Microsoft

Kickoff workshop - Azure OpenAI Community Champs

Franck Gaillard – Cloud Solution Architect AI

Frederic Gisbert - Cloud Solution Architect Data Analytics

Narjes Majdoub - Cloud Solution Architect AI

14.06.2023

Agenda

1	Introduction to AOAI community champs program
2	Introduction to AOAI
3	Enterprise AOAI
4	Introduction to Langchain
5	Embeddings and Vector DBs
6	Deep Dive – Architecture - AOAI Embeddings QnA

AOAI* = Azure Open AI service

1

Introduction to AOAI community champs program

Share your insights, feedback, and best practices with other community members and Microsoft engineers.
Share your insights, feedback, and best practices with other community members and Microsoft engineers.
Share your insights, feedback, and best practices with other community members and Microsoft engineers.

Community Champs Program – Give & Get

- A community of experts on Azure OpenAI
- You are the referents within your company on Azure OpenAI topics
- Share your insights, feedback, and best practices with other community members and Microsoft engineers
- Discuss customer use cases
- Meetup, hackathon…
- Suggest what you'd like to see in the next sessions
- Other ideas?

2

Introduction to AOAI

Our partnership with OpenAI

OpenAI

Ensure that artificial general intelligence (AGI) benefits humanity.

+

Microsoft

Empower every person and organization on the planet to achieve more

2019

Azure OpenAI Service

GPT-4

DALL·E

ChatGPT

Deployed in your Azure subscription, secured by you, and tied to your datasets and applications

Large, pretrained AI models to unlock new scenarios

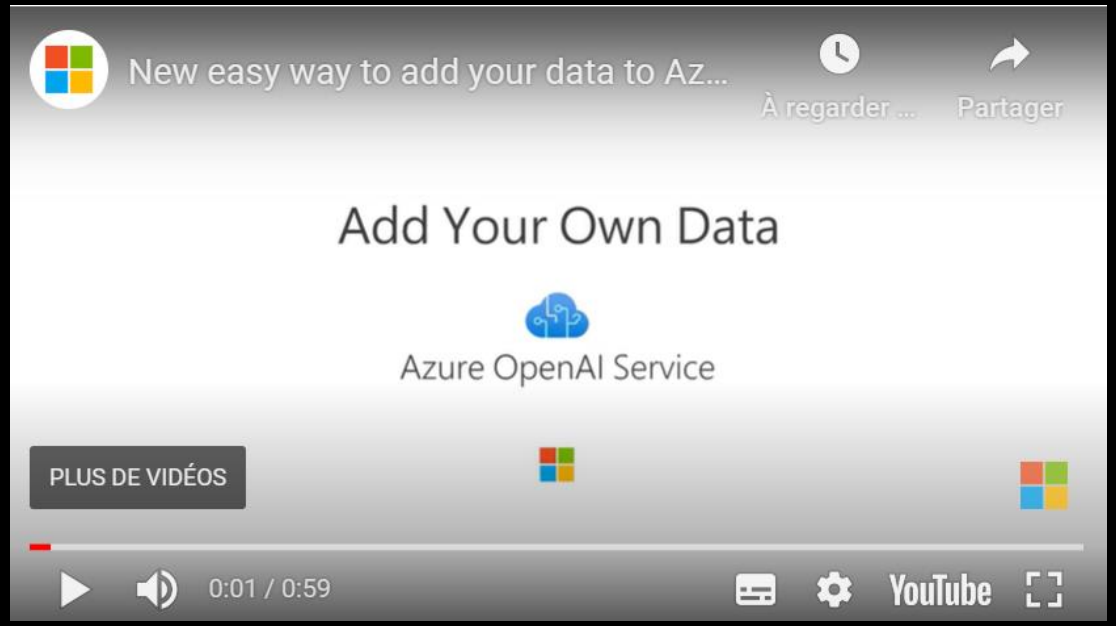
AI models, some custom-tunable with your data and hyperparameters

Built-in responsible AI to detect and mitigate harmful use

Enterprise-grade security with role-based access control (RBAC) and private networks

Azure OpenAI Service on your data

- With just a few clicks, developers can now ground powerful conversational AI models, such as OpenAI's ChatGPT and GPT-4, on their own data.
- With Azure OpenAI Service on your data, coming to public preview, and Azure Cognitive Search, employees, customers, and partners can discover information buried in the volumes of data, text, and images using natural language-based app interfaces.

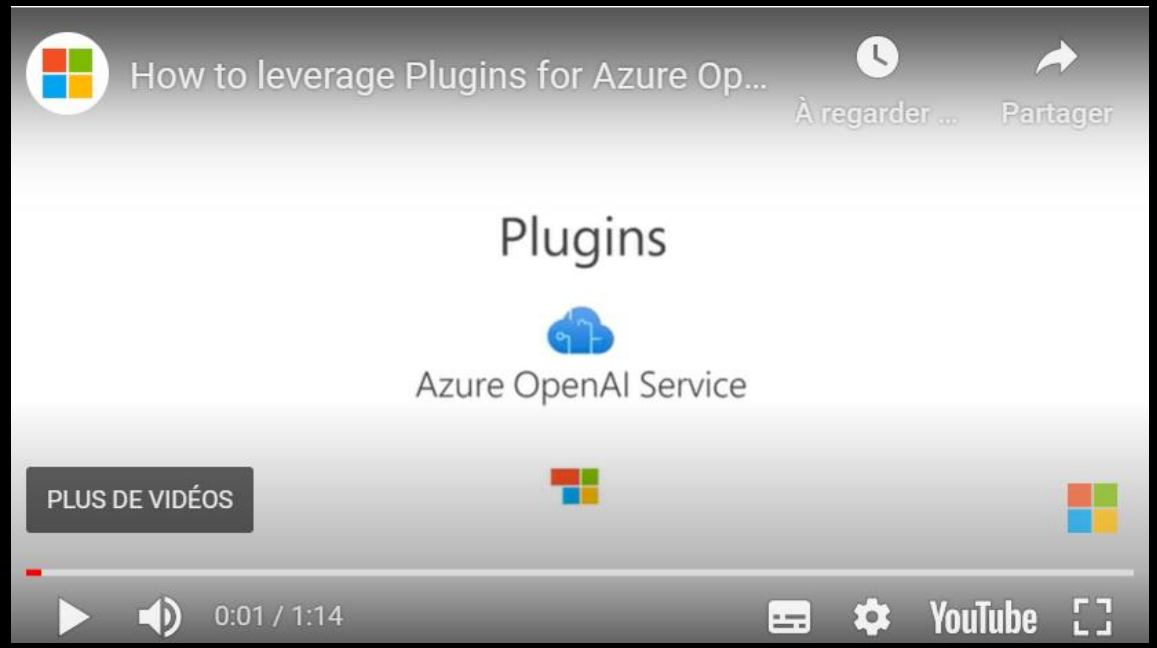


Azure OpenAI Service Plugins

Plugins are a standardized interface that allows developers to build and consume APIs to extend the capabilities of large language models (LLMs) and enable a deep integration of GPT-4 across Azure and the Microsoft ecosystem.

In the limited preview coming to developers in July, the following plugins are included:

- Bing Search
- Azure Cognitive Search
- Azure SQL
- Azure Cosmos DB
- Microsoft Translator



Although only prebuilt plugins will be available in the preview, soon customers will soon be able to use their own plugins with Azure OpenAI Service too. We'll follow the same standards for building plugins as OpenAI, so plugins will be fully interoperable between the two platforms.

Azure OpenAI Service Other News

Provisioned Throughput Model

- The provisioned throughput feature is a new offering coming to Azure OpenAI Service that allows customers to have more control over the configuration and performance of OpenAI's large language models at scale. It provides a dedicated connection to OpenAI models with a guaranteed throughput, **measured in tokens/sec** for prompts and completions. The feature will be generally available with limited access in June.

Quotas

- Azure OpenAI Service now includes quotas to prevent overuse and ensure fair usage across all customers. Quotas can be set at the resource group level, with separate quotas for prompts, completions, and training. Customers can view their current usage and remaining quota in the Azure portal.

Azure OpenAI Service Other News

Configurable Content Filters

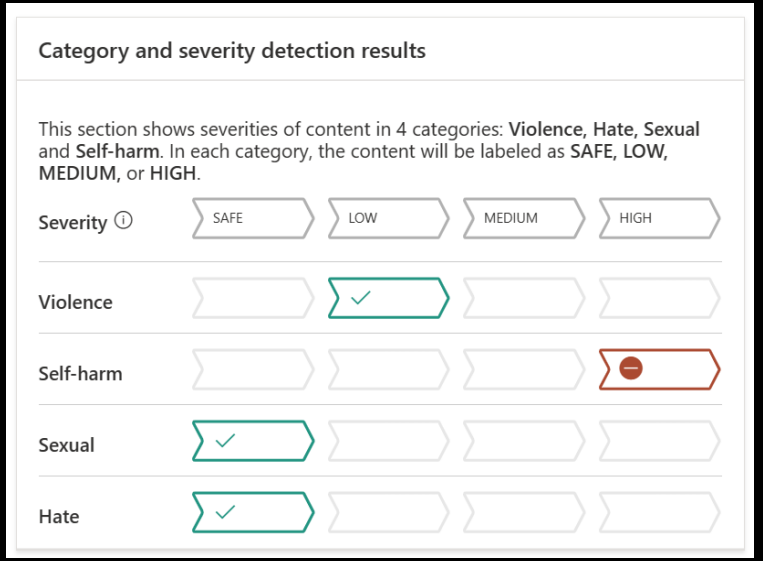
- Configurable content filters allow customers to specify a list of banned or allowed words, phrases, and entities. These filters are applied to all text prompts and completions and help ensure that the output is appropriate for their intended audience. The feature is fully customizable, with customers able to specify their own filters or use the default filters provided by OpenAI.

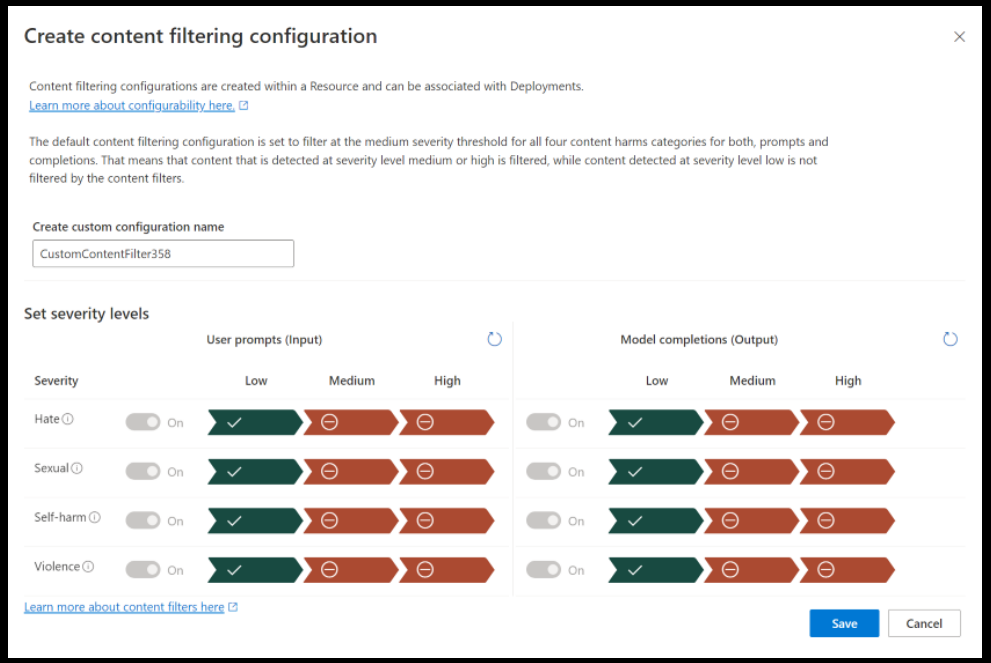
Our Commitment to Responsible AI

- At Microsoft, we're committed to the advancement of AI driven by principles that put people first. Generative models such as the ones available in Azure OpenAI Service have significant potential benefits, but without careful design and thoughtful mitigations, such models have the potential to generate incorrect or even harmful content. Microsoft has made significant investments to help guard against abuse and unintended harm, which includes requiring applicants to show well-defined use cases, incorporating Microsoft's principles for responsible AI use, building content filters to support customers, and providing responsible AI implementation guidance to onboarded customers.

Azure AI Content Filtering

- The content filtering system integrated into Azure OpenAI Service runs alongside the core models and uses an ensemble of multi-class classification models to detect four categories of harmful content (violence, hate, sexual, and self-harm) at four severity levels respectively (safe, low, medium, and high).





Foundation Models in Azure Machine Learning

- Public preview of foundation models in Azure Machine Learning, which empowers users to discover, customize and operationalize large foundation models at scale through the model catalog.
- It offers a collection of Open Source models curated by AzureML, a Hugging Face hub community partner collection, and a collection of Azure OpenAI Service models. With this new capability, our customers can easily access the latest foundation models and accelerate the use of these models for fine-tuning, evaluation, deployment and operationalization in their own specific workloads.
- The foundation models in AzureML include optimizations like Deepspeed and ORT (ONNX RunTime), which speed up fine-tuning, and LoRA (Low-Rank Adaptation of Large Language Models), which greatly reduces memory and compute requirements for fine-tuning.

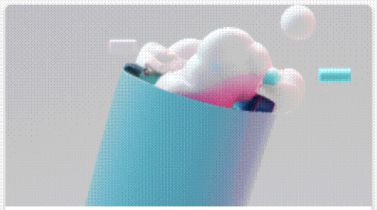
All workspaces
Home
Model catalog PREVIEW
Authoring
Notebooks
Automated ML
Designer
Prompt flow PREVIEW
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Linked Services
Data Labeling

mlw-contoso-819prod

New | Customize view

View prompt flow

Generative AI with Prompt flow PREVIEW



Ask Wikipedia

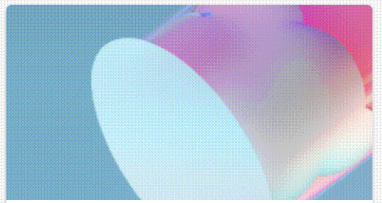
Q&A with GPT3.5 using information from wikipedia to make the answer more grounded.

Start

Bring Your Own Data QnA

Q&A with GPT3.5 using data from your own indexed files to make the answer more grounded.

Start



Web Classification

Using large language models to classify URLs into multiple categories.

Start

Generative AI models PREVIEW

View all

gpt-4-32k
Chat completions

gpt-4
Chat completions

gpt-35-turbo
Chat completions

Notebook samples

View all

Get started: Train and deploy a model

Train and deploy a sample image classification model.

Start 25 minutes

Arbitration agents using LangChain

Demonstration of LangChain agent to implement the ReAct logic using Azure OpenAI endpoints.

Start 20 minutes

Index and search your own data with GPT

Bring your own data to look up using GPT with LangChain

Start 20 minutes

Distributed GPU training

Run a sample multi-GPU image classification experiment.

Start 30 minutes

Automat

Create a p
credit defa

Start

Shortcuts

Create notebook

Add compute

Connect data

Train a model

Deploy a

Azure Machine Learning prompt flow

- Prompt Engineering plays a crucial role in harnessing the full potential of LLMs by creating effective prompts that cater to specific business scenarios. This process enables developers to create tailored AI solutions, making AI more accessible and useful to a broader audience.
- Prompt Engineering, an essential process for generating high-quality content using LLMs, remains an iterative and challenging task. This process involves several steps, including data preparation, crafting tailored prompts, executing prompts using the LLM API, and refining the generated content. These steps form a flow that users iterate on to fine-tune their prompts and achieve the best possible content for their business scenario.

All workspaces

Home
Model catalog PREVIEW

Authoring

Notebooks
Automated ML
Designer
Prompt flow PREVIEW

Assets

Data
Jobs
Components
Pipelines
Environments
Models
Endpoints

Manage

Compute
Monitoring PREVIEW
Data Labeling
Linked Services

Vector DB QnA Step 1

Runtime * No runtime available | Add runtime | Bulk test | Run

LLM | Prompt | Python | More tools

Inputs

Fill value from data

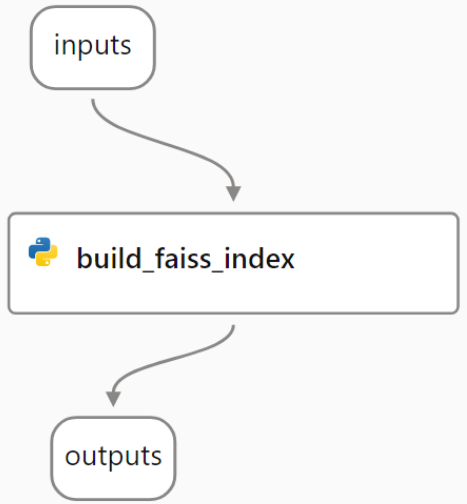
Name	Type	Value	Show description
blob_store_path	string	embeddingstore/vector_db_qna_example/faiss_index_store	

Add Input

Outputs

Name	Value
blob_store_url	${build_faiss_index.output}

Add Output



build_faiss_index

python

Code | Wrap text | Diff mode

```python
import os
import shutil
from typing import List
import urllib.request
from bs4 import BeautifulSoup
from langchain.text_splitter import RecursiveCharacterTextSplitter

from embeddingstore.core.utils import CommonUtils
from embeddingstore.core.contracts import (
    EmbeddingModelType,
    StorageType,
    StoreCoreConfig,
)
from embeddingstore.tool.utils.store_core_provider import StoreCoreProvider
from promptflow.connections import AzureOpenAIConnection
from promptflow import tool
```

Azure Cognitive Search News

- Vector search for Azure Cognitive Search, the retrieval system for new large language models (LLM) apps, is coming soon in preview.
- Vector search allows developers to easily store, index and search by concept in addition to keywords, using organizational data including text, images, audio, video and graphs.
- Developers will be able to build apps to generate personalized responses in natural language, deliver product recommendations, detect fraud, identify data patterns and more. Azure Cognitive Search offers pure vector search and hybrid retrieval – as well as a sophisticated re-ranking system powered by Bing in a single integrated solution. Sign up for the Vector search preview today.

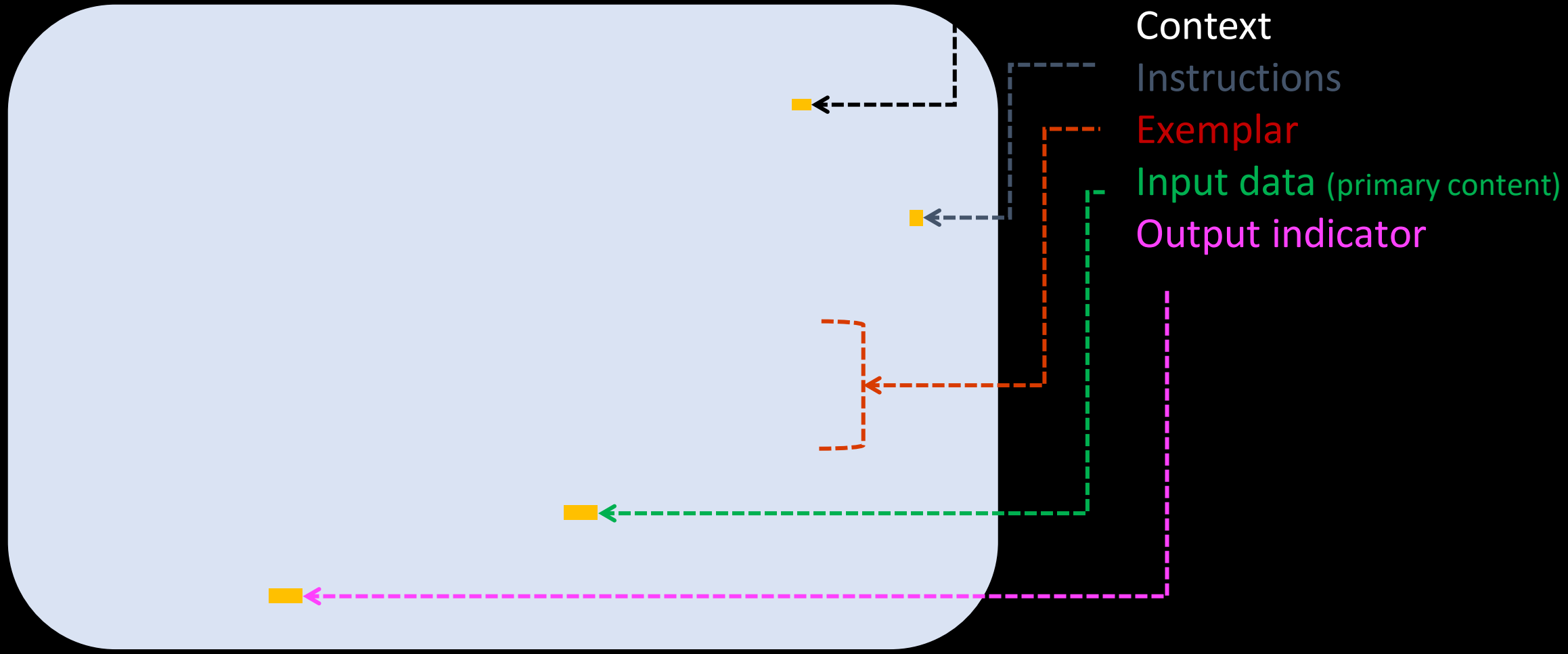
3

Enterprise AOAI

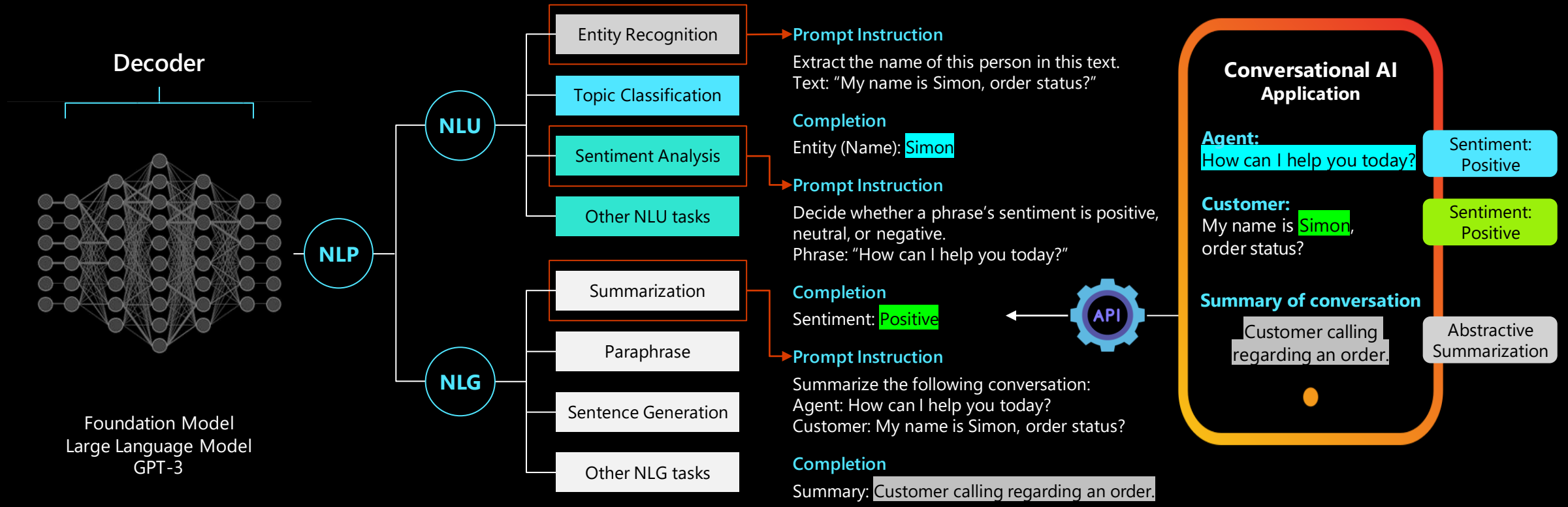
How to build my GPT Use Case ?

> My instructions *(prompt)* 
> My examples *(shot learning vs fine-tuning)*
> My data *(private database vs public)*

> My instructions is given by the prompt



Model use out of the box—prompting



Reduce Hallucination using Prompt Engineering

Include	Restrict	Add	Repeat	Position
Include instructions of requesting the model not to make up stuff but stay with facts.	Restrict the output (e.g., choose from a confined list instead of generating free form strings)	Add Chain of Thought style of instruction, "Solve the problem step by step."	Repeat most important instructions in the prompt a couple of times.	Position most important instructions in the last making use of latency effect.

How to build my Use Case ?

> My instructions *(prompt)*
> My examples *(shot learning vs fine-tuning)*
> My data *(private database vs public)*

> My examples adapt GPT model for your task

- **1st option :** learn by giving my examples within the prompt

- **2nd option :** build a dataset based on your examples for fine-tuning

Advanced techniques : Chain of thought (CoT / ToT) – Program-aided Language Model (PAL) – Reason + Act (ReAct) – Reprompting -…….

[2305.09993] Reprompting: Automated Chain-of-Thought Prompt Inference Through Gibbs Sampling (arxiv.org)

Luyu Gao, etc., PAL: Program-aided Language Models, arXiv 11/18/2022, arXiv:2211.10435

Shunyu Yao, etc., ReAct: Synergizing Reasoning and Acting in Language Models, arXiv 10/06/2022, arXiv:2210.03629

> My examples adapt GPT model for your task

1st option : learn by giving my examples within the prompt

Zero-shot - Predicting with no sample provided

One-shot - Predicting with one sample provided

Few-shot – Predicting with a few samples provided

Zero-shot
The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:      <--- task description
2  Cheese =>                         <--- prompt
```

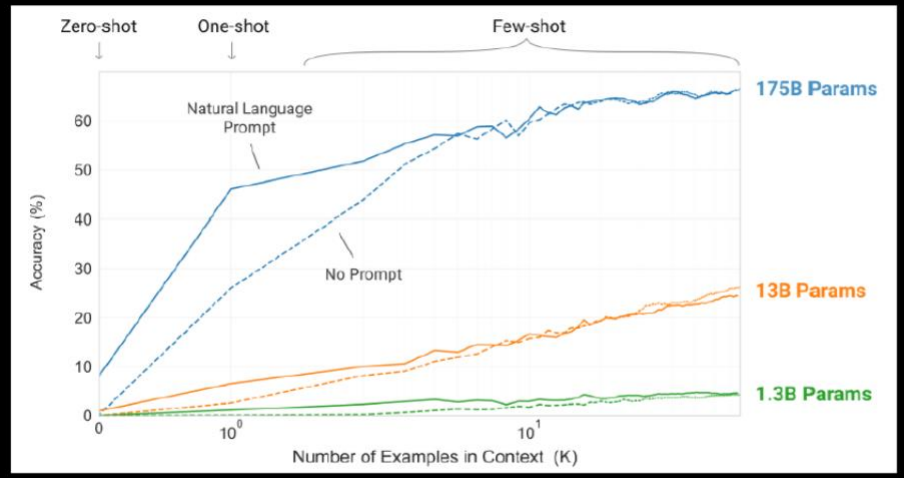
One-shot
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:      <--- task description
2  Sea otter => loutre de mer        <--- example
3  Cheese =>                         <--- prompt
```

Few-shot
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed

```
1  Translate English to French:      <--- task description
2  Sea otter => loutre de mer        <---
3  Peppermint => menthe poivre       <--- examples
4  Plush giraffe => girafe peluche   <---
5  Cheese =>                         <--- prompt
```

Larger models make increasingly efficient use of in-context information



These "learning" curves involve no gradient updates or fine-tuning, just increasing numbers of demonstrations given as conditioning

> My examples adapt GPT model for your task

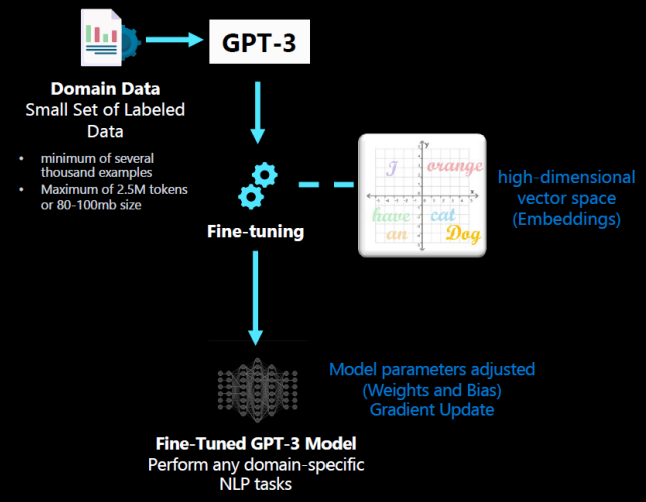
2nd option : build a dataset based on your examples for fine-tuning

Fine Tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

Prepare and upload training data → Train a new fined tuned model → Use your fine-tuned model

Fine-Tuning



Fine-tuning results is a new model being generated with updated weights and biases.

This is in contrast to few-shot learning in which model weights and biases are not updated.

> My examples adapt GPT model for your task

Training data is how you teach GPT-3 what you'd like it to say.

Your data must be a JSONL document, where each line is a prompt-completion pair corresponding to a training example.

To fine-tune a model, you'll need a set of training examples that each consist of a **single input ("prompt")** and its **associated output ("completion").** This is notably different from using our base models, where you might input detailed instructions or multiple examples in a single prompt.

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"} ...
```

> Fine-tuning (or not)

Can

- Fine-tuning teaches your model the structure and semantics of how to generate a completion for a given prompt.
- It speeds up response times because the model knows exactly what structure to produce for a given prompt.
- Prompts no longer require metaprompts or prompt engineering

Can’t

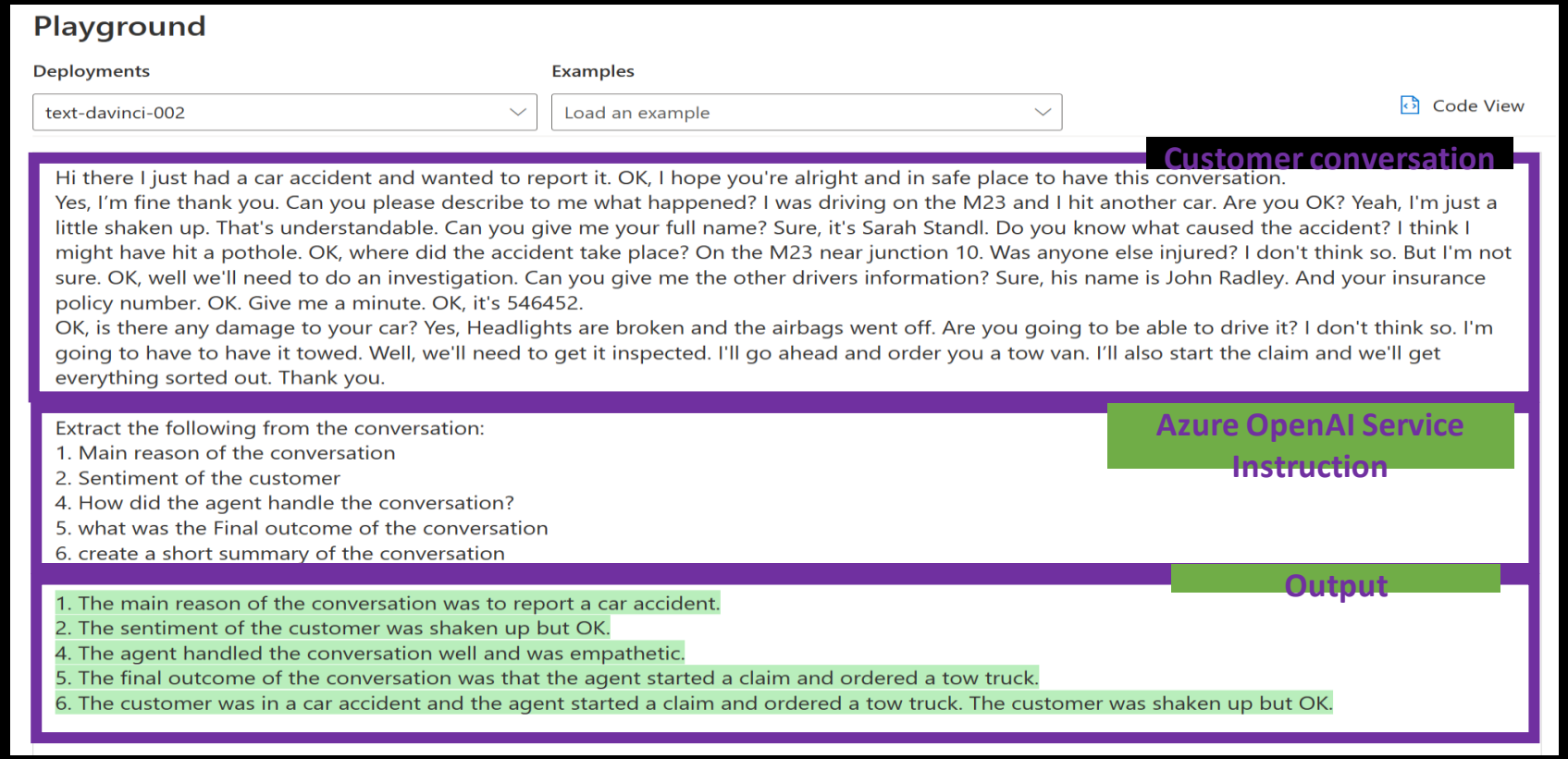
- Fine-tuning requires prompt-completion pairs and does not redo the pretraining phase of the model. **In other words, you can't "feed it your documents" to learn your content**.

- Fine-tuning does not teach the model about your facts or your data

How to build my Use Case ?

> My instructions *(prompt)*
> My examples *(shot learning vs fine-tuning)*
> My data *(private database vs public)*

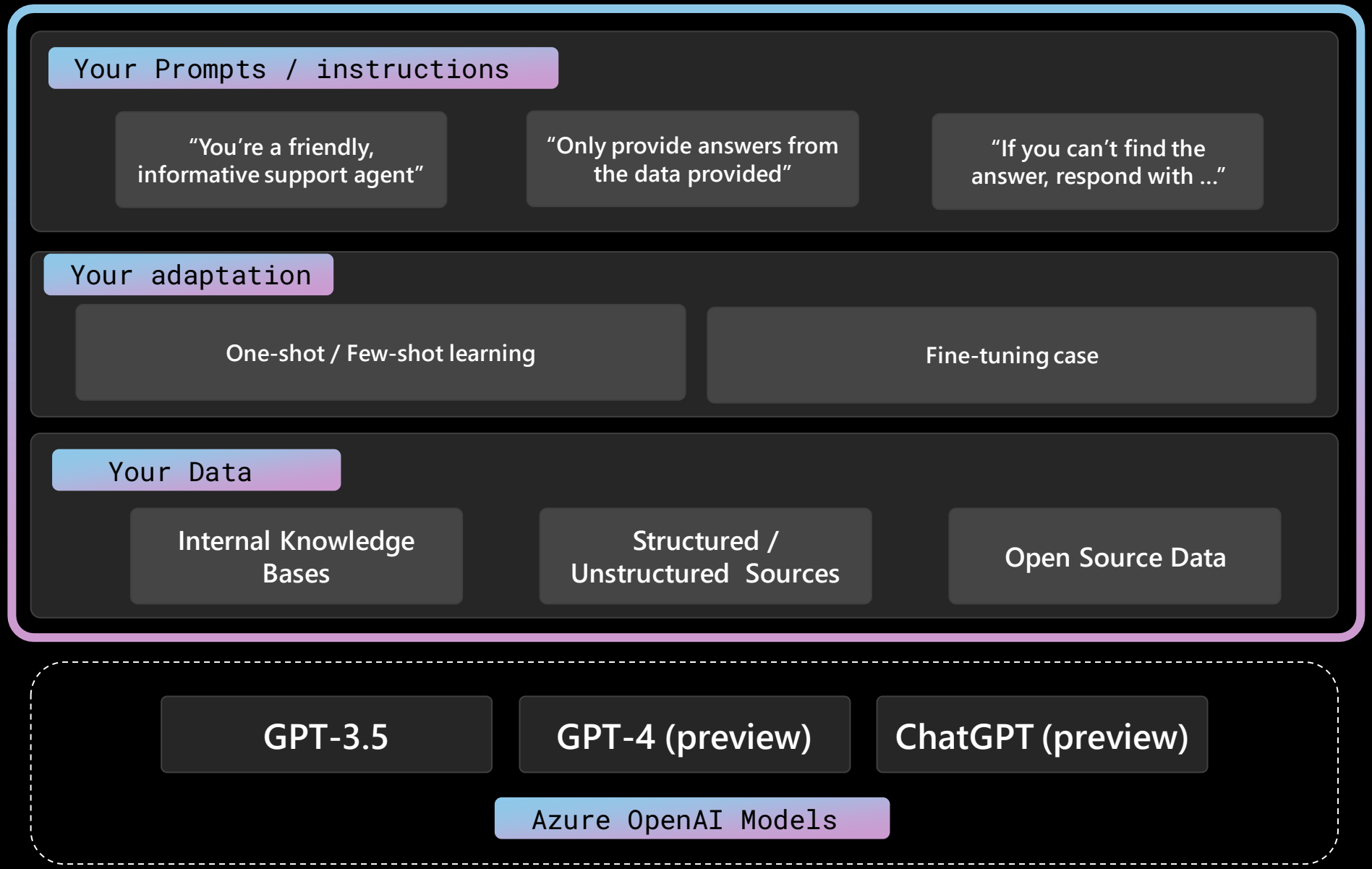
> My data feeds the (private) model



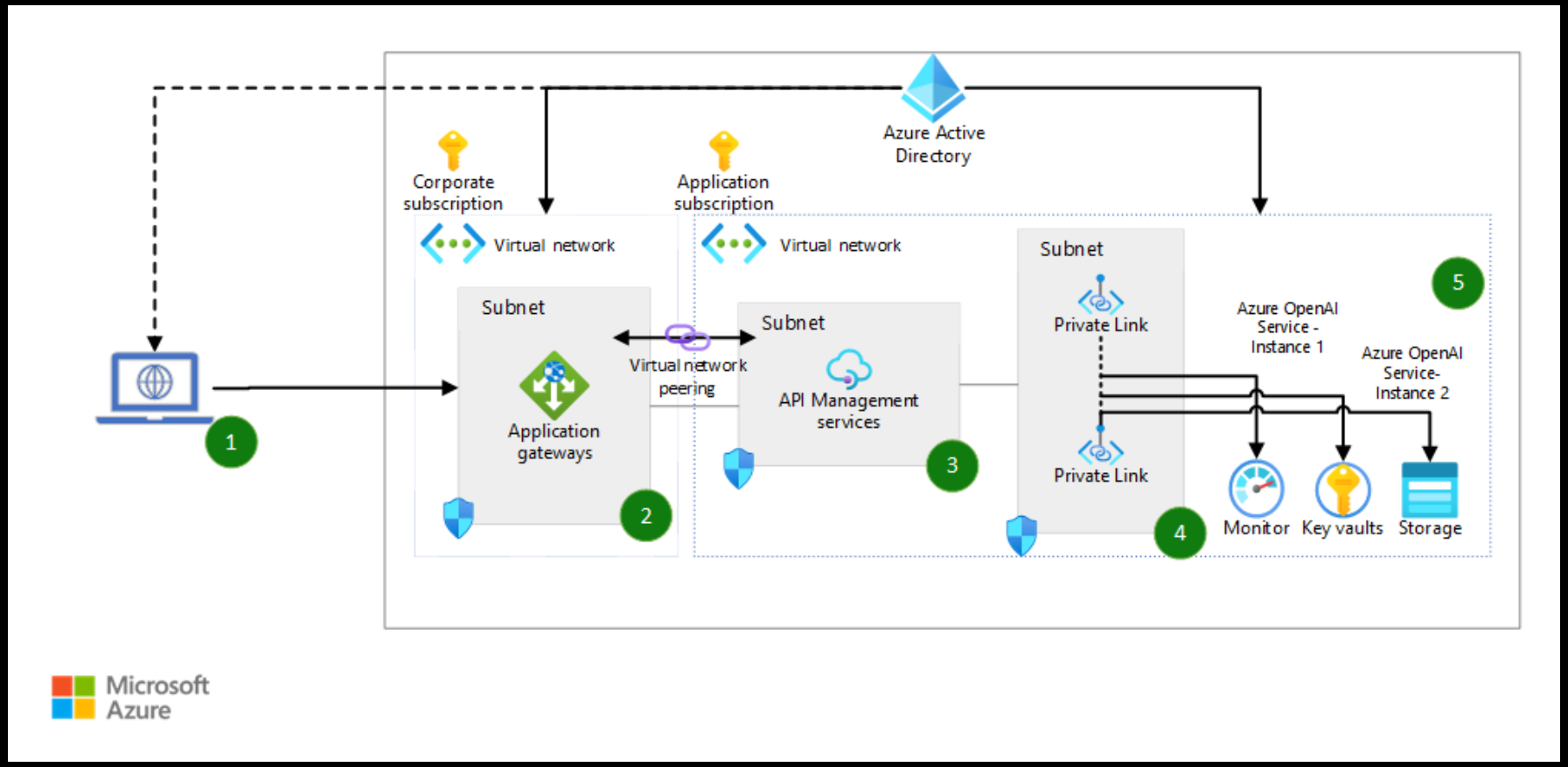
On top of the context, we inject our data

In the prompt, you can choose to let « public knowledge » accessible, use connectors (ex : Wikipedia, Bing) or having answer only based on my context (my enterprise knowledge)

> Overview (How to build my GPT Use Case ?)



Enterprise Architecture



Azure-Samples/openai-python-enterprise-logging (github.com)

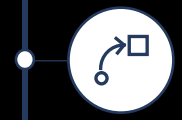
Implement logging and monitoring for Azure OpenAI large language models - Azure Architecture Center | Microsoft Learn

Enterprise Features

- VNET support
- Managed Identity
- RBAC
- Customer Managed Key (CMK)
- SDK/CLI
- Automatic content filtering

* Prompts and completions are evaluated against Microsoft content policy with automated systems. High severity content will be filtered.

Safety Best Practices



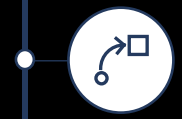
Conduct Adversarial Testing

Human in the loop

Prompt Engineering

Constrain user input and limit output tokens

Safety Best Practices



Allow users to report issues

Use End-user IDs

4

Introduction to Langchain

LangChain

- LangChain is a powerful open-source framework for developing applications powered by language models.
- It connects to the AI models you want to use, such as OpenAI or Hugging Face, and links them with outside sources, such as Google Drive, Notion, Wikipedia, or even your Apify Actors.
- The most powerful and differentiated applications will not only call out to a language model via an API but will also be data-aware and agentic. This means that they will connect a language model to other sources of data and allow it to interact with its environment.
- The LangChain framework is designed around these principles and provides modular abstractions for the components necessary to work with language models.

Example of Use Case

- Imagine you want to create a personal assistant program that can help you manage your schedule and answer your questions.
- You could use LangChain to connect to a powerful language understanding system like OpenAI or Hugging Face and link it with your calendar and email.
- This would allow the personal assistant program to understand your schedule and answer your questions about it. For example, you could ask the personal assistant “Do I have any meetings today?” and it would be able to check your calendar and respond with an answer.

LangChain Use Cases

- Autonomous Agents - These are long-running agents that take many steps in an attempt to accomplish an objective. Examples include AutoGPT and BabyAGI.
- Agent Simulations - This use case involves putting agents in a sandbox and observing how they interact with each other and react to events. This can be an effective way to evaluate their long-range reasoning and planning abilities.
- Personal Assistants - This is one of the primary use cases for LangChain. Personal assistants need to take actions, remember interactions, and have knowledge about the user's data.
- Question Answering - This is another common use case for LangChain. It involves answering questions over specific documents, using only the information in those documents to construct an answer.
- Chatbots - Language models are well-suited for creating chatbots that can engage in natural language conversations with users.
- Querying Tabular Data - This use case involves using language models to query structured data such as CSVs, SQL databases, or dataframes.

These are just a few examples of the many possible use cases for LangChain. The framework is designed to be flexible and customizable, allowing developers to create applications for a wide variety of use cases.

What are some benefits of using LangChain

LangChain is not a language model itself, but rather a framework for developing applications powered by language models.

Some benefits of using LangChain over other frameworks or building an application from scratch include:

1. **Modular abstractions** - LangChain provides modular abstractions for the components necessary to work with language models, making it easier to develop and maintain applications.
2. **Use-Case Specific Chains** - LangChain provides pre-built chains that can be assembled in particular ways to best accomplish a specific use case, making it easier to get started with a new project.
3. **Data-awareness and agency** - LangChain is designed to connect a language model to other sources of data and allow it to interact with its environment, enabling the creation of more powerful and differentiated applications.

Modular abstractions can be helpful when using LangChain

- Imagine you're a developer creating a chatbot application that can answer customer questions about a product.
- With LangChain, you could use the "Models" module to easily integrate a language model like OpenAI or Hugging Face into your application.
- Then, you could use the "Prompts" module to manage and optimize the questions that the chatbot asks the customer.
- Finally, you could use the "Memory" module to keep track of previous interactions with the customer and provide more personalized responses.
- By breaking down the process of creating a chatbot into these smaller, more manageable modules, LangChain makes it easier for developers to create and maintain their applications.

Use-Case Specific Chains

- “Use-Case Specific Chains” refers to pre-built sequences of actions that can be used to accomplish a specific task or goal.
- These chains are designed to make it easier for developers to get started with a new project because they provide a starting point that can be customized to fit the specific needs of the application.
- For example, imagine you’re a developer creating a personal assistant application that can help users manage their schedule. With LangChain, you could use a pre-built “Personal Assistant” chain that includes actions like checking the user’s calendar, sending reminders, and answering questions about the user’s schedule. You could then customize this chain to fit the specific needs of your application by adding or removing actions as needed.

Data-awareness and agency

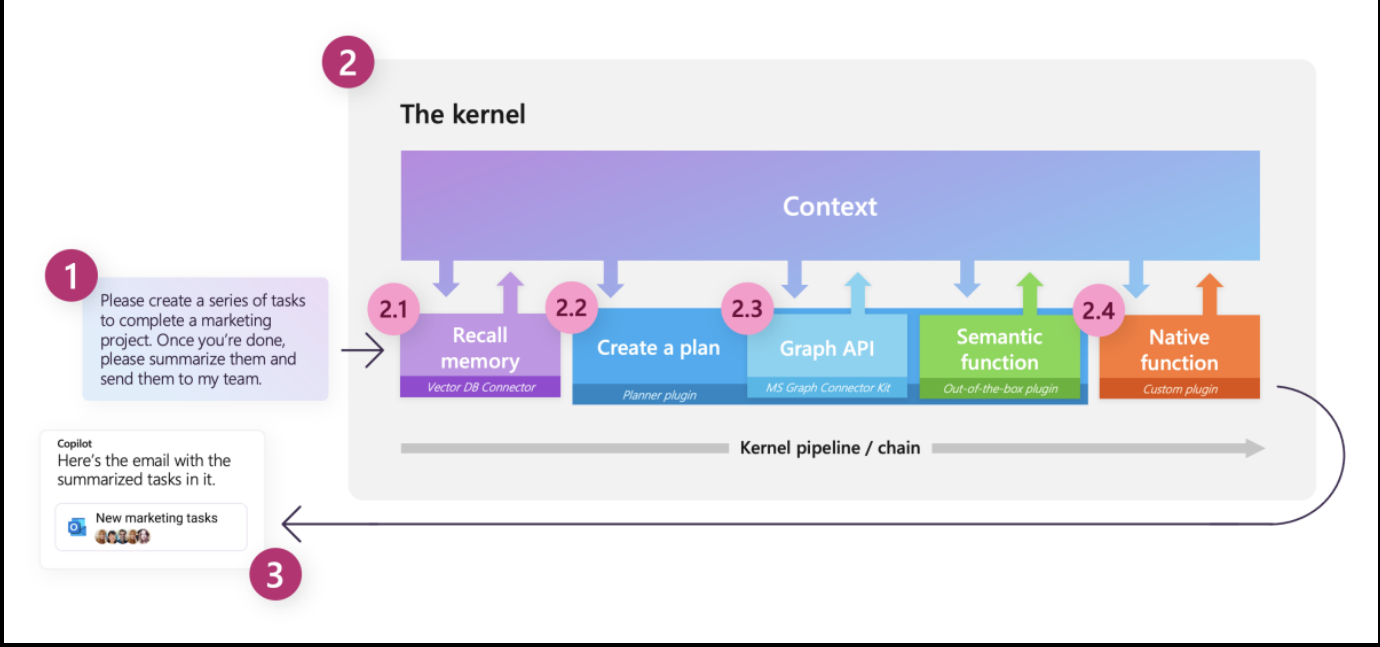
- Imagine you're a developer creating a personal shopping assistant application that can help users find and purchase products.
- With LangChain, you could use the "Indexes" module to connect your application to external sources of data like product catalogs or user purchase history. This would allow the shopping assistant to understand the user's preferences and make personalized recommendations.
- In addition, LangChain allows the shopping assistant to interact with its environment by taking actions like adding items to a shopping cart or completing a purchase. This agency enables the shopping assistant to provide a more seamless and helpful experience for the user.

Langchain components

- These components are designed to be easy to use and can be combined in different ways to create a wide variety of applications.
- Some of the core components provided by LangChain include:
  - Models - This module provides support for different types of language models and integrations with popular language model providers like OpenAI and Hugging Face.
  - Prompts - This module provides tools for managing, optimizing, and serializing the prompts that are used to generate responses from a language model.
  - Memory - This module provides tools for persisting state between calls to a chain or agent, allowing applications to remember previous interactions and provide more personalized responses.
  - Indexes - This module provides interfaces and integrations for loading, querying, and updating external data sources, allowing applications to combine the power of language models with application-specific data.
  - Chains - This module provides tools for creating structured sequences of calls to a language model or other utilities.
  - Agents - This module provides tools for creating agents that can repeatedly decide on an action, execute the action, and observe the outcome until a high-level directive is complete.
  - Callbacks - This module provides tools for logging and streaming the intermediate steps of any chain, making it easier to observe, debug, and evaluate the internals of an application.

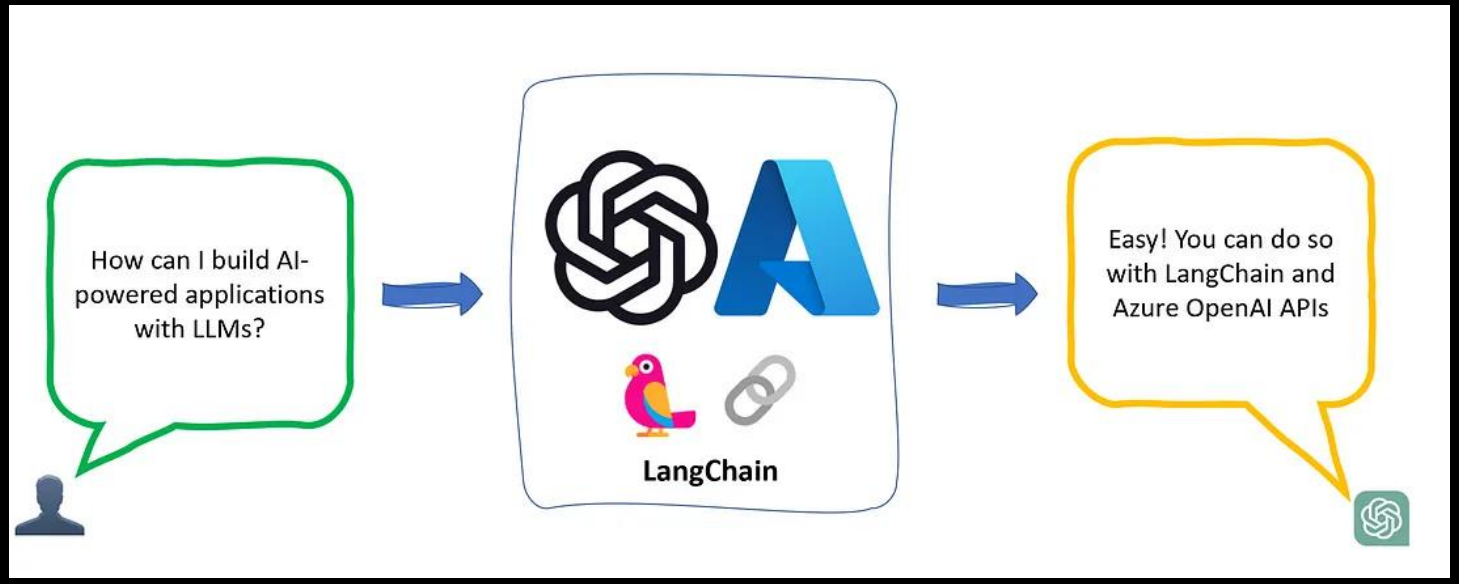
Semantic Kernel – An alternative to Langchain

- Semantic Kernel is a lightweight open-source orchestration SDK that lets you easily mix-and-match AI prompts with conventional programming languages like C# and Python.
- Semantic Kernel provides a set of abstractions that make it easy to create and manage prompts, native functions, memories, and connectors.
- You can then orchestrate these components using Semantic Kernel pipelines to complete users' requests or automate actions.



Step	Component	Description
1	Ask	It starts with goal being sent to Semantic Kernel as an ask by either a user or developer.
2	Kernel	The kernel orchestrates a user's ask. To do so, the kernel runs a pipeline / chain that is defined by a developer. While the chain is run, a common context is provided by the kernel so data can be shared between functions.
2.1	Memories	With a specialized plugin, a developer can recall and store context in vector databases. This allows developers to simulate memory within their AI apps.
2.2	Planner	Developers can ask Semantic Kernel to auto create chains to address novel needs for a user using planner. Planner is able to use any of the plugins that have already been loaded into the kernel to create new additional steps.
2.3	Connectors	To get additional data or to perform autonomous actions, you can use out-of-the-box connectors like the Microsoft Graph Connector kit or create a custom connector to connect to your own services.
2.4	Custom functions	As a developer, you can create custom functions that run inside of Semantic Kernel. These can either be LLM prompts (semantic functions) or native C# or Python code (native function). This allows you to add new AI capabilities and integrate your existing apps and services into Semantic Kernel.
3	Response	Once the kernel is done, you can send the response back to the user to let them know the process is complete.

Demo


How can I build AI-powered applications with LLMs?
LangChain
Easy! You can do so with LangChain and Azure OpenAI APIs

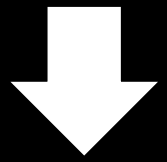
5

Embeddings and Vector DBs

- Embedding & Azure Open AI

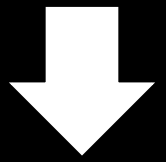
Embeddings make it possible to map content to a “semantic space”

A neutron star is the collapsed core of a massive supergiant star



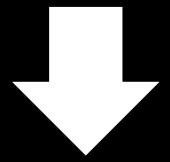
[15 34 24 13 ...]

A star shines for most of its active life due to thermonuclear fusion.



[16 22 89 26 ...]

The presence of a black hole can be inferred through its interaction with other matter



[20 13 31 89 ...]

Model Capabilities – Embeddings models

Text Similarity	Capturing semantic similarity between two or more pieces of text	Clustering, regression, anomaly detection, visualization
Text Search	Measure long documents are relevant to a short query	Search, context relevance, information retrieval
Code Search	Useful for embedding code snippets and embedding nature language search queries	Code search and relevance

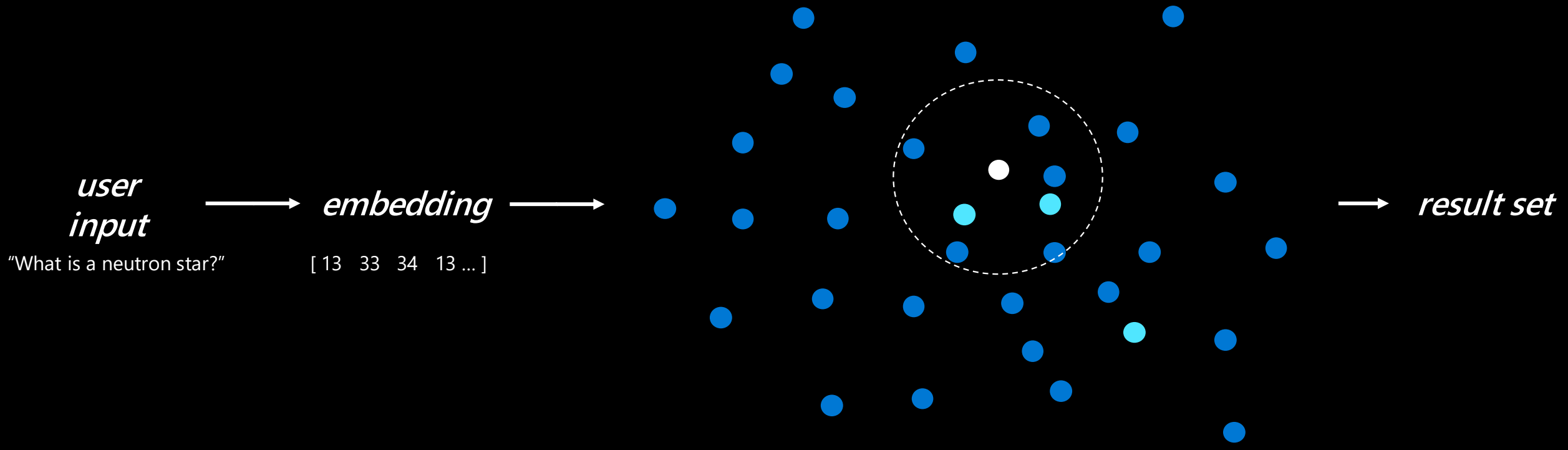
Second-generation models

MODEL NAME	TOKENIZER	MAX INPUT TOKENS	OUTPUT DIMENSIONS
text-embedding-ada-002	cl100k_base	8191	1536

https://openai.com/blog/new-and-improved-embedding-model

Similarity Search with embeddings

Once you encode your content as embeddings, you can then get an embedding from the user input and use that to find the most semantically similar content.



Azure OpenAI embeddings tutorial - Azure OpenAI | Microsoft Learn

- Vector DB

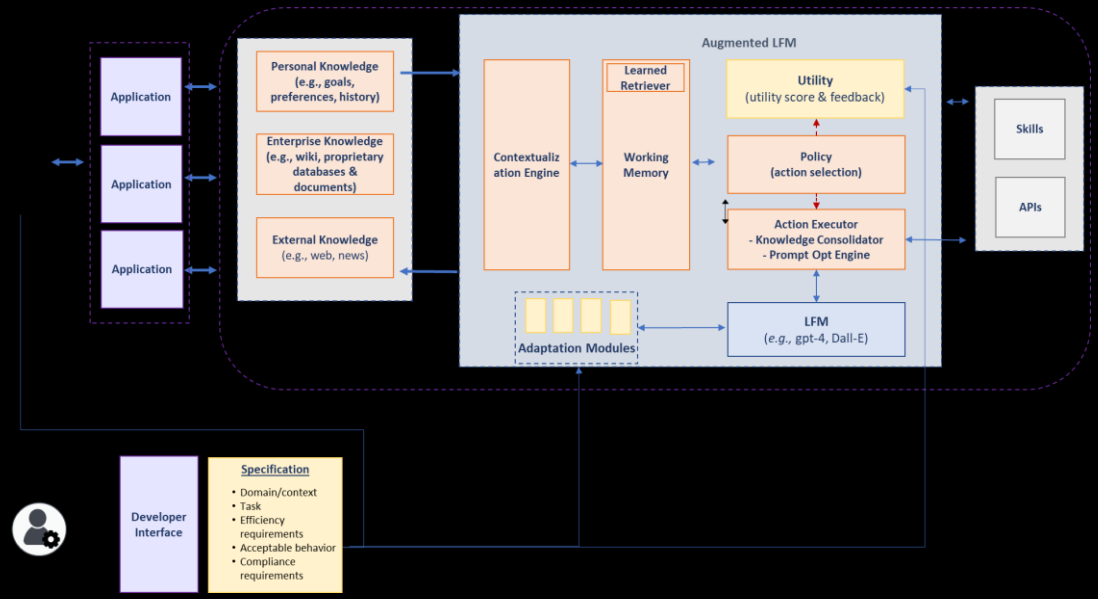
Introduction

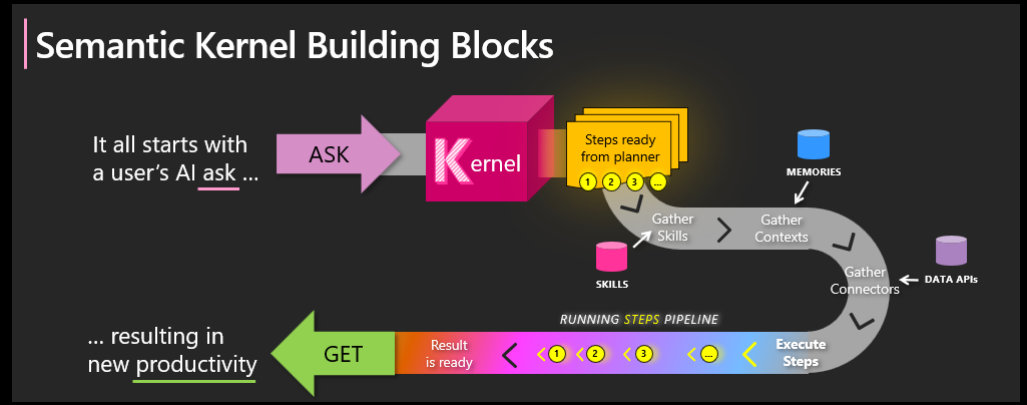
Vector Database | Microsoft Learn

- A vector database is a type of database that stores data as high-dimensional vectors, which are mathematical representations of features or attributes.
- Each vector has a certain number of dimensions, which can range from tens to thousands, depending on the complexity and granularity of the data
- The vectors are usually generated by applying some kind of transformation or embedding function to the raw data, such as text, images, audio, video, and others.

Augmenting LFMS with External Knowledge

- One of the best ways to augment an LFM is by supplementing it with external knowledge that was not used in the training of the model
- A vector database can be used to ground prompts
- Vector databases are the evolution of Approximate Nearest Neighbor (ANN) indexes
- MSR has some of the most differentiated and deployed ANN technology
  - Part of core search infrastructure for Ads, Bing search and Enterprise
  - Vector indexing algorithm implemented in C++ that enables trillion scale indexes



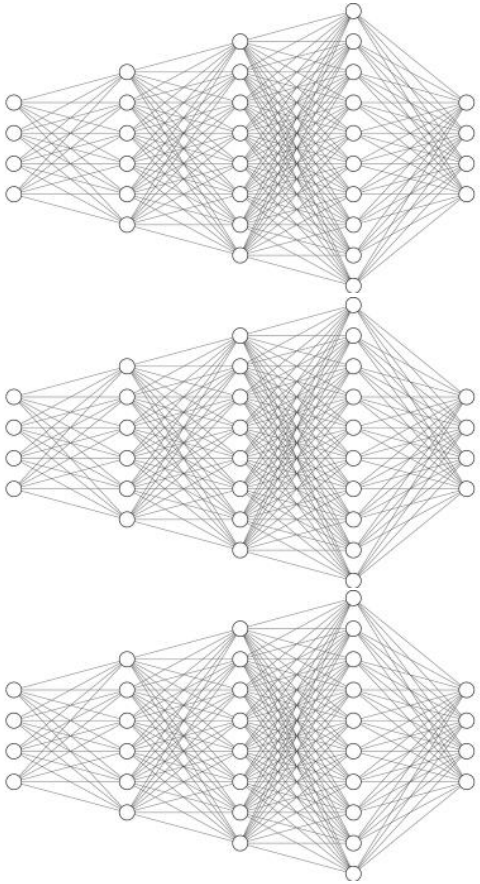


Our Goals

- Any 3rd party copilot solution leveraging Azure can use a vector DB.
- Allow ML and other internal developers to leverage same ecosystem
- Solve and support the heterogeneity of needs

Not just Language models LFMs

Input



Model

Embedding

Vector Database

Existing *connectors*

- **Available connectors to vector databases**

- One use case for storing information in a vector database is to enable large language models (LLMs) to generate more relevant and coherent text based on an AI plugin.

- Today, we have several connectors to vector databases that you can use to store and retrieve information. These include:
  - Azure Cognitive Search
  - COSMOS DB
  - Pinecone
  - Postgres
  - Qdrant
  - Sqlite

openai/chatgpt-retrieval-plugin: The ChatGPT Retrieval Plugin lets you easily find personal or work documents by asking questions in natural language. (github.com)

Vector Databases *supported* in Azure

- Azure Cognitive Search
- Cosmos DB for MongoDB vCore
- Cosmos DB for PostgreSQL
- Azure Data Explorer
- Azure Cache for Redis Enterprise

Azure Data Explorer as a Vector Database

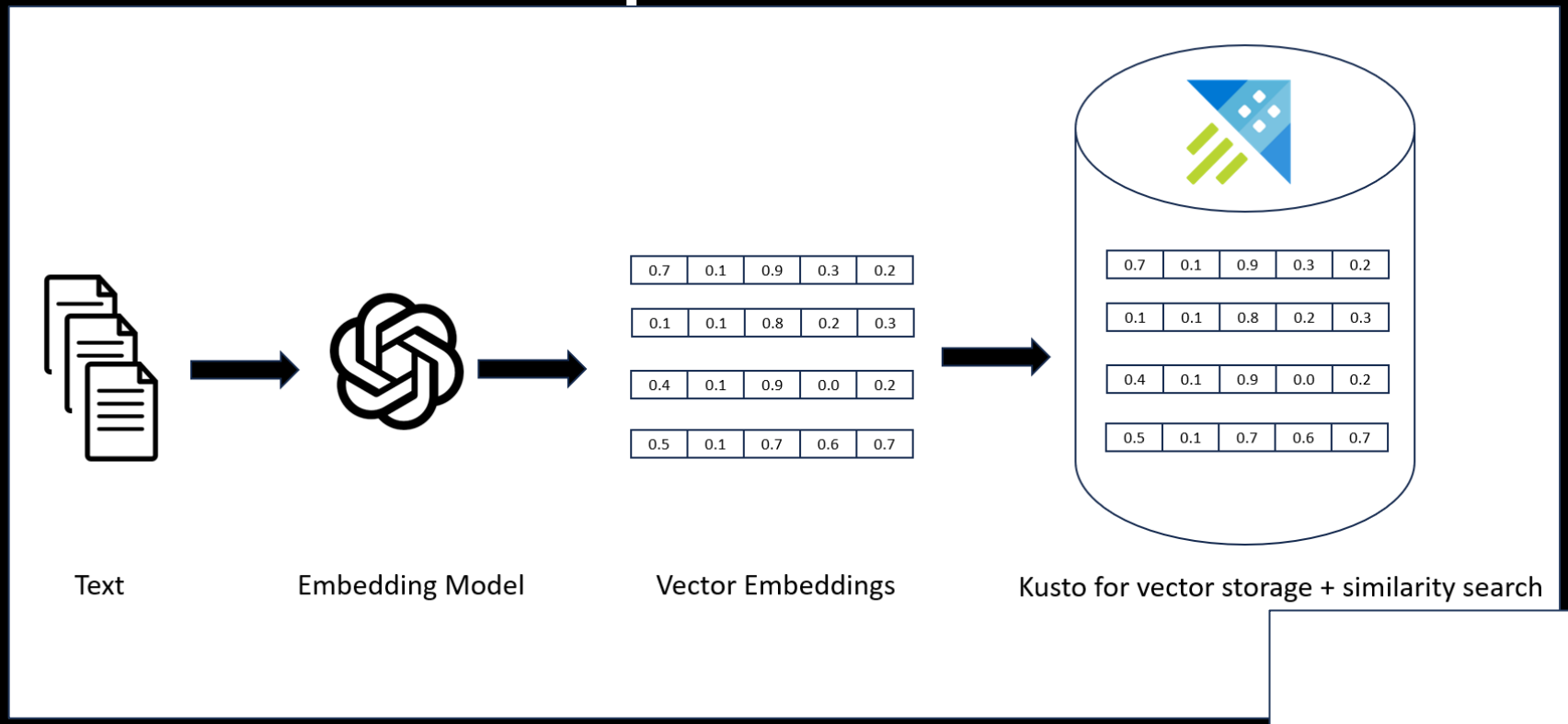
- Azure Data Explorer aka Kusto is a cloud-based data analytics service that enables users to perform advanced analytics on large datasets in real-time.

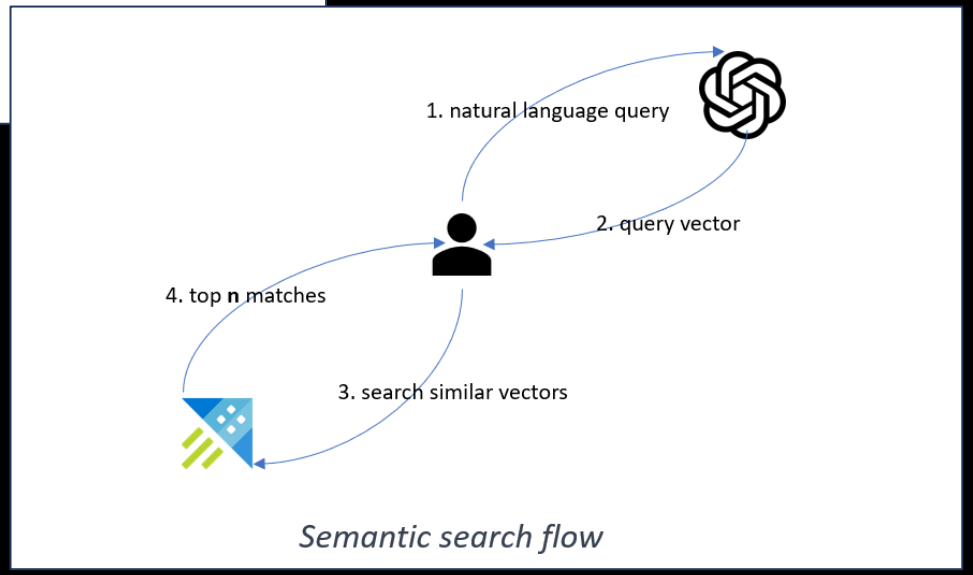
- Kusto supports a special data type called dynamic, which can store unstructured data such as arrays and properties bag. Dynamic data type is perfect for storing vector values.

- Kusto also supports in-built function series_cosine_similarity_fl to perform vector similarity searches.

Get started with Kusto for free.

Azure Data Explorer as a Vector Database





Azure Data Explorer as a Vector Database

- You need to execute the below function code in your ADX resource's Query editor. This creates a function **series_cosine_similarity_fl** to perform vector similarity searches.

```
.create-or-alter function with (docstring = "Calculate the Cosine similarity of 2 numerical arrays",folder = "Packages\\Series") series_cosine_similarity_fl(vec1:dynamic,vec2:dynamic,vec1_size:real=real(null),vec2_size:real=real(null)) {
    let dp = series_dot_product(vec1, vec2);
    let v1l = iff(isnull(vec1_size), sqrt(series_dot_product(vec1, vec1)), vec1_size);
    let v2l = iff(isnull(vec2_size), sqrt(series_dot_product(vec2, vec2)), vec2_size);
    dp/(v1l*v2l)
}
```

series_cosine_similarity_fl() - Azure Data Explorer | Microsoft Learn

Azure Data Explorer as a Vector Database

Using _kusto_queries | Confidential\Microsoft Extended · Saved

Search

Trial: 59 days left

Home Edit Run Data View

Comment Editing Share

Run all | Stop session | Language PySpark (Pytho... | Open in VS Code

Lakehouse explorer

FGILakeDB

Tables

Files

Synapse notebooks and Spark job definitions are in Preview. Other users in your organization may have access to this workspace. Do not use these items unless you trust all other users who may have access to the workspace.

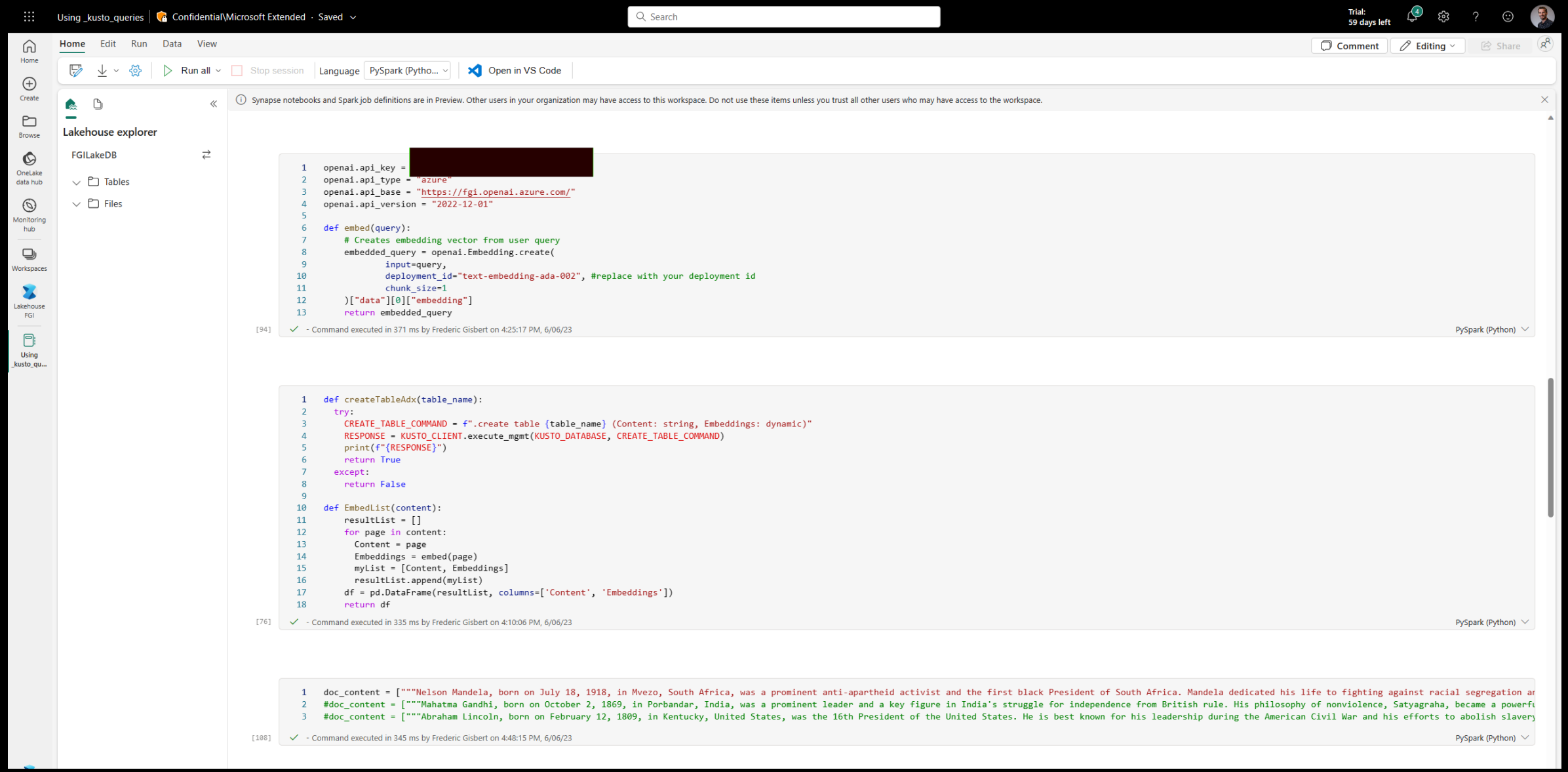
```
from azure.kusto.data import KustoConnectionStringBuilder,KustoClient
from azure.kusto.ingest import QueuedIngestClient, IngestionProperties
from azure.kusto.data.data_format import DataFormat
from azure.kusto.data.helpers import dataframe_from_result_table
import openai
import pandas as pd
```

[3] ✓ - Command executed in 346 ms by Frederic Gisbert on 2:59:28 PM, 6/06/23 PySpark (Python)

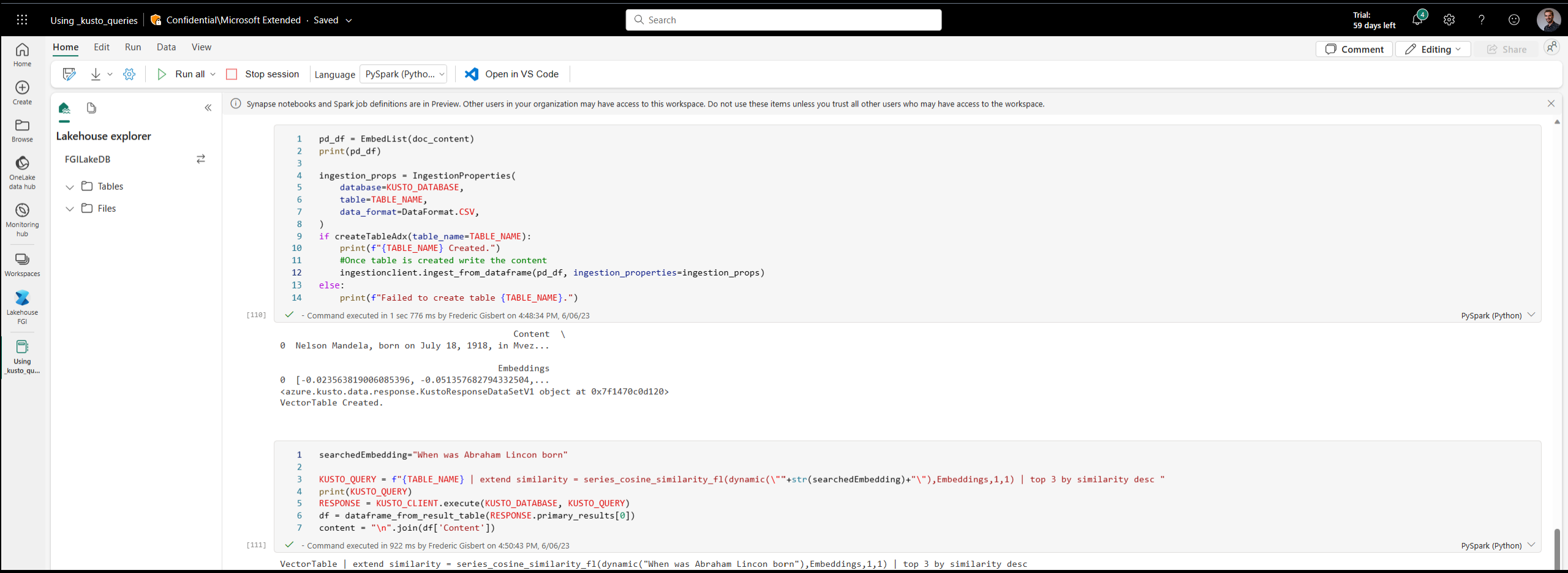
```
KUSTO_DATABASE = "Vector"
TABLE_NAME = "VectorTable" #We will be creating first then querying the same table
cluster = "https://fgiadx.francecentral.kusto.windows.net"
ingest_cluster = "https://ingest-fgiadx.francecentral.kusto.windows.net"
aad_app_id [redacted]
aad_app_key
tenant_id
connection_string = KustoConnectionStringBuilder.with_aad_application_key_authentication(
    cluster, aad_app_id, aad_app_key,tenant_id)
KUSTO_CLIENT = KustoClient(connection_string)

#Cliengt for Ingestion - when we need ingest embedding in database
kcsb = KustoConnectionStringBuilder.with_aad_application_key_authentication(ingest_cluster, aad_app_id, aad_app_key, tenant_id)
ingestionclient = QueuedIngestClient(kcsb)
```

[69] ✓ - Command executed in 344 ms by Frederic Gisbert on 4:07:36 PM, 6/06/23 PySpark (Python)

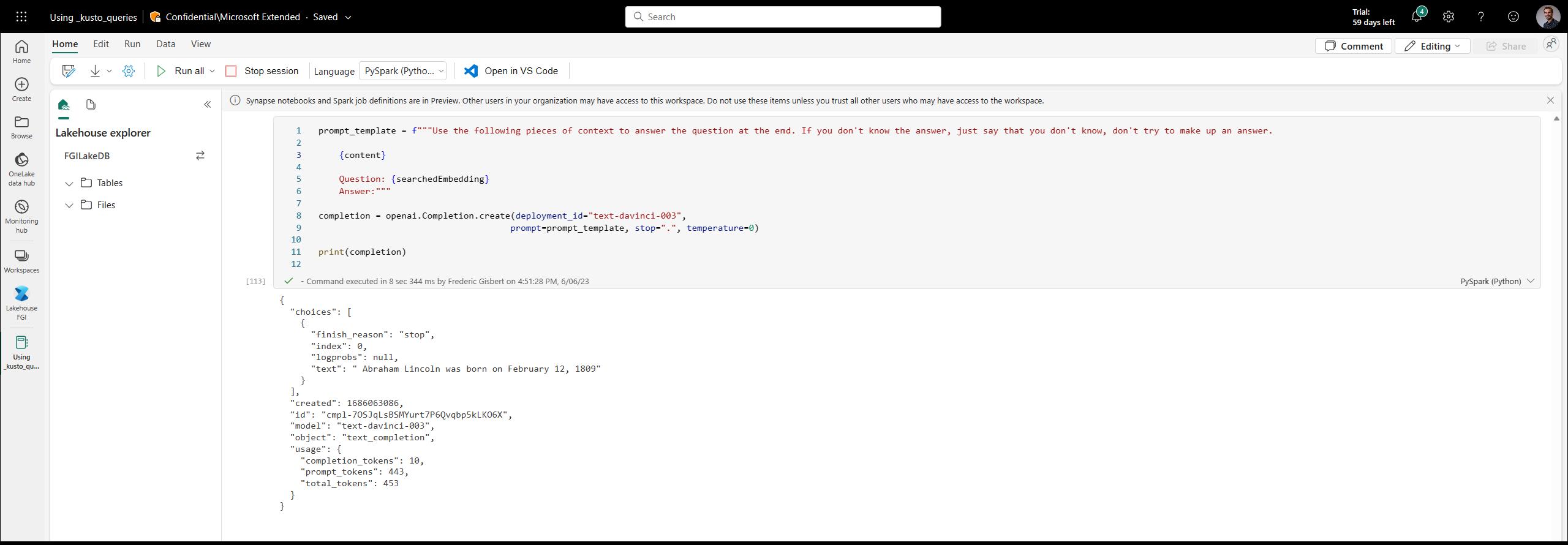
Azure Data Explorer as a Vector Database



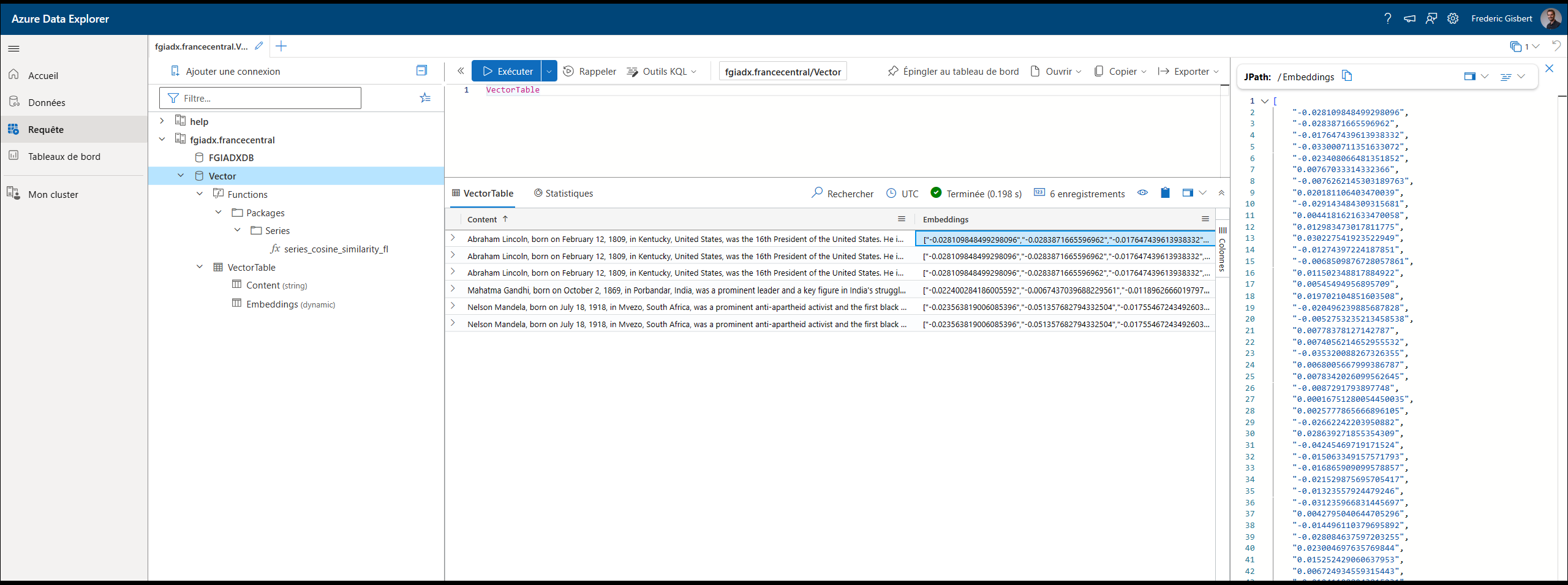
Azure Data Explorer as a Vector Database



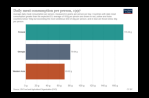
Azure Data Explorer as a Vector Database



Azure Data Explorer as a Vector Database



Art of the Possible: Multimodal Visual & Text QnA



Extracted Text from Image

Image Text

Image Corpus

Extract/Crop images from documents

Vector Image text embeddings

Top K Image Text

Vector Text embeddings

Top K Text

Image Text + Text = response

Input Prompt

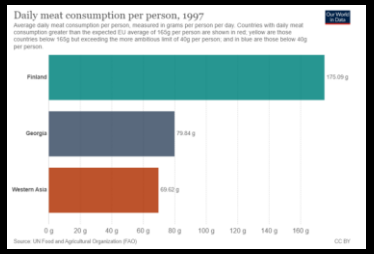
What is the sum of average daily meat consumption for Georgia and Western Asia?

Compietion

To find the sum of average daily meat consumption for Georgia and Western Asia, we will follow these steps:
Identify the average daily mean consumption for Georgia.
Identify the average daily meat consumption for Western Asia.
Add the values from steps 1 and 2.

Step 1: From the provided chart, we see that the average daily meat consumption for Georgia is 79.84 grams per person per day.
Step 2: From the chart, we see that the average daily meat consumption for Western Asia is 69.62 grams per person per day.
Step 3: Add the values from steps 1 and steps 2.
79.84 g (Georgia) + 69.62 g (Western Asia) = 149.46 g

So, the sum of average daily meat consumption of Georgia and Western Asia is 149.46 grams per person per day



Text Corpus

Form Recognizer (OCR)

+

Cognitive Search (KB)

*This approach uses image text embeddings for scenario's where the relevant image could come from another source document

Step-by-step process flow in the Jupyter Notebook

Step-by-step process flow

- Use Form Recognizer to extract text from papers
- Chunk the text and create embeddings using the embeddings model in Azure OpenAI
- Spin up Azure Cache for Redis (Enterprise) and store the embeddings
- Leverage vector Similarity feature of Cache for Redis to get top n similar chunks
- Take top n similar texts as a passage and prompt engineer into passage based QnA solution

GitHub Repo: azure-openai-samples/use_cases at main · Azure/azure-openai-samples (github.com)

6

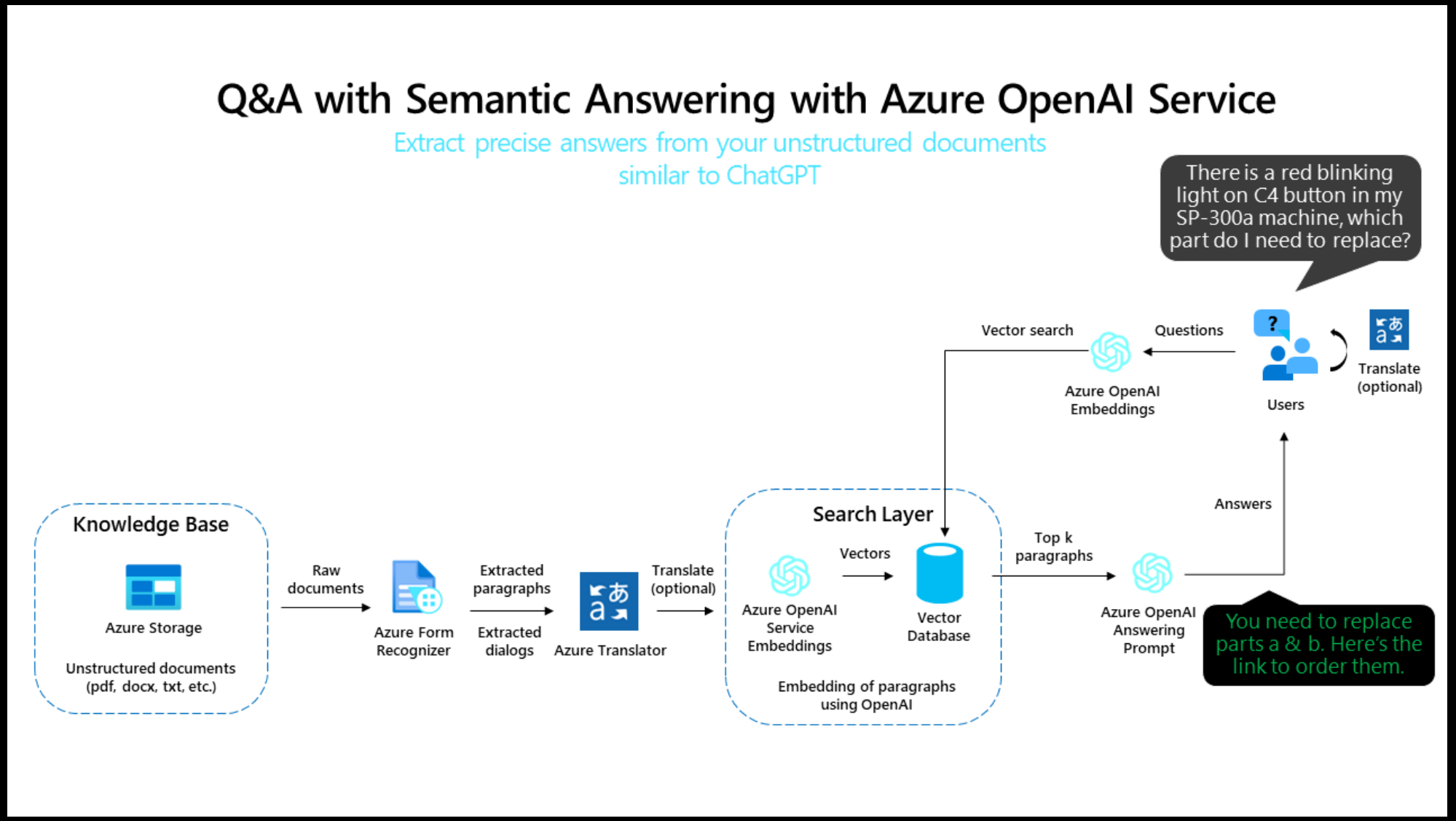
Architecture - AOAI Embeddings QnA

Azure OpenAI Embeddings QnA

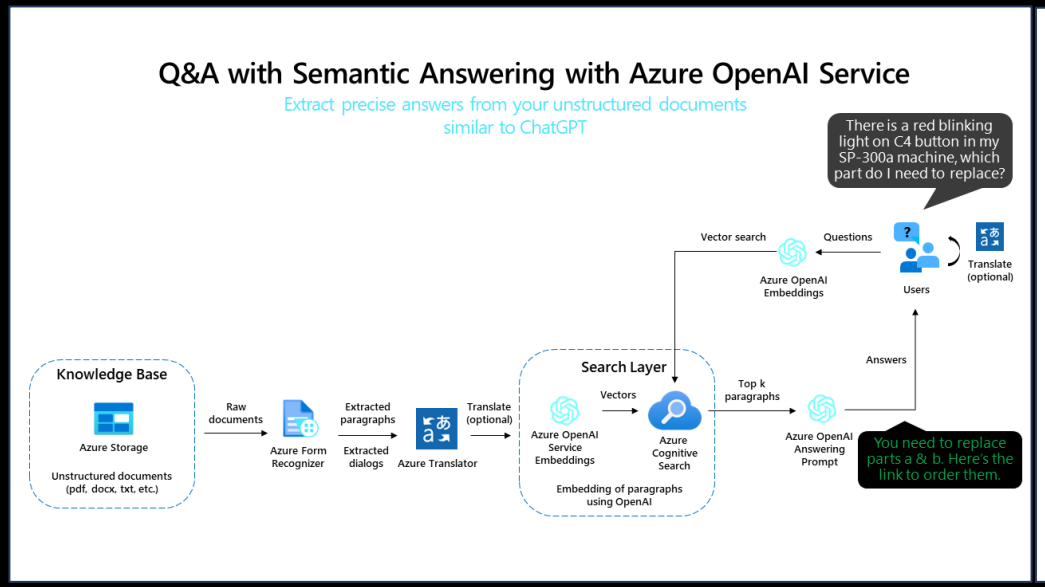
- A simple web application for a OpenAI-enabled document search.
- This repo uses Azure OpenAI Service for creating embeddings vectors from documents.
- For answering the question of a user, it retrieves the most relevant document and then uses GPT-3 to extract the matching answer for the question.

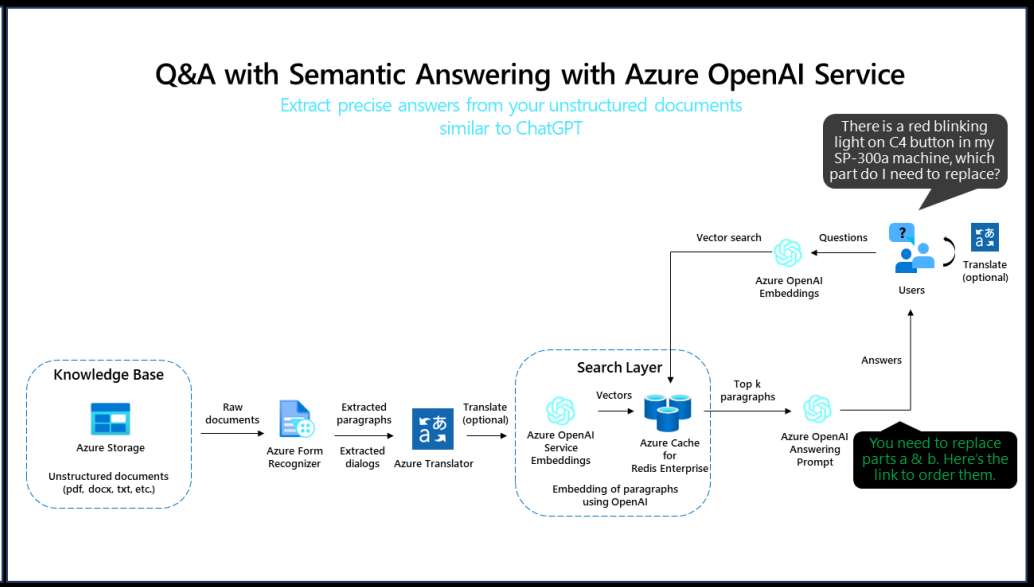
https://github.com/ruoccofabrizio/azure-open-ai-embeddings-qna

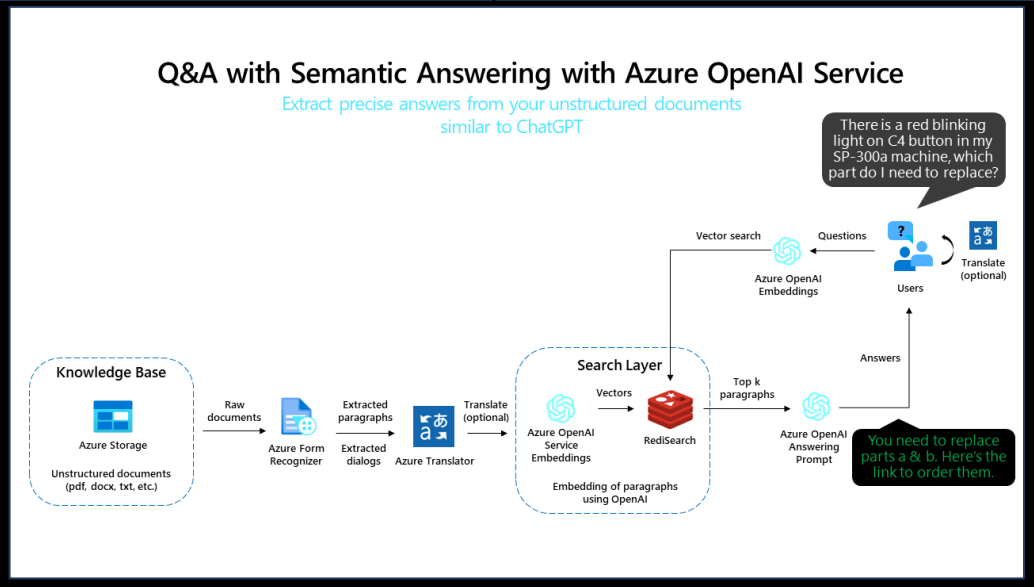
Azure OpenAI Embeddings QnA



Azure OpenAI Embeddings QnA







Azure Cognitive Service Vector Search

Storage and vector index size limits

The size of your vector index in memory is restricted based on the reserved memory for the chosen SKU. It is calculated **per partition** and is a hard limit. The storage and vector index size quotas are not separate quotas; vector indexes share the same underlying storage quota. For example, if storage quota is exhausted but there is remaining vector quota, you won't be able to index any more documents, regardless if they're vector documents, until you scale up in partitions or delete some documents.

The limits are set conservatively and are preliminary during this period. We are still investigating performance if the limits are raised.

An approximate translation into the number of floating point numbers is provided. **This is not the same number as vector dimensionality.** The dimensionality of a vector field affects how many floating point numbers are contained in one vector embedding.

For example, using the most popular Azure OpenAI model, `text-embedding-ada-002` with 1,536 dimensions means one document would consume 1,536 floats. Similarily, 100 documents with a single, 1,536-dimensional vector field would consume in total 100 docs x 1536 floats/doc = 153,600 floats. 1,000 documents with two 768-dimensional vector fields, consume 1000 docs x 2 fields x 768 floats/doc = 1,536,000 floats.

SKU	Storage quota (GB)	Vector index size quota per partition (GB)	Approx. floats per partition
Basic	2	0.5	134 million
S1	25	1	268 million
S2	100	6	1,611 million
S3	200	12	3,221 million
L1	1,000	12	3,221 million
L2	2,000	36	9,664 million

- Cognitive Search can index and store vectors, but it doesn't generate them out of the box.

- The documents that you push to your search service must contain vectors within the payload
- Alternatively, you can use the Indexer to pull vectors from your data sources such as Blob Storage JSON files or CSVs.
- You can also use a custom skill to generate embeddings as part of an AI enrichment process.

Azure OpenAI Embeddings QnA

```python
class AzureSearch(VectorStore):
    def __init__(
        self,
        azure_cognitive_search_name: str,
        azure_cognitive_search_key: str,
        index_name: str,
        embedding_function: Callable,
        semantic_configuration_name: str = None,
        semantic_query_language: str = "en-us",
        **kwargs: Any,
    ):
        """Initialize with necessary components."""
        try:
            from azure.search.documents import SearchClient
        except ImportError:
            raise ValueError(
                "Could not import requests python package. "
                "Please install it with `pip install --index-url https://pkgs.dev.azure.com/azure-sdk/public/
                )
        # Initialize base class
        self.embedding_function = embedding_function
        self.azure_cognitive_search_name = azure_cognitive_search_name
        self.azure_cognitive_search_key = azure_cognitive_search_key
        self.index_name = index_name
        self.semantic_configuration_name = semantic_configuration_name
        self.semantic_query_language = semantic_query_language
        self.client = get_search_client(
            self.azure_cognitive_search_name, self.azure_cognitive_search_key, self.index_name, self.semantic

    def add_texts(
        self,
        texts: Iterable[str],
        metadatas: Optional[List[dict]] = None,
        **kwargs: Any,
    ) -> List[str]:
        """Add texts data to an existing index."""
        keys = kwargs.get("keys")
        keys = list(map(lambda x: x.replace(':','_'), keys)) if keys else None
```

```python
            raise Exception(response)

    def similarity_search(
        self, query: str, k: int = 4, **kwargs: Any
    ) -> List[Document]:
        """
        Returns the most similar indexed documents to the query text.

        Args:
            query (str): The query text for which to find similar documents.
            k (int): The number of documents to return. Default is 4.

        Returns:
            List[Document]: A list of documents that are most similar to the query
        """
        docs_and_scores = self.similarity_search_with_score(
            query, k=k, filters=kwargs.get("filters", None))
        return [doc for doc, _ in docs_and_scores]

    def similarity_search_with_score(
        self, query: str, k: int = 4, filters: str = None
    ) -> List[Tuple[Document, float]]:
        """Return docs most similar to query.

        Args:
            query: Text to look up documents similar to.
            k: Number of Documents to return. Defaults to 4.

        Returns:
            List of Documents most similar to the query and score for each
        """
        results = self.client.search(
            search_text=""
```

```python
import os
import openai
from dotenv import load_dotenv
import logging
import re
import hashlib

from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.llms import AzureOpenAI
from langchain.vectorstores.base import VectorStore
from langchain.chains import ChatVectorDBChain
from langchain.chains import ConversationalRetrievalChain
from langchain.chains.qa_with_sources import load_qa_with_sources_chain
from langchain.chains.llm import LLMChain
from langchain.chains.chat_vector_db.prompts import CONDENSE_QUESTION_PROMPT
from langchain.prompts import PromptTemplate
from langchain.document_loaders.base import BaseLoader
from langchain.document_loaders import WebBaseLoader
from langchain.text_splitter import TokenTextSplitter, TextSplitter
from langchain.document_loaders.base import BaseLoader
from langchain.document_loaders import TextLoader
from langchain.chat_models import ChatOpenAI
from langchain.schema import AIMessage, HumanMessage, SystemMessage

from utilities.formrecognizer import AzureFormRecognizerClient
from utilities.azureblobstorage import AzureBlobStorageClient
from utilities.translator import AzureTranslatorClient
from utilities.customprompt import PROMPT
from utilities.redis import RedisExtended
from utilities.azuresearch import AzureSearch

import pandas as pd
import urllib

from fake_useragent import UserAgent

class LLMHelper:
    def __init__(self,
        document_loaders : BaseLoader = None,
        text_splitter: TextSplitter = None,
        embeddings: OpenAIEmbeddings = None,
        llm: AzureOpenAI = None,
        temperature: float = None,
        max_tokens: int = None,
```

[Azure/cognitive-search-vector-pr: The official documentation and code samples for the Vector search feature (preview) in Azure Cognitive Search. (github.com)]

- OpenAI Queries
- Chat
- Add Document
- Document Viewer
- Index Management
- Sandbox
- Utils - Document Summary
- Utils - Conversation Data Extraction
- Utils - Prompt Exploration

LLM is working!

Embedding is working!

Translation is working!

Azure Cognitive Search is working!

Check deployment

Settings

OpenAI Semantic Answer

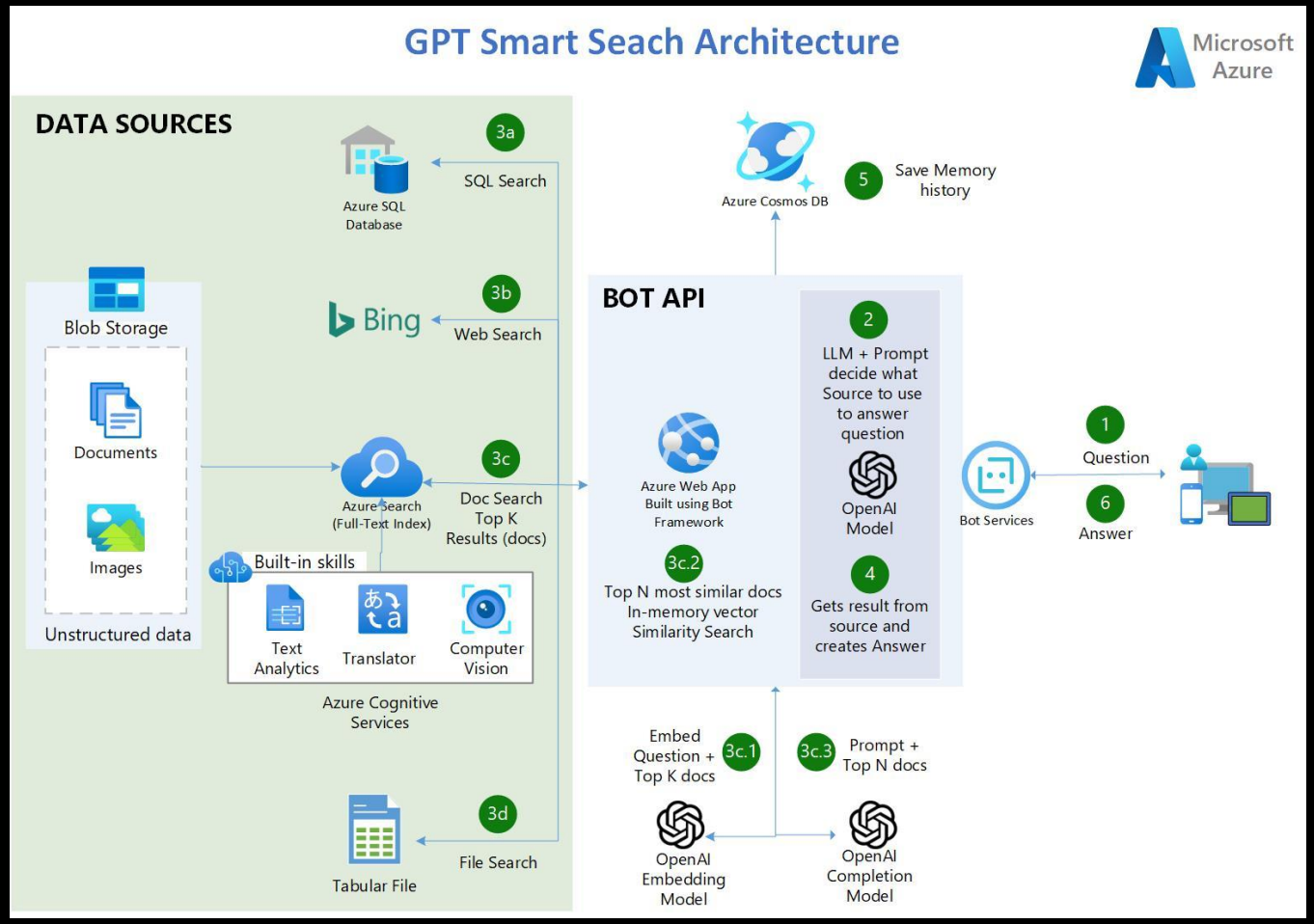
Made with Streamlit

7

Architecture – Semantic Search with AOAI

GPT Azure Search Engine

- The **Azure-Cognitive-Search-Azure-OpenAI-Accelerator** repository is a 3-5 days POC VBD powered by Azure Search, Azure OpenAI, Bot Framework, Langchain, Azure SQL, CosmosDB and Bing Search API.
- It is designed to provide a Multi-Channel Smart Chatbot and a search engine capable of comprehending diverse types of data scattered across various locations.
- Additionally, the conversational chatbot should be able to provide answers to inquiries, along with the source and an explanation of how and where the answer was obtained.
- The goal of the MVP POC is to show/prove the value of a GPT Virtual Assistant built with Azure Services, with your own data in your own environment.
- The deliverables are a Backend Bot API built with Bot Framework and exposed to multiple channels (Web Chat, MS Teams, SMS, Email, Slack, etc) and a Frontend web application with a Search and a Bot UI.



Demo

1.The user asks a question.

2.In the app, an OpenAI LLM uses a clever prompt to determine which source contains the answer to the question.

3.Four types of sources are available:

1. 3a. Azure SQL Database - contains COVID-related statistics in the US.
2. 3b. Azure Bing Search API - provides online web search for current information.
3. 3c. Azure Cognitive Search - contains AI-enriched documents from Blob Storage (10k PDFs and 52k articles).
    1. 3c.1. Uses an LLM (OpenAI) to vectorize the top K document chunks from 3c.
    2. 3c.2. Uses in-memory cosine similarity to get the top N chunks.
    3. 3c.3. Uses an OpenAI GPT model to craft the response from the Cog Search Engine (3c) by combining the question and the top N chunks.
4. 3d. CSV Tabular File - contains COVID-related statistics in the US.

4.The app retrieves the result from the source and crafts the answer.

5.The tuple (Question and Answer) is saved to CosmosDB to keep a record of the interaction.

6.The answer is delivered to the user.

Azure OpenAI Service Learning Guide

Start Here!

- Create an Azure subscription
- ★ Apply for access to the Azure OpenAI Service: https://aka.ms/oai/access
- Watch the video: An Introduction to Azure OpenAI
- Get Familiarized with the product content: Azure OpenAI Page
- Review how other customers are using Azure AI

Basic

- Understand "What is Azure OpenAI?": compare Azure OpenAI and OpenAI. Review key concepts.
- ★ Work through Introduction to Azure OpenAI Service training module
- Explore key Responsible AI guidelines and principles

Intermediate

- How-to guides: Generate embeddings and Fine-tune
- Complete embeddings and document search tutorial
- How your/your customer data is processed, used and stored in Azure OpenAI: Data, privacy and security
- ★ Review the Enterprise Data with ChatGPT tech blog, and work through the accompanying GitHub repository.
- Become familiar with the APIs

Advanced

- Learn more in the Azure OpenAI Workshop
- ★ See examples in the Open AI Cookbook
- Start experimenting and developing your MVP
- Get support and help

Microsoft

Thank you