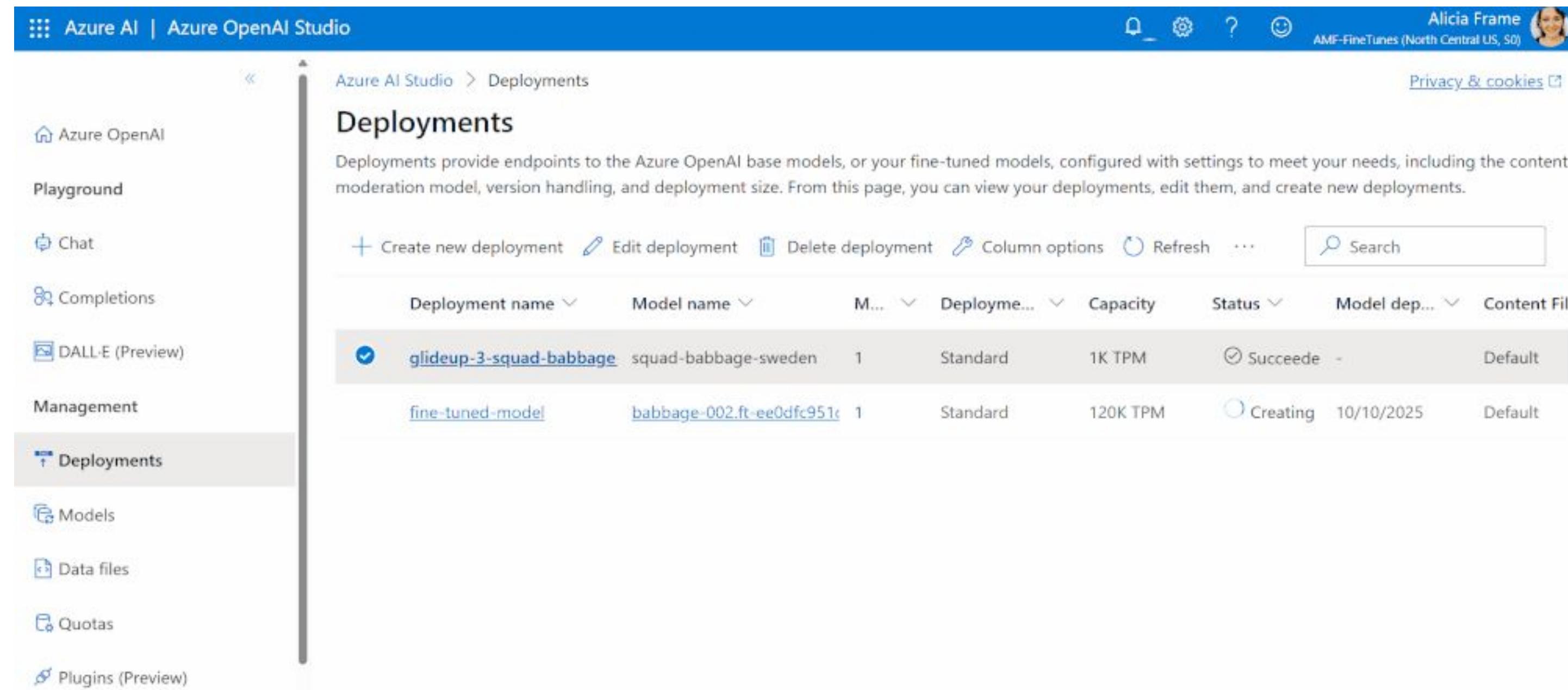
The screenshot shows the Azure OpenAI Studio interface, specifically the 'Models' section. The top navigation bar includes 'Azure AI | Azure OpenAI Studio', a user profile for 'Alicia Frame', and links for 'Privacy & cookies'. The left sidebar has icons for Home, Models, Playground, and Help. The main content area is titled 'Models' and contains a brief description of Azure OpenAI's capabilities. It features two tabs: 'Custom models' (selected) and 'Base models'. Below the tabs are buttons for 'Deploy', 'Create a custom model', 'Delete', 'Column options', and 'Refresh', along with a search bar. A table lists five custom models:

| Model name | M... | Cr... | Base model | Status | Deploya... | Training job ID |
|-----------------------------------|------|------------|-------------|--|--------------------------------------|------------------------------|
| babbage-002.ft- | 1 | 10/8/2021 | babbage-002 | <input checked="" type="radio"/> Succeeded | <input checked="" type="radio"/> Yes | ftjob-2afb79d352654aa6944d6 |
| babbage-002.ft- | 1 | 10/7/2021 | babbage-002 | <input checked="" type="radio"/> Succeeded | <input checked="" type="radio"/> Yes | ftjob-c598c864f3664b21a6c5c2 |
| babbage-002.ft- | 1 | 10/13/2021 | babbage-002 | <input checked="" type="radio"/> Succeeded | <input checked="" type="radio"/> Yes | ftjob-d93edafae63d41919f688 |
| babbage-002.ft- | 1 | 10/10/2021 | babbage-002 | <input checked="" type="radio"/> Succeeded | <input checked="" type="radio"/> Yes | ftjob-e734e9b35f1f4a3d948e31 |
| davinci-002.ft-1x | 1 | 10/7/2021 | davinci-002 | <input checked="" type="radio"/> Succeeded | <input checked="" type="radio"/> Yes | ftjob-146a08ea2c254e819cbfe1 |

The number of training examples and training steps needed depends on the task

| Task | Suggested # of examples | Training steps |
|----------------|-------------------------|----------------|
| Classification | 100+ | 100-500 |
| Summarization | 100-500+ | 200-1000 |
| Extractive QA | 100+ | 100-500 |

Tuned models are available from AzureOpenAI Studio



The screenshot shows the Azure OpenAI Studio interface with the 'Deployments' section selected. The left sidebar includes links for Azure OpenAI, Playground, Chat, Completions, DALL-E (Preview), Management, Deployments (selected), Models, Data files, Quotas, and Plugins (Preview). The top navigation bar shows 'Azure AI | Azure OpenAI Studio' and the user 'Alicia Frame AMF-FineTunes (North Central US, SD)'.

Deployments

Deployments provide endpoints to the Azure OpenAI base models, or your fine-tuned models, configured with settings to meet your needs, including the content moderation model, version handling, and deployment size. From this page, you can view your deployments, edit them, and create new deployments.

| Deployment name | Model name | M... | Deployme... | Capacity | Status | Model dep... | Content Fil |
|-------------------------|---------------------------|------|-------------|----------|---|--------------|-------------|
| glideup-3-squad-babbage | squad-babbage-sweden | 1 | Standard | 1K TPM | <input checked="" type="checkbox"/> Succeeded | - | Default |
| fine-tuned-model | babbage-002.ft-ee0dfc951e | 1 | Standard | 120K TPM | <input type="radio"/> Creating | 10/10/2025 | Default |

Gen AI Studio will generate the code for using tuned models

View code

PYTHON PYTHON COLAB CURL

Use this script to request a model response in your application.

1. Set up the [Vertex AI SDK for Python](#)
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.preview.language_models import TextGenerationModel

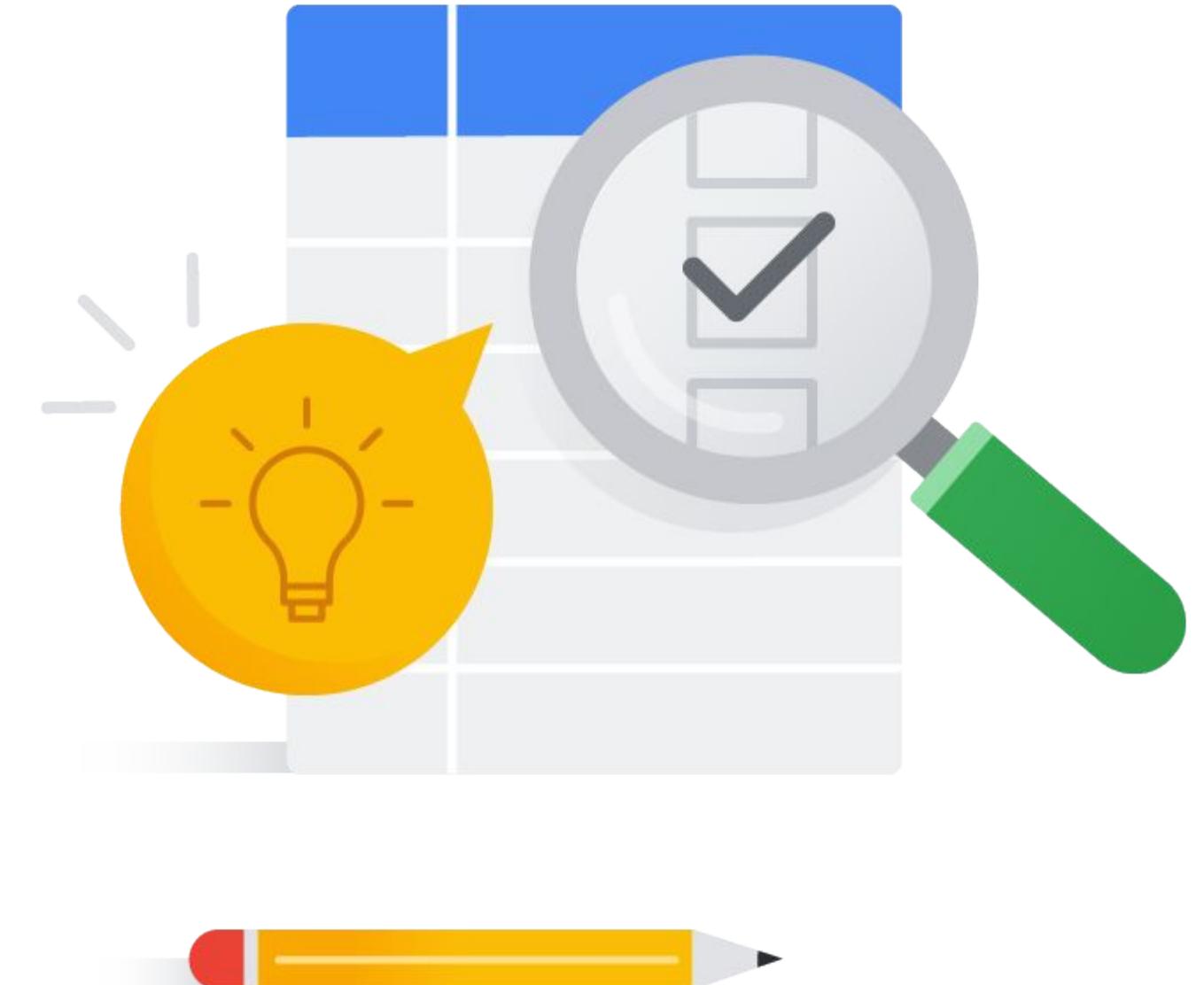
vertexai.init(project="982785856251", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

model = TextGenerationModel.from_pretrained("text-bison@001")
model = model.get_tuned_model("projects/982785856251/locations/us-central1/models/167990643268360"
response = model.predict(
    "What is the capital of France?",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Demo

⌚ 15 min ⚙️

Fine-Tuning Models for Specific Use Cases



A step-by-step guide for gpt-3.5-turbo fine-tuning

Install the openai package

```
Mudit@LAPTOP-K9RFR9TJ MINGW64 ~/Desktop/DC
$ pip install openai
```

Import the necessary packages

```
from pprint import pprint
from dotenv import load_dotenv
import pandas as pd
import json
import openai
import os
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Set up your azure openAI api

```
load_dotenv()
openai.api_type = "azure"
openai.api_base = os.getenv("AZURE_OPENAI_ENDPOINT")
openai.api_version = "2023-05-15"
openai.api_key = os.getenv("AZURE_OPENAI_KEY")
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Now load up your dataset (in this example we're using [RecipeNLG dataset](#))

```
# Read in the dataset we'll use for this task.  
# This will be the RecipesNLG dataset, which we've cleaned to only contain documents from www.cookbooks.com  
recipe_df = pd.read_csv("data/cookbook_recipes_nlg_10k.csv")  
  
recipe_df.head()
```

| | title | ingredients | directions | link | source |
|---|-----------------------|---|---|---|--|
| 0 | No-Bake Nut Cookies | ["1 c. firmly packed brown sugar", "1/2 c. eva... | ["In a heavy 2-quart saucepan, mix brown sugar... | www.cookbooks.com/Recipe-Details.aspx?id=44874 | ["brown sugar", "milk", "vanilla", "nuts", "bu... |
| 1 | Jewell Ball'S Chicken | ["1 small jar chipped beef, cut up", "4 boned ... | ["Place chipped beef on bottom of baking dish.... | www.cookbooks.com/Recipe-Details.aspx?id=699419 | ["beef", "chicken breasts", "cream mushroom... |
| 2 | Creamy Corn | ["2 (16 oz.) pkg. frozen corn", "1 (8 oz.) pkg... | ["In a slow cooker, combine all ingredients. C... | www.cookbooks.com/Recipe-Details.aspx?id=10570 | ["frozen corn", "cream cheese", "butter", "gar... |
| 3 | Chicken Funny | ["1 large whole chicken", "2 (10 1/2 oz.) cans... | ["Boil and debone chicken.", "Put bite size pi... | www.cookbooks.com/Recipe-Details.aspx?id=897570 | ["chicken", "chicken gravy", "cream of mushroom... |

A step-by-step guide for gpt-3.5-turbo fine-tuning

We need our database to follow the ChatCompletions foormat so we write these functions

```
training_data = []

system_message = "You are a helpful recipe assistant. You are to extract the generic ingredients from each of the recipes provided."

def create_user_message(row):
    return f"""Title: {row['title']}\n\nIngredients: {row['ingredients']}\n\nGeneric ingredients: """

def prepare_example_conversation(row):
    messages = []
    messages.append({"role": "system", "content": system_message})

    user_message = create_user_message(row)
    messages.append({"role": "user", "content": user_message})

    messages.append({"role": "assistant", "content": row["NER"]})

    return {"messages": messages}
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Separate the data into training and validation data and transform training data according to chatCompletions api using our function

```
# use the first 100 rows of the dataset for training
training_df = recipe_df.loc[0:100]

# apply the prepare_example_conversation function to each row of the training_df
training_data = training_df.apply(prepare_example_conversation, axis=1).tolist()
```

Do the same for our validation data

```
validation_df = recipe_df.loc[101:200]
validation_data = validation_df.apply(prepare_example_conversation, axis=1).tolist()
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Now we write a function to turn our data into the JSONL format

We need to save our data as .jsonl files with each line being a training example conversation

```
def write_jsonl(data_list: list, filename: str) -> None:
    with open(filename, "w") as out:
        for ddict in data_list:
            jout = json.dumps(ddict) + "\n"
            out.write(jout)
```

```
training_file_name = "tmp_recipe_finetune_training.jsonl"
write_jsonl(training_data, training_file_name)

validation_file_name = "tmp_recipe_finetune_validation.jsonl"
write_jsonl(validation_data, validation_file_name)
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

File Upload:

We can now upload the files to our Files endpoint to be used by the fine-tuned model.

```
training_response = openai.File.create(  
    file=open(training_file_name, "rb"), purpose="fine-tune"  
)  
training_file_id = training_response["id"]  
  
validation_response = openai.File.create(  
    file=open(validation_file_name, "rb"), purpose="fine-tune"  
)  
validation_file_id = validation_response["id"]  
  
print("Training file ID:", training_file_id)  
print("Validation file ID:", validation_file_id)
```

```
Training file ID: file-jcdvN127iuBMZfwI4q30IIka  
Validation file ID: file-01440IHkZ1xjB32ednBmb0XP
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Fine tuning:

Now we create a fine tuning job using our file id

```
response = openai.FineTuningJob.create(  
    training_file=training_file_id,  
    validation_file=validation_file_id,  
    model="gpt-3.5-turbo",  
    suffix="recipe-ner",  
)  
  
job_id = response["id"]  
  
print("Job ID:", response["id"])  
print("Status:", response["status"])  
  
Job ID: ftjob-9xVzrp00em9rWj2UFWDcFLqT  
Status: created
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Fine tuning:

After sometime we can get a model Id of our fine tuned model

```
response = openai.FineTuningJob.retrieve(job_id)
fine_tuned_model_id = response["fine_tuned_model"]

if fine_tuned_model_id is None:
    raise RuntimeError("Fine-tuned model ID not found. Your job has likely not been completed yet.")

print("Fine-tuned model ID:", fine_tuned_model_id)
```

```
Fine-tuned model ID: ft:gpt-3.5-turbo-0613:openai:recipe-ner:7qTvyJ81
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Inference:

We Simply call ChatCompletions with our new fine-tuned model name filling the model parameter.

```
test_df = recipe_df.loc[201:300]
test_row = test_df.iloc[0]
test_messages = []
test_messages.append({"role": "system", "content": system_message})
user_message = create_user_message(test_row)
test_messages.append({"role": "user", "content": create_user_message(test_row)})

pprint(test_messages)

[{'content': "You are a helpful recipe assistant. You are to extract the 'generic ingredients from each of the recipes provided.",
  'role': 'system'},
 {'content': "Title: Pancakes\n\n'Ingredients: ["1 c. flour", "1 tsp. soda", "1 tsp. salt", "1 Tbsp. sugar", "1 egg", "3 Tbsp. margarine, melted", "1 c. buttermilk"]\n\n'Generic ingredients: '",
  'role': 'user'}]
```

A step-by-step guide for gpt-3.5-turbo fine-tuning

Inference:

We Simply call ChatCompletions with our new fine-tuned model name filling the model parameter.

```
response = openai.ChatCompletion.create(  
    model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500  
)  
print(response["choices"][0]["message"]["content"])  
  
["flour", "soda", "salt", "sugar", "egg", "margarine", "buttermilk"]
```

In this module, you learned to ...

- 01 Evaluate scenarios for creating tuned models
- 02 Build workflows for tuning and deploying models
- 03 Use tuned models in your applications



Vector databases and OpenAI embeddings

Vector Embeddings

Vector embeddings are numerical representations of high-dimensional data such as text, images, audio, and video. They are created using mathematical algorithms that transform the raw data into vectors with a fixed number of dimensions

Vector embeddings are used to capture the semantic or contextual meaning of the data, allowing it to be compared and analyzed in a meaningful way.

For example, in natural language processing, words can be represented as vectors in a high-dimensional space, where words with similar meanings are closer together in the space than words with different meanings.

These embeddings can be created using various methods such as neural networks, word2vec, and other machine learning techniques. The resulting embeddings can be used for a wide range of applications, such as language translation, sentiment analysis, recommendation systems, and image recognition.

Vector databases and OpenAI embeddings

Vector Databases

A vector database is a specialized type of database that stores data as high-dimensional vectors. Vectors are mathematical representations of features or attributes that can range from tens to thousands of dimensions depending on the complexity of the data. The vectors are generated by applying an embedding function to the raw data, such as text, images, audio, and video.

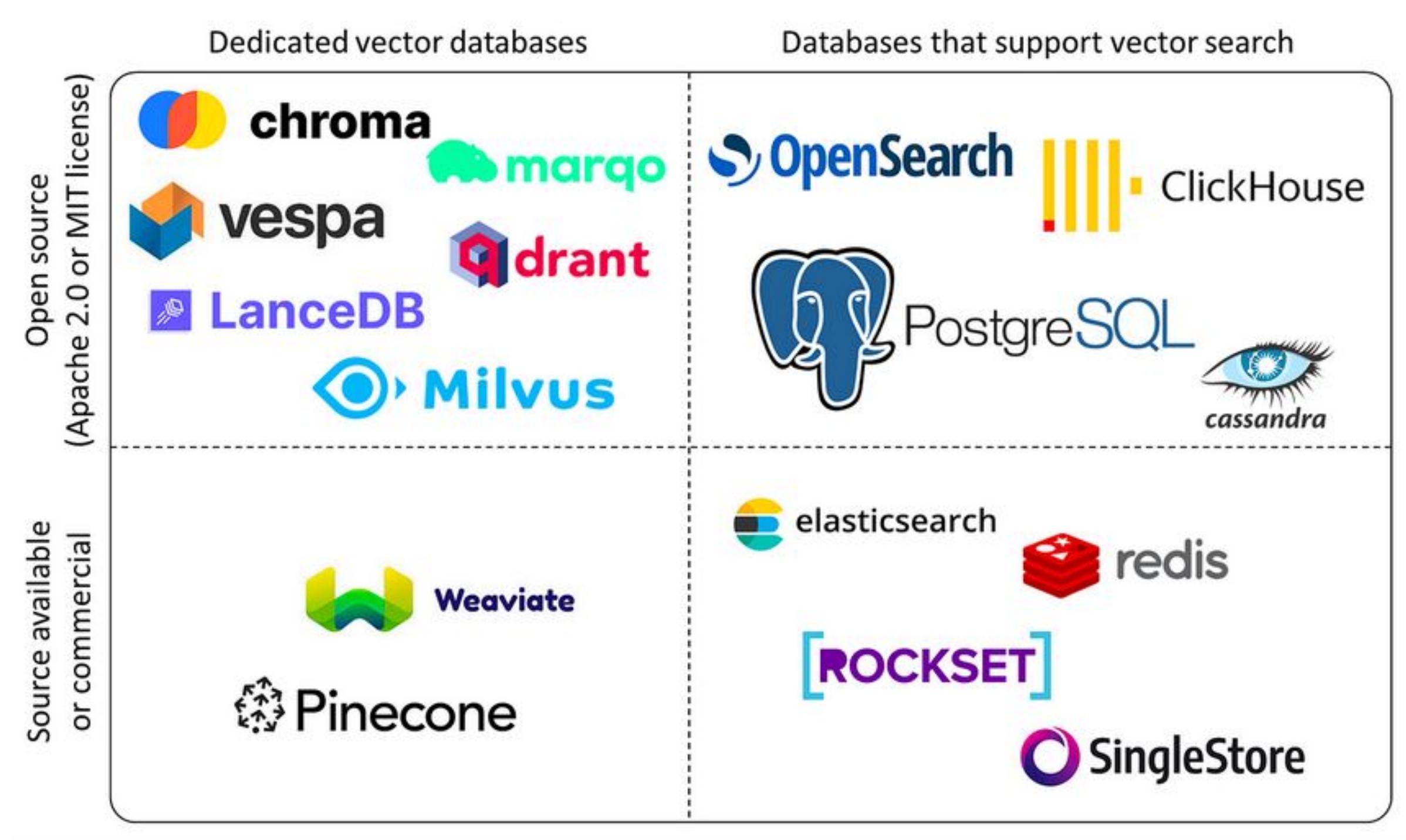
The embedding function can be based on machine learning models, word embeddings, or feature extraction algorithms. Vector databases offer the advantage of fast and accurate similarity search and retrieval of data based on their vector distance or similarity.

Unlike traditional databases that rely on exact matches or predefined criteria for querying data, vector databases allow you to find the most similar or relevant data based on their semantic or contextual meaning. This feature makes vector databases ideal for a wide range of applications such as natural language processing, image recognition, and recommendation systems.

The main advantage of using vector databases is how fast and accurate they are in similarity search and retrieval, based on vector distance. This works better than query-based databases which try to find exact matches, a vector database gets the most similar or relevant data based on semantic or contextual meaning. Below are usage strategies for using vector databases:

Vector databases and OpenAI embeddings

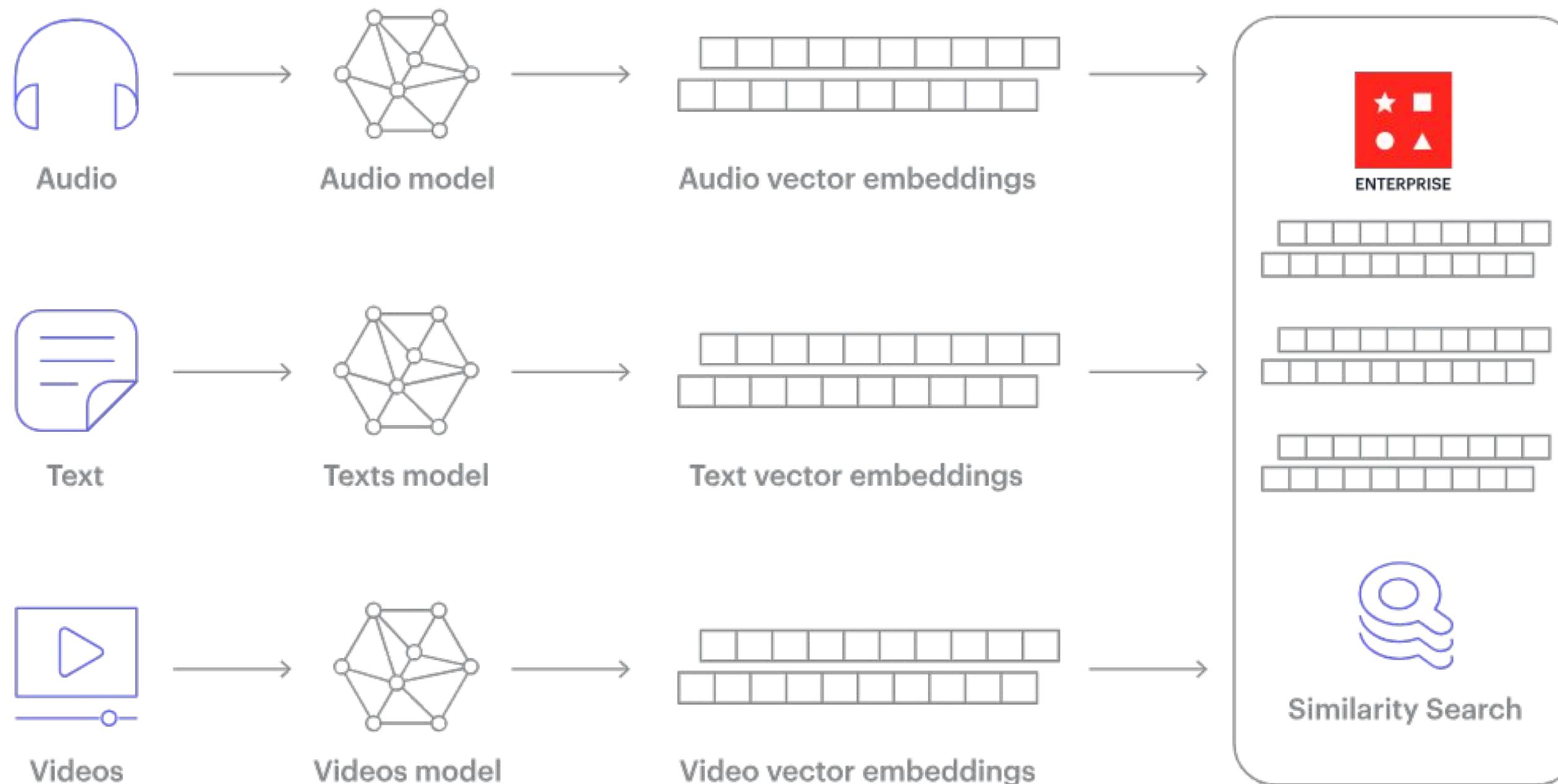
The vector database landscape



Vector databases and OpenAI embeddings

Common vector database usage:

- Search images based on their visual content and style
- Search documents based on their topic and sentiment
- Search products based on their features and ratings



Azure OpenAI | When to use it

-  I need a general-purpose model that can handle multiple tasks.
e.g., translation+entity recognition+sentiment analysis
-  I need to generate high-quality content, while preserving data privacy and security
e.g., abstractive summarization, content writing, paraphrasing, code
-  I need rapid prototyping and quick time to market for many use cases
-  I could use a model with little or no additional training
-  I want to explore solutions / use cases that have been described previously

Azure OpenAI Service



Chat Completions API

Versatile interface use for all scenarios—not just chat



Model adheres to instructions in “system” message

It sets the behavioral guidelines for the model, including responsible AI steering

Examples go into “User”, “Assistant” fields

```
import openai
openai.api_type = "azure"
:
response = openai.ChatCompletion.create(
    engine ="gpt-4",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"},
        {"role": "assistant", "content": "The LA Dodgers won the World Series in 2020."},
        {"role": "user", "content": "Where was it played?"}
    ]
)
```

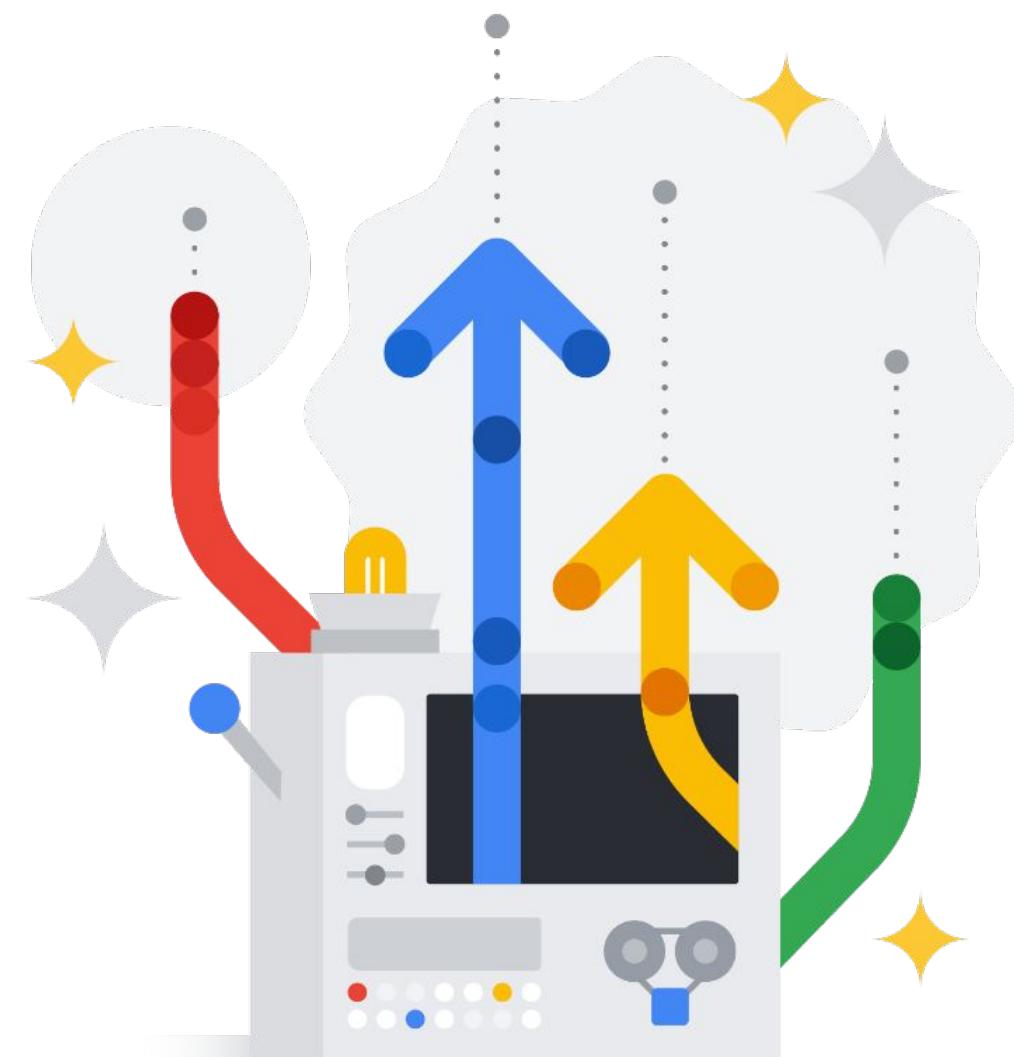
Vertex Model Garden

Choose a model that fits your needs

| Model Type | Model Description | Model Details |
|-------------|---|---|
| First Party | <p>Foundation models</p> <p>Leverage Google's multimodal models across vision, dialog, code generation/completion</p> | <ul style="list-style-type: none">• PaLM for Text PaLM for Chat• Imagen for text-to-image• Codey for code completion• Chirp for speech-to-text |
| | <p>Pre-trained APIs</p> <p>Build and deploy AI applications faster with our pre-trained APIs powered by the best Google AI research and technology</p> | <ul style="list-style-type: none">• Speech-to-Text• Natural Language Processing• Translation• Vision |
| Open Source | <p>Open Source</p> <p>Access a wide variety of enterprise-ready open source models</p> | <ul style="list-style-type: none">• StableDiffusion• ViT• EfficientNet |
| 3rd Party | <p>3rd Party</p> <p>Over time Model Garden will support 3rd-party models from partners with foundation models</p> | Coming soon |

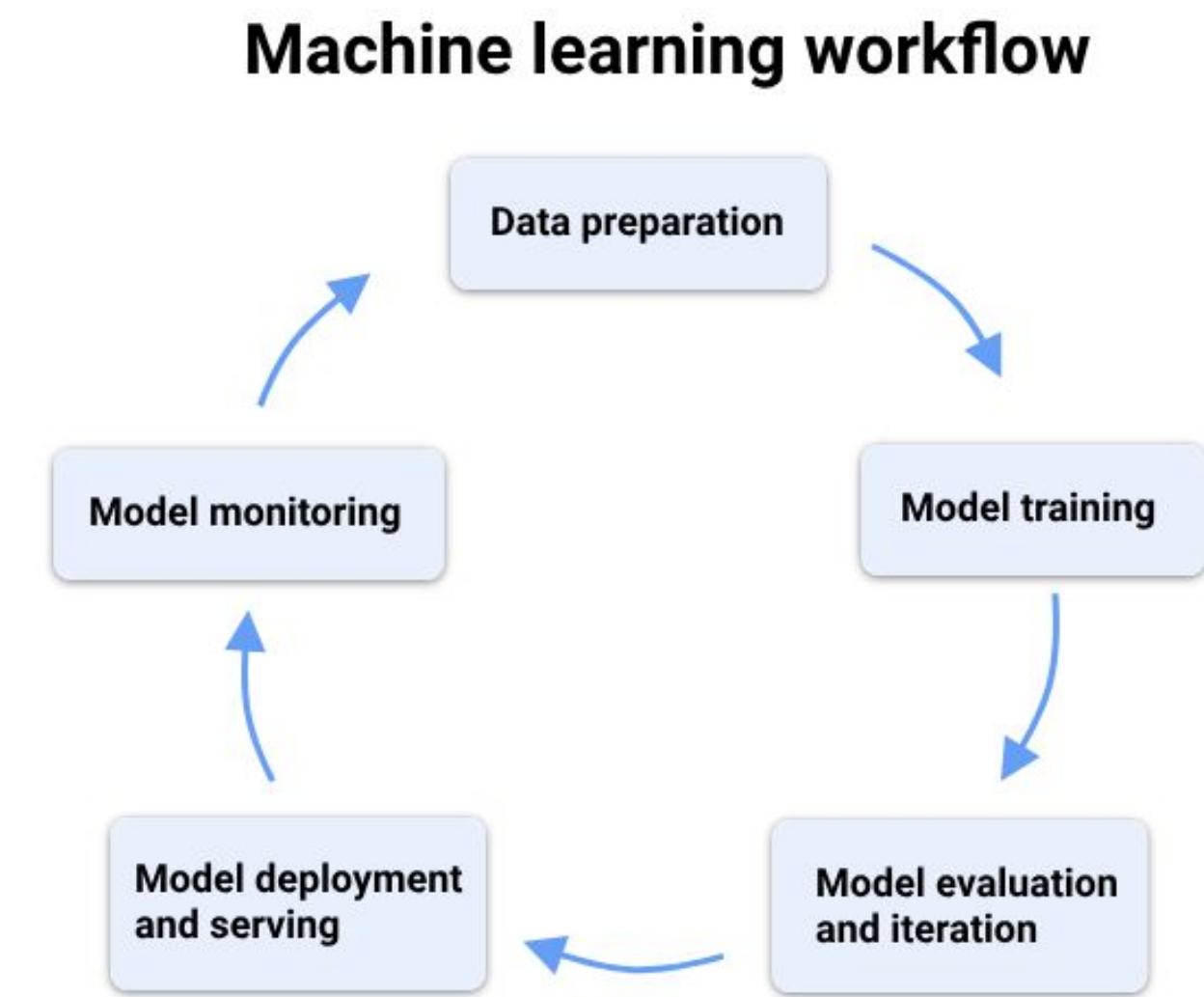
Generative AI Studio simplifies prompting, tuning, and deploying Google foundational models

- Low-code, customizable solution for building generative AI solutions
- Easy access to Google foundational models
 - PaLM 2 (Pathways Language Model) for text and chat
 - Codey for code generation, completion, and chat
 - Chirp for speech to text
 - Imagen for text to image generation



Azure AI helps with all steps in an AI workflow

- Data preparation: clean and explore data, apply transformations and feature engineering
- Model training: choose a training method, train and tune the model
- Model evaluation and iteration: evaluate the model, make adjustments and iterate
- Model serving: deploy the model to production and get predictions
- Model monitoring: monitor the performance of the deployed model and retrain as needed



Azure AI features by task

01

Data Preparation

Datasets
Labeling tasks
Feature store
AI Pipelines

02

Model Training

Auto ML
Model Garden
Gen AI Studio
Online Training
Workbench

03

Model Evaluation

Built-in metrics
Experiments

04

Model Deployment

AI Pipelines
Endpoints
Batch predictions

05

Model Monitoring

Operations Suite
Monitoring
Logging

Topics

- 01 What is Generative AI
- 02 Vertex AI on Google Cloud
- 03 Generative AI Options on Google Cloud
- 04 Introduction to the Course Use Case



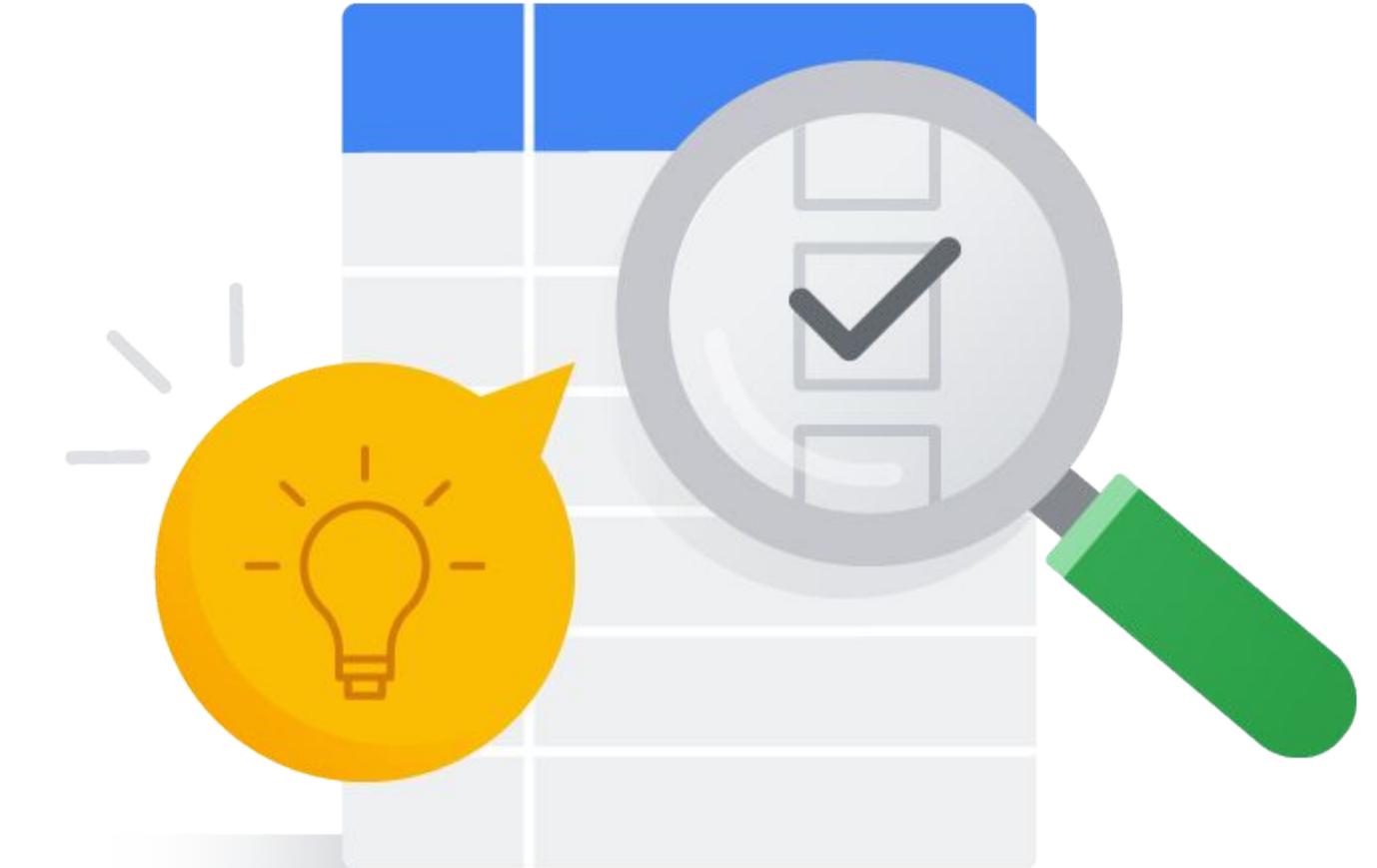
Group Discussion

10 min



Take a few minutes to come up with some use cases for generative AI that would benefit you or your organization

You will share your ideas with the class



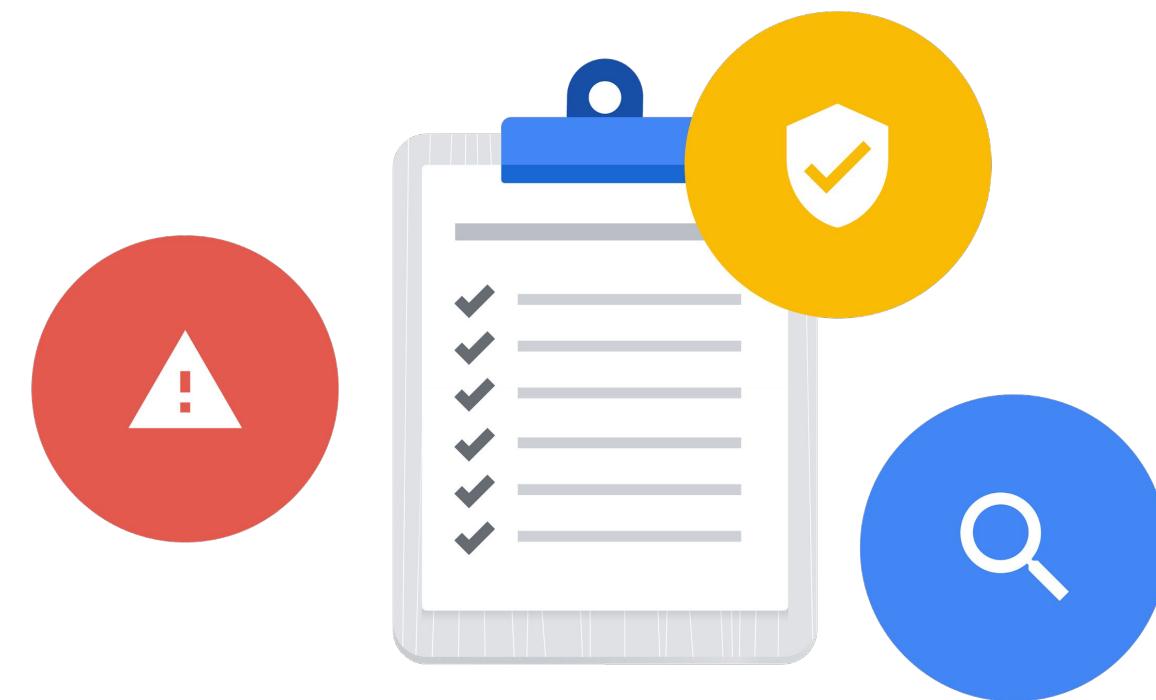
Topics

- 01 What is Generative AI
- 02 Vertex AI on Google Cloud
- 03 Generative AI Options on Google Cloud
- 04 Introduction to the Course Use Case

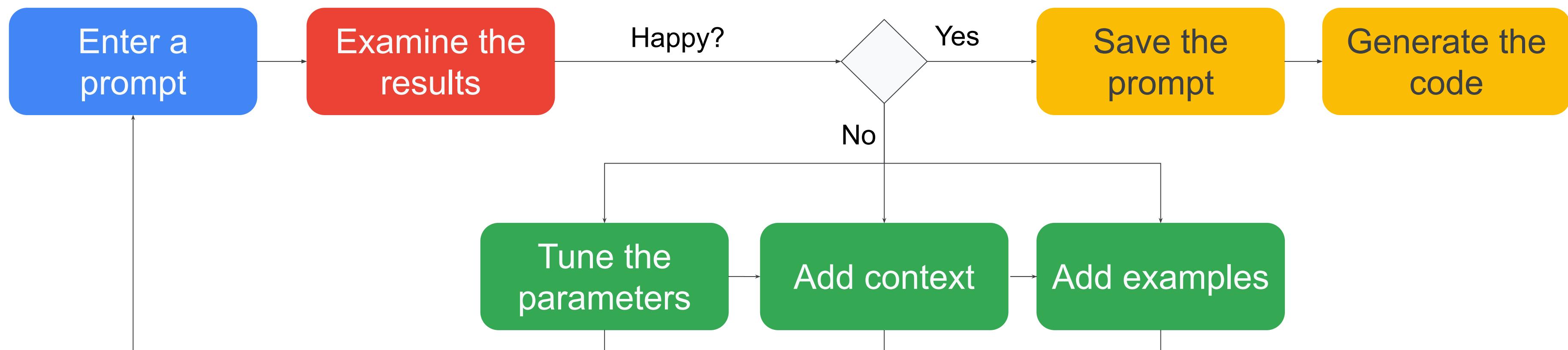


This course focuses on using generative AI for language-based content creation use cases

- Writing
- Summarization
- Ideation
- Classification
- Sentiment analysis
- Extraction



You will learn the process of designing, tuning, and deploying prompts to generate content for each use case



Topics

01

What is Generative AI

02

[Vertex AI on Google Cloud](#)

03

Generative AI Options on Google Cloud

04

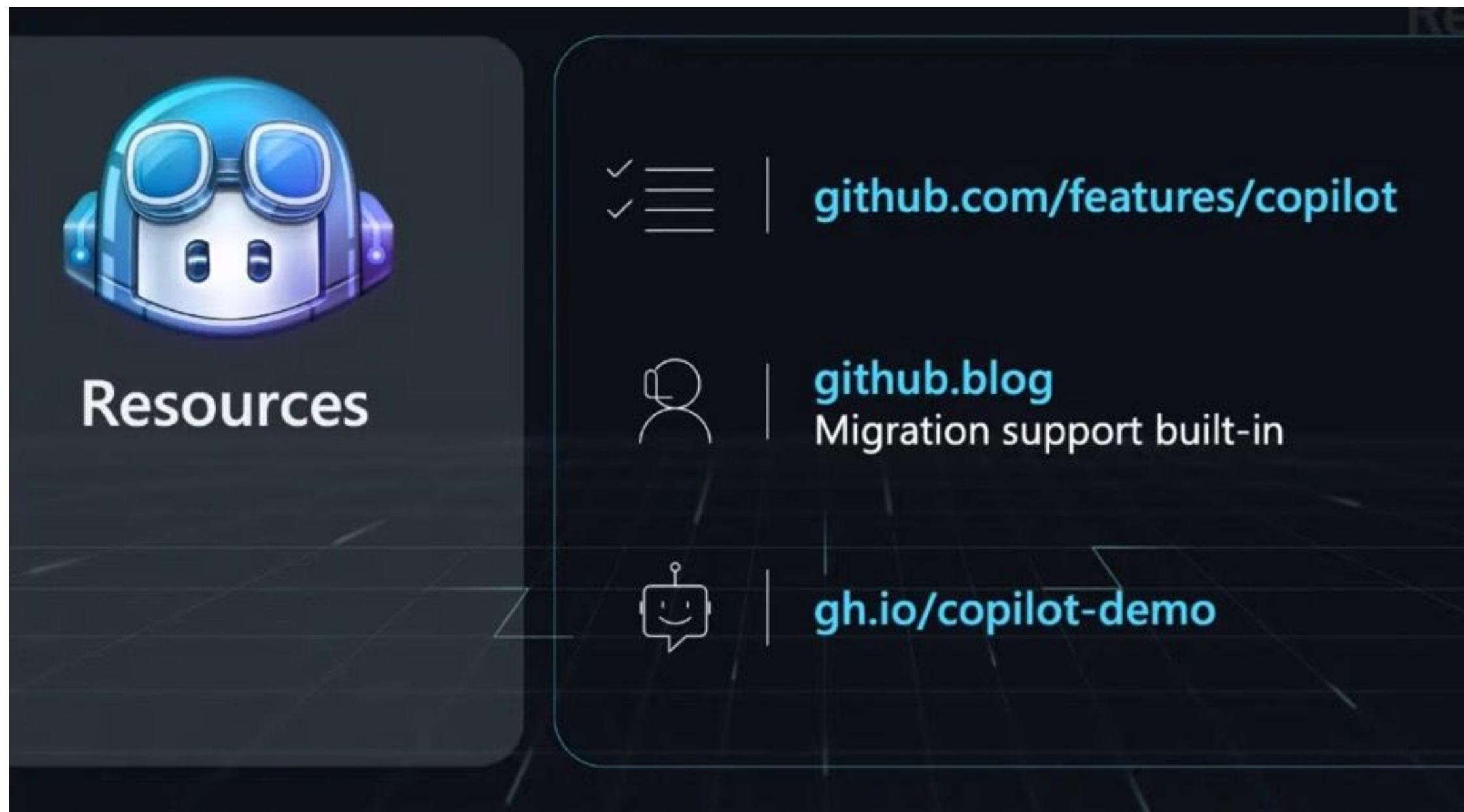
Introduction to the Course Use Case



Vertex AI is a machine learning platform that helps you build, deploy, and manage ML models

- Combines data engineering, data science, and ML engineering workflows
- Provides several options for model training and deployment
 - AutoML
 - Custom Training
 - Model Garden
 - Generative AI Studio
- Uses serverless infrastructure that you can customize based on your performance and budget needs
- Supports Python, the Google Cloud Console, the gcloud command line tool, client libraries, and Terraform





Google Vision API tells you what's in a picture

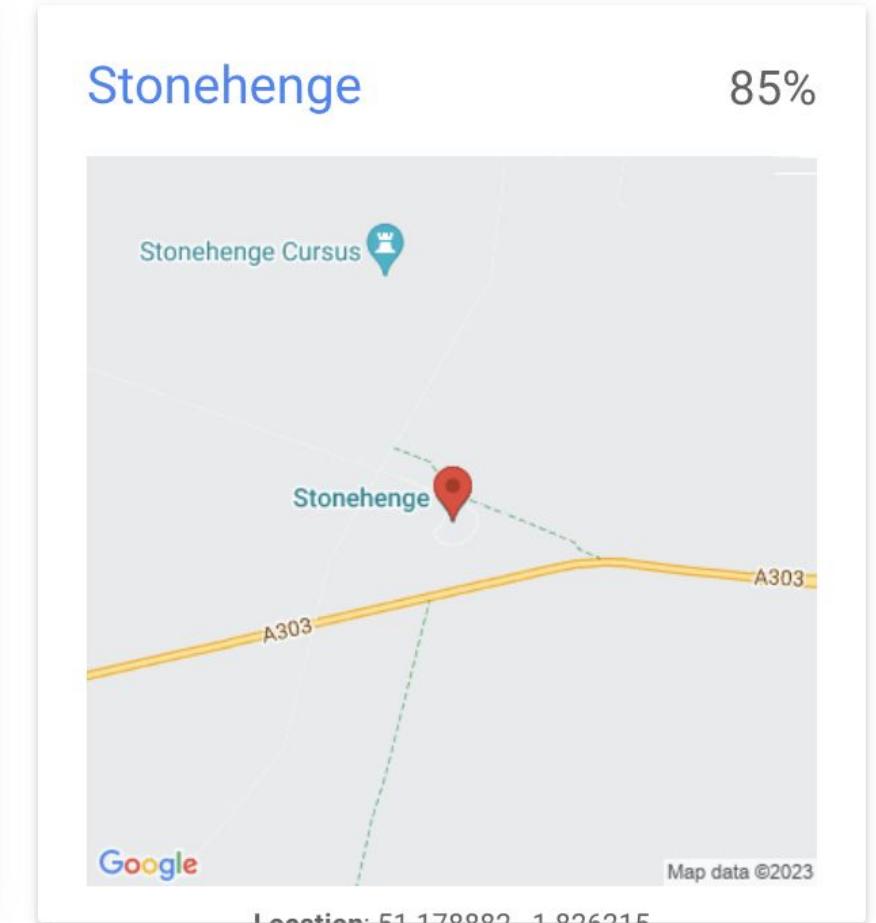
- Detects objects and landmarks
- Labels photos
- Recognizes text
- Detects inappropriate content

Landmarks Labels Text Properties Safe Search



IMG_20180218_131834.jpg

Stonehenge 85%



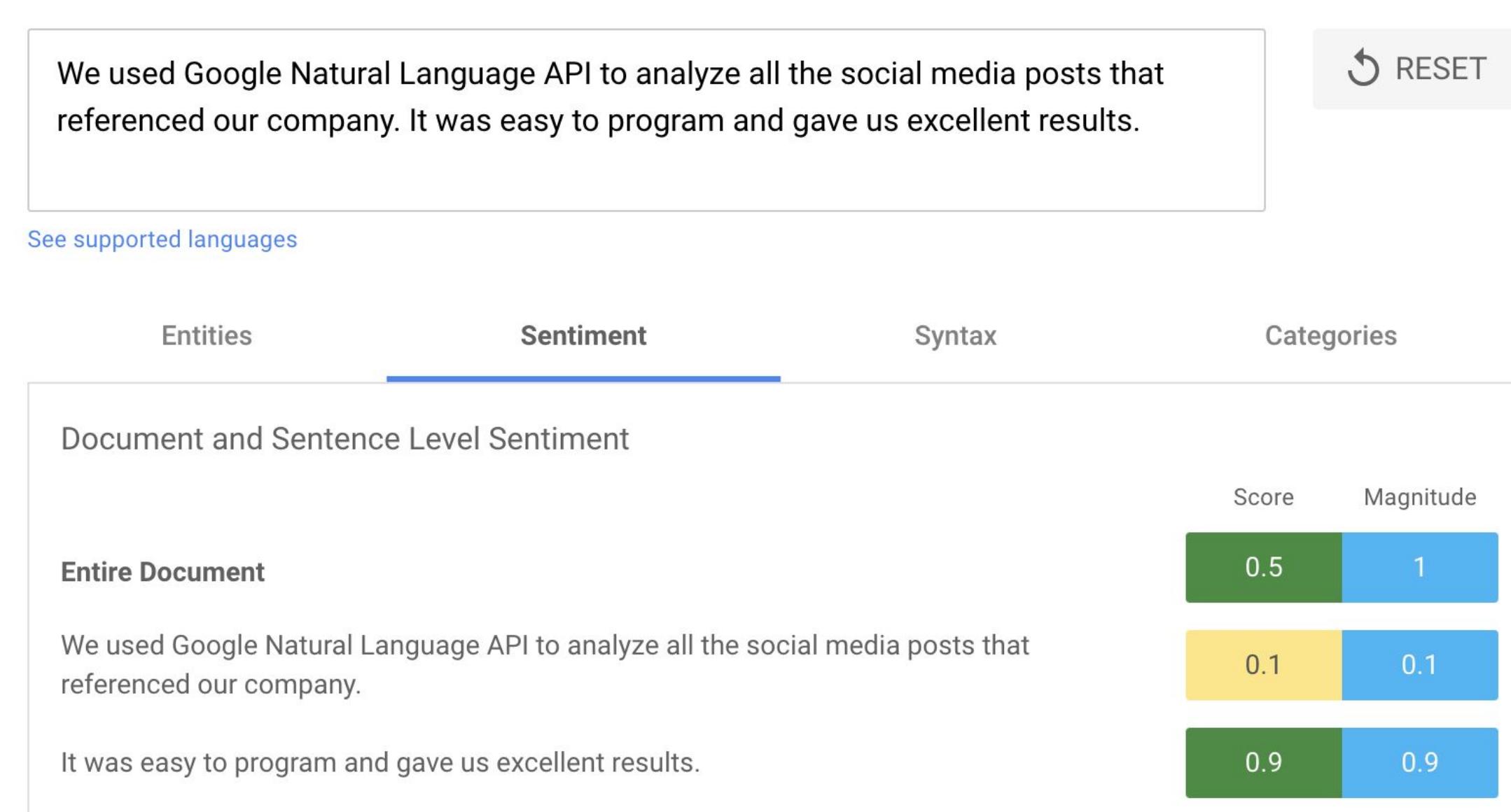
<https://cloud.google.com/vision/docs/drag-and-drop>

You can use Google ML to translate between different languages

The screenshot shows the Google Translate web interface. At the top, there's a navigation bar with the Google Translate logo, a menu icon, and a user profile icon. Below the logo are four tabs: Text (selected), Images, Documents, and Websites. Underneath these tabs, there are language selection dropdowns. The left side shows the source text in English: "Google Translate is a multilingual neural machine translation service developed by Google to translate text, documents and websites from one language into another." The right side shows the translated text in Spanish: "Google Translate es un servicio de traducción automática neuronal multilingüe desarrollado por Google para traducir texto, documentos y sitios web de un idioma a otro." Below the text boxes are various interaction icons like microphone, speaker, and share.

<https://translate.google.com/>

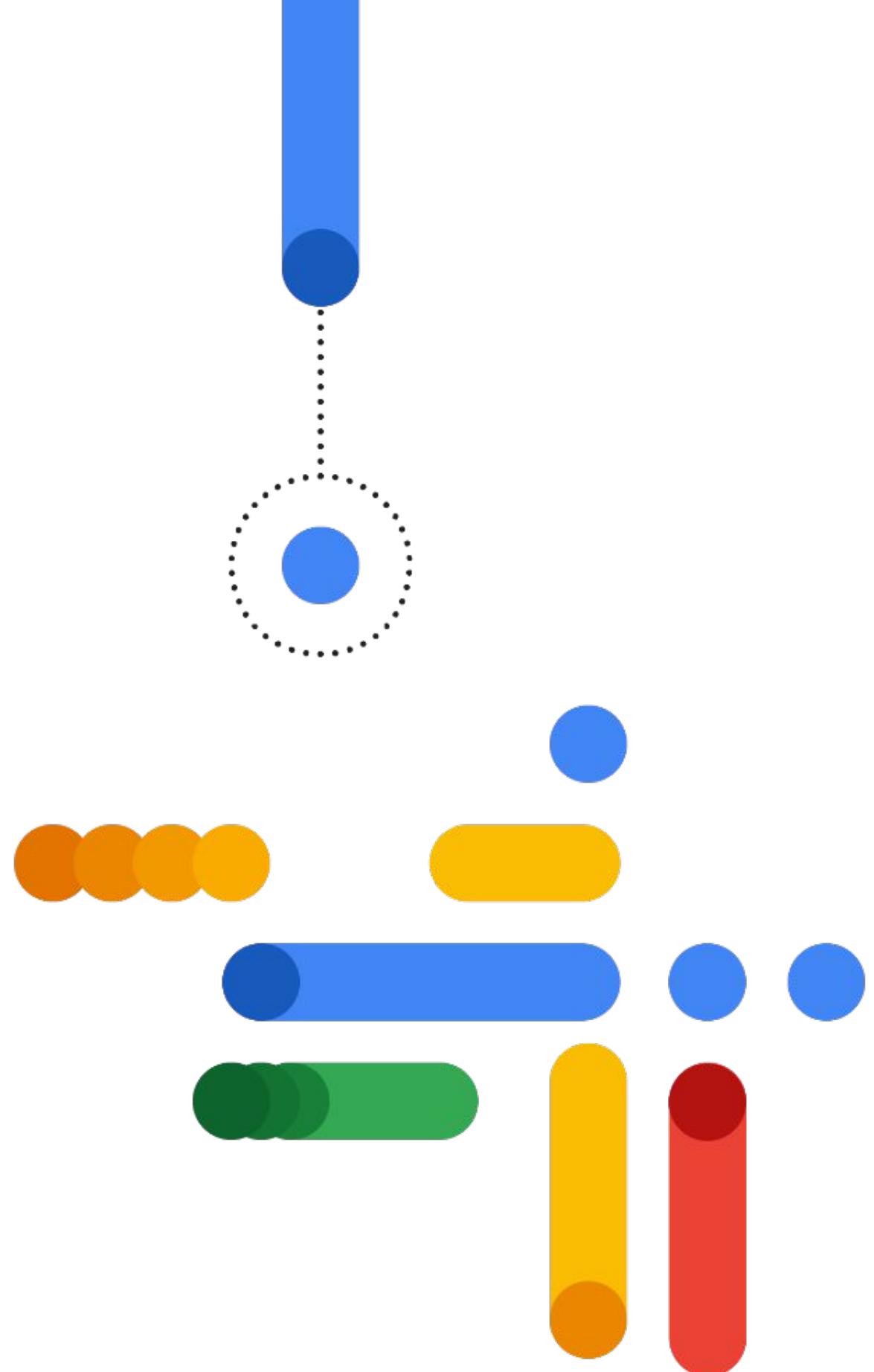
Natural Language API can analyze text, extract entities, and perform sentiment analysis



<https://cloud.google.com/natural-language/>

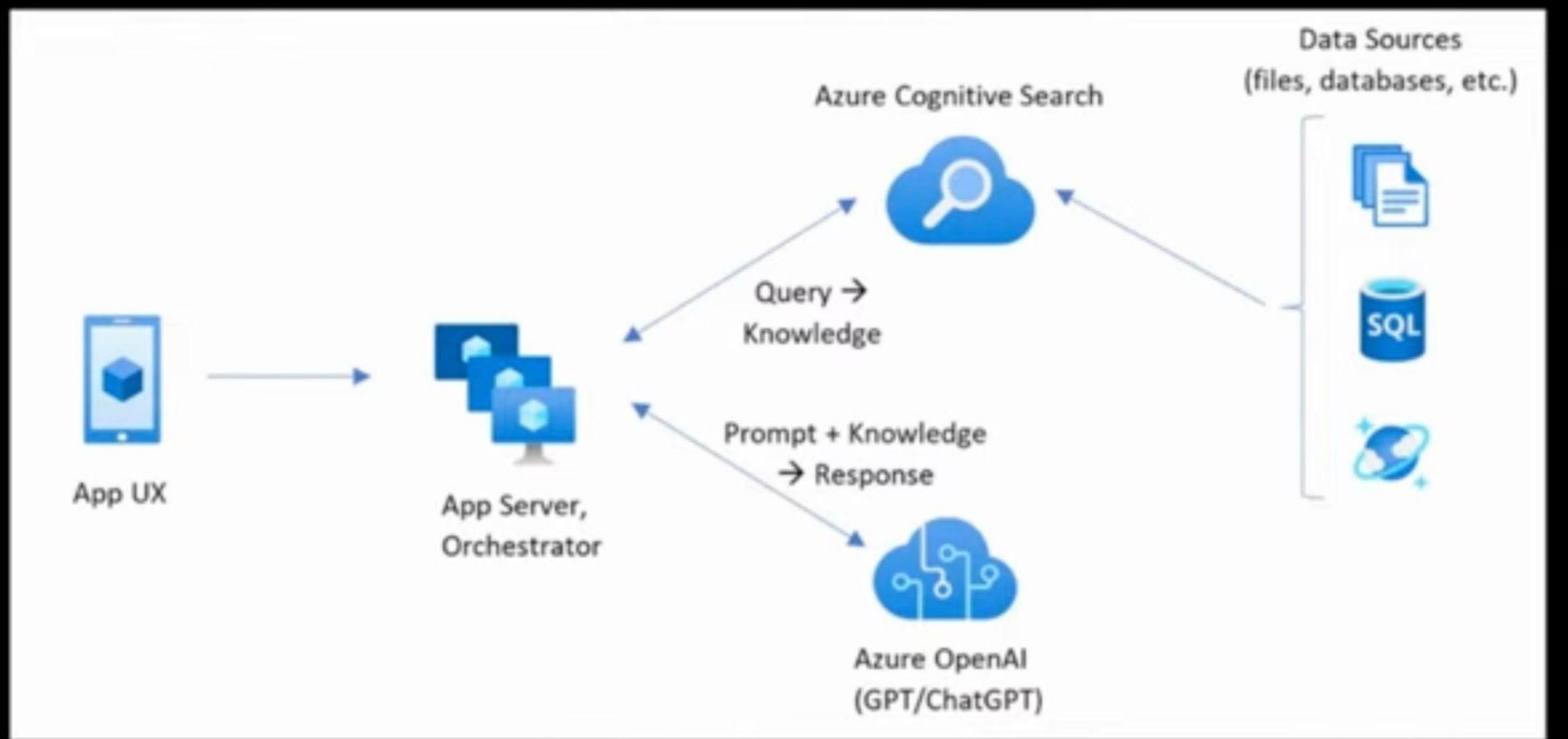
Vertex AI simplifies model training with AutoML

- Supports datasets created with image, text, tabular, and video data
- Zero-code, custom-model training for various use cases
 - Image detection and classification
 - Text classification, entity extraction, and sentiment analysis
 - Linear regression, classification, and forecasting from tabular datasets
 - Video action recognition and object tracking
- Automated deployment of models to service endpoints managed by Google



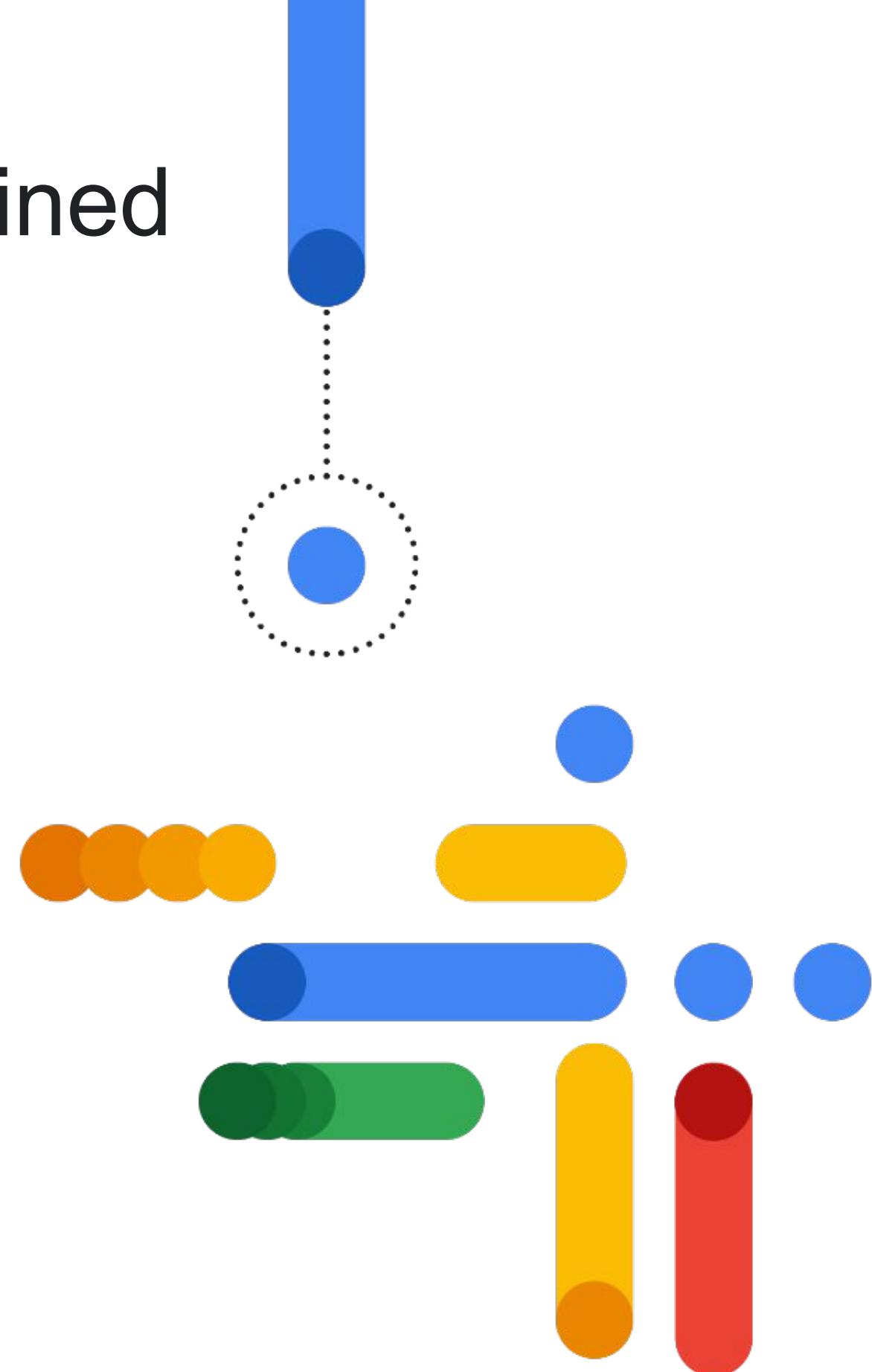
GPT with your data (Enterprise ChatGPT)

Extract rich insights, reason over and use Chat over your internal data

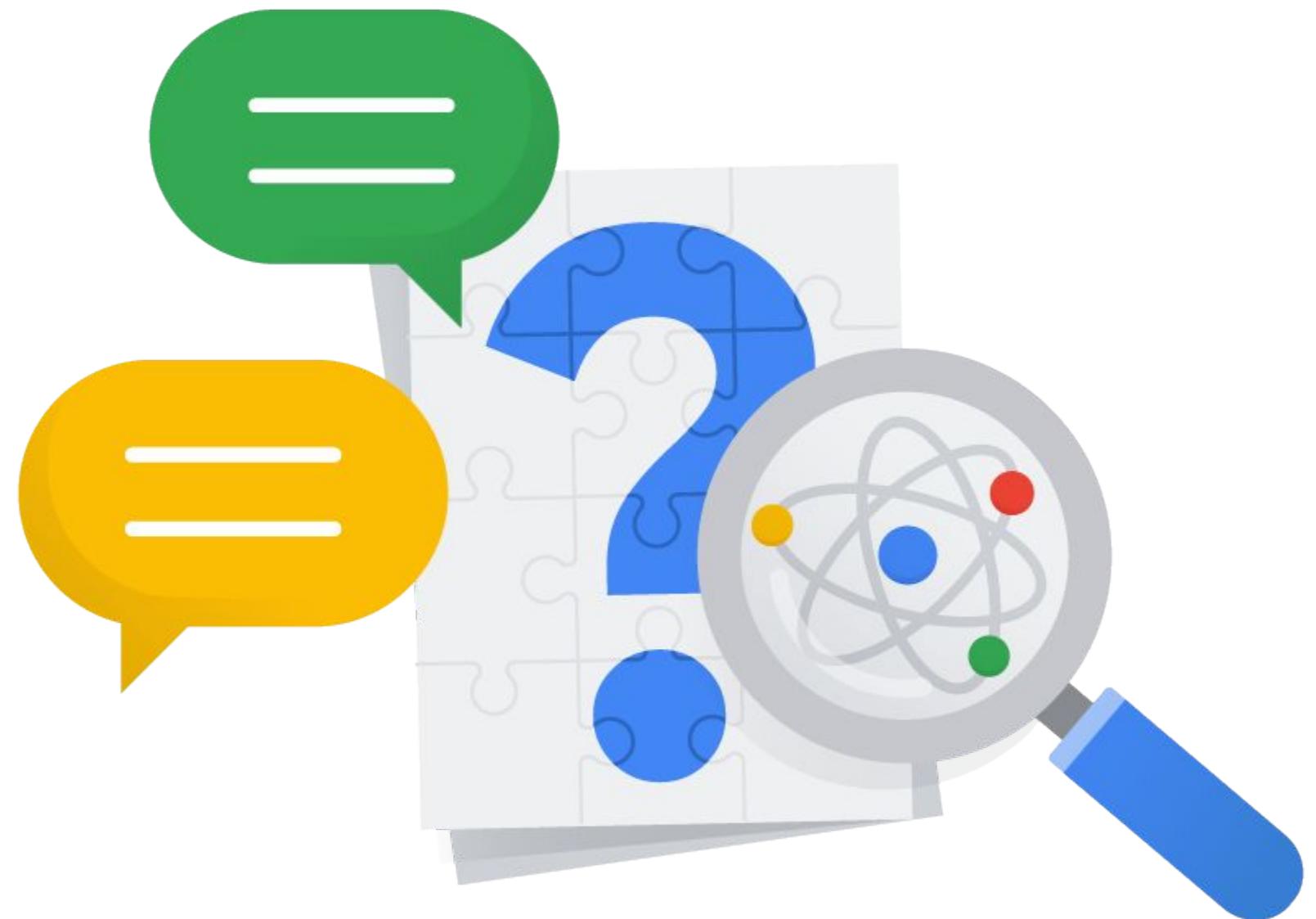


Model Garden provides a catalog of pre-trained models to build AI applications

- Foundation models are pre-trained multitask large models that can be tuned or customized for specific tasks
- Fine-tunable models are models that you can fine-tune using a custom notebook or pipeline
- Task-specific solutions are pre-built models that are ready to use and can be customized using your own data
- Model garden provides and easy to use interface for searching for models
 - Documentation and code samples are provided



Questions and answers



In this module, you learned to ...

- 01 Differentiate between machine learning in general and generative AI
- 02 Automate ML tasks using Vertex AI on Google Cloud
- 03 Choose from the available Generative AI options on Google Cloud
- 04 Explore the text generation AI use case



Quiz question

What is a type of artificial intelligence that allows computers to learn based on data without having to be explicitly programmed?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

LANGUAGE MODEL SIZES TO DEC/2022

- BERT 340M
- GPT-1 117M
- GPT-2 1.5B

T5

Megatron-11B

Plato-XL 11B

Macaw 11B

Cohere

52.4B

GPT-3 175B

Jurassic-1 178B

LaMDA 137B

GPT-J 6B

BlenderBot2.0 9.4B

MT-NLG
530B

Cedille 6B

Fairseq 13B

Anthropic-LM

52B

Gopher
280B

GPT-NeoX-20B 20B

Luminous
200B

CM3 13B
VLM-4 10B
mGPT 13B

BLOOM
176B

Atlas 11B

NLLB 54.5B

GLM-130B

OPT-175B
BB3 175B

UL2 20B

YaLM
100B

GAL 120B

PaLI 17B

Z-Code++ 710M

AlexaTM

VIMA 200M

10B

Parameters

AI lab/group

Available

Closed

Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. Alan D. Thompson, December 2022. <https://larchitect.ai/>



LifeArchitect.ai/models

Quiz question

What is a type of artificial intelligence that allows computers to learn based on data without having to be explicitly programmed?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

What is a type of artificial intelligence that can create new content, such as text, images, or music?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

What is a type of artificial intelligence that can create new content, such as text, images, or music?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

When generative AI systems just make something up that isn't true, it is called what?

- A: A lie
- B: A hallucination
- C: A bug
- D: It is not possible for that to happen