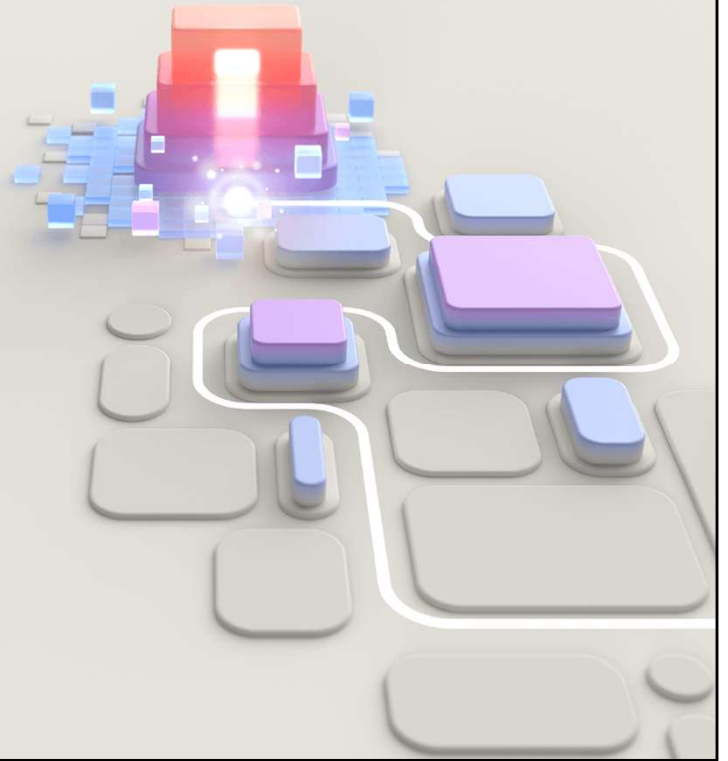





Develop Generative AI Solutions with Azure OpenAI Service



© Copyright Microsoft Corporation. All rights reserved.

Agenda



- Get started with Azure OpenAI Service
- Develop apps with Azure OpenAI Service
- Apply prompt engineering with Azure OpenAI Service
- Use your own data with Azure OpenAI Service

© Copyright Microsoft Corporation. All rights reserved.

Get started with Azure OpenAI Service

© Copyright Microsoft Corporation. All rights reserved.

Before delivering this presentation, review the associated module on Microsoft Learn (<https://aka.ms/mslearn-start-azure-openai>).

The goal of this section is to describe how to provision the Azure OpenAI service, select and deploy a model, and use the Playground in the **Azure AI Foundry portal** to submit prompts and review results from models.

The estimated time to present these slides is 40 minutes, plus 20 minutes to complete the demo.

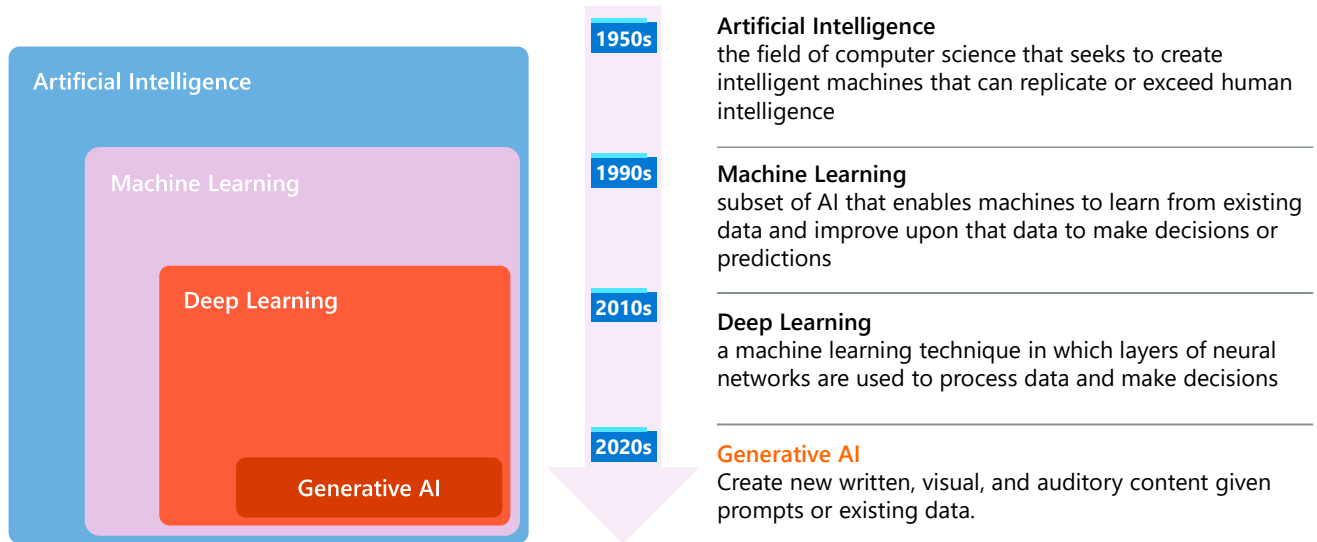
Learning Objectives

After completing this module, you will be able to:

- 1 Describe what generative AI is
- 2 Provision a resource and deploy a model
- 3 Use the Azure AI Foundry portal

© Copyright Microsoft Corporation. All rights reserved.

What is generative AI?



© Copyright Microsoft Corporation. All rights reserved.

Den här bilden berättar en fascinerande historia om hur artificiell intelligens (AI) har utvecklats genom decennier och lett fram till dagens generativa AI – en av de mest banbrytande teknologierna vi har.

Vi börjar på 1950-talet, där idén om AI föddes – drömmen om maskiner som kan tänka som människor. Forskare arbetade med att skapa datorprogram som kunde simulera mänsklig intelligens, men det var fortfarande mycket teoretiskt.

Sedan hoppar vi till 1990-talet, där maskininlärning (ML) börjar förändra spelplanen. Istället för att hårdkoda regler, lärde sig datorer genom data. De kunde hitta mönster och göra förutsägelser – något som snabbt revolutionerade allt från finans till sjukvård.

På 2010-talet exploderar deep learning, tack vare framsteg inom neurala nätverk och ökad datorkraft. Här börjar AI verkligen bli kraftfull, med system som kan förstå bilder, text och ljud på ett sätt som liknar mänsklig perception. Deep learning banar väg för avancerad bildigenkänning, självstyrande bilar och röstassistenter som Siri och Alexa.

Och nu är vi i 2020-talet, där generativ AI tar världen med storm. Det handlar inte bara om att förstå och analysera data – nu kan AI skapa! Från att skriva texter och komponera musik till att generera bilder och videor, generativ AI ger oss en ny typ av kreativitet som tidigare var reserverad för människor.

Bilden visar snyggt hur generativ AI är en del av denna evolution, byggd på lager av innovation. Det är som en rysk docka där AI innehåller ML, ML innehåller deep learning, och i kärnan av allt – generativ AI, en teknik som förändrar hur vi skapar, arbetar och kommunicerar.

Framtiden? AI som inte bara hjälper oss att fatta beslut, utan som samarbetar med oss som en kreativ partner. Och vi är bara i början.

The slide presents an approximate timeline of the evolution of artificial intelligence, not everything is exactly in the decades shown. Each major milestone in the evolution builds on the previous ones.

Note that *most* (but not all) generative AI models are based on deep learning training techniques, so we show generative AI as a subset of deep learning.

Introduce Azure OpenAI as a type of generative model, which you're diving into today. Don't worry about spending much time on this slide.

Provision an Azure OpenAI resource in Azure

[Home](#) > [Azure AI services](#) | [Azure OpenAI](#) >

Create Azure OpenAI

Create an Azure OpenAI resource in the Azure portal

Deploy a resource in Azure, then deploy and use a model in Azure AI Foundry

Alternatively, use the Azure CLI

```
az cognitiveservices account create \  
-n MyOpenAIResource \  
-g MyResourceGroup \  
-l eastus \  
--kind OpenAI \  
--sku s0 \  
--subscription subscriptionID
```

1 Basics 2 Network 3 Tags 4 Review + submit

Enable new business solutions with OpenAI's language generation capabilities powered by GPT-3 models. These models have been pretrained with trillions of words and can easily adapt to your scenario with a few short examples provided at inference. Apply them to numerous scenarios, from summarization to content and code generation.

[Learn more](#)

Project Details

Subscription * ⓘ

Resource group * ⓘ

[Create new](#)

Instance Details

Region ⓘ

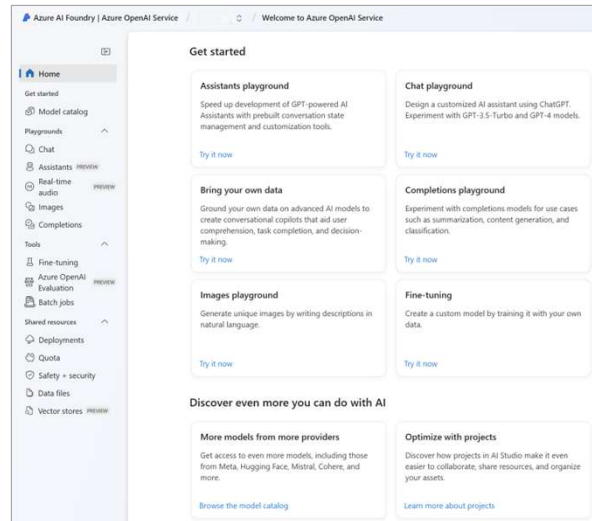
Name * ⓘ

Pricing tier * ⓘ

© Copyright Microsoft Corporation. All rights reserved.

Azure AI Foundry portal

- Web portal for working with AI models: <https://ai.azure.com/>
- View and deploy base models
- Connect to other AI Services or your own data source
- Manage fine tuning and data files for custom models
- Test models in visual playgrounds:
 - **Chat** (GPT-3.5-Turbo and later models)
 - **Assistants** (Custom and Copilot-like experiences)
 - **Images** (DALL-E)
 - **Completions** (GPT-3)



© Copyright Microsoft Corporation. All rights reserved.

Azure AI Foundry är Microsofts webbaserade plattform för att arbeta med AI-modeller. Bilden visar en översikt av dess funktioner och hur du kan använda den för att experimentera med AI på olika sätt.

Vad är Azure AI Foundry?

Det är en portal där du kan:

- Se och distribuera grundmodeller (som GPT-3.5, GPT-4, DALL-E, m.fl.).
- Ansluta till andra AI-tjänster eller ladda upp din egen data för att anpassa modeller.
- Finjustera modeller med egna träningsdata.
- Testa modeller i interaktiva sandlådor

(Playgrounds) för olika AI-applikationer.

Vilka AI-funktioner kan du testa i portalen?

Portalen erbjuder olika "playgrounds" där du kan experimentera med AI direkt:

- Chat:** Bygg en anpassad AI-assistent med GPT-3.5-Turbo eller nyare.
- Assistants:** Skapa egna AI-assistenter, liknande Microsoft Copilot.
- Images:** Generera bilder med DALL-E utifrån textbeskrivningar.
- Completions:** Testa textgenerering, sammanfattningar och klassificeringar med GPT-3.

Varför är detta intressant?

Med Azure AI Foundry får du en centraliserad plattform för att utveckla AI-baserade applikationer utan att behöva sätta upp en massa infrastruktur själv. Du kan anpassa modeller med din egen data, experimentera och iterera snabbt.

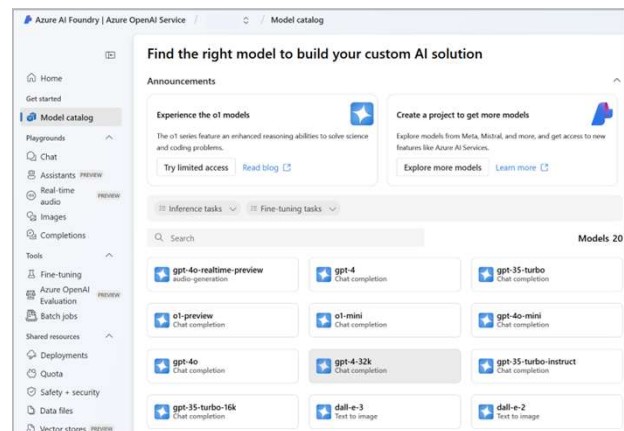
Vem är detta för?

Det passar utvecklare, data scientists och företag som vill bygga eller integrera AI i sina applikationer. Oavsett om du vill skapa en chatbot, generera bilder eller bygga avancerade Copilot-liknande lösningar, så erbjuder plattformen de verktyg du behöver.

Azure AI Foundry provides a web-based portal in which you can manage the models and deployments in your Azure OpenAI resource. You can access it from the resource page in the Azure portal or directly at <https://ai.azure.com/>.

Types of generative AI model

Model Family	Description
Base GPT	Collection of the most capable chat-based models for language and code generation, such as GPT-4
Multimodal	Generative models that combine more than one modality, such as the Whisper model
Embeddings	Models that use embeddings for specific tasks (similarity, text search, and code search)
Images	Image-generation model (such as DALL-E 3, which is generally available for use with the REST APIs)



© Copyright Microsoft Corporation. All rights reserved.

Här är en översikt av de olika generativa AI-modellfamiljerna med fokus på OpenAI:s modeller, som visas i bilden.

Generativa AI-modellfamiljer

Det finns många olika AI-modeller från olika aktörer, som Mistral, Meta, Microsoft och OpenAI. Men här fokuserar vi på OpenAI:s modeller, som används i Azure AI Foundry.

1. Base GPT-modeller (Språk och kodgenerering)

Dessa modeller är specialiserade på textgenerering, kodskrivning och konversationer.

- GPT-4o och GPT-4o-mini:** De nyaste och mest avancerade modellerna för språk och kodgenerering. GPT-4o erbjuder förbättrad prestanda och kostnadseffektivitet jämfört med tidigare generationer.
- GPT-3.5-Turbo:** En optimerad version av GPT-3.5, känd från den ursprungliga ChatGPT-tjänsten. Används ofta för kostnadseffektiv textgenerering och chatbot-tillämpningar.
- Äldre GPT-3-modeller** (som Davinci, Curie, Babbage, Ada) används numera mest som bas för **Embeddings** (se nedan).

2. Multimodala modeller (Flera modaliteter: text, ljud, bild)

Den här familjen kombinerar flera olika typer av data, såsom ljud och text.

- Whisper:** En modell för automatisk taligenkänning (speech-to-text).
- GPT-4o:** Har vissa multimodala funktioner och kan hantera både text och bilder.

3. Embeddings-modeller (Text- och kodsökning, likhetsanalys)

Embeddings är specialiserade AI-modeller för att analysera text- och kodrelationer.

- Används för att hitta likheter mellan texter, förbättra sökresultat och utföra kodsökning.**
- Snabbare än fullständiga GPT-modeller** och optimerade för uppgifter som att jämföra text.

4. Bildgenereringsmodeller (Text-till-bild med DALL-E)

Dessa modeller skapar bilder utifrån textbeskrivningar.

- DALL-E 3:** Den senaste och mest avancerade versionen, tillgänglig via REST API.
- DALL-E 2:** En tidigare version, fortfarande tillgänglig i vissa tjänster.

Sammanfattning

- Base GPT** = Text och kodgenerering (GPT-4o, GPT-3.5-Turbo)

- **Multimodal** = Modeller som hanterar flera modaliteter (exempel: Whisper)
- **Embeddings** = Optimerad text- och kodanalys
- **Images** = Bildgenerering med DALL-E

Om du vill bygga AI-lösningar i Azure AI Foundry, är det här de centrala modellfamiljerna du kan arbeta med.

Mention that there are lots of various models (Mistral, Meta, Microsoft, OpenAI etc), but this is focusing on OpenAI models. There are other courses covering the full range of models.

Base GPT models

The newer GPT-4 models are generally the best choice for all language and code generation tasks. This includes models like 4o and 4o-mini

The GPT-3 family includes the GPT-3.5-Turbo model on which the original release of the OpenAI ChatGPT service is based, and which supports the Chat API (and corresponding playground).

****Info for trainer, leave out unless asked -** This also includes four older generations of language model, versions from A (Ada) to D (Davinci). Older models are less complex than newer ones (they're smaller and consist of fewer layers). The right choice for a specific tasks depends on a balance of the speed at which the model can perform (smaller models tend to be faster) and how sophisticated their completions need to be (larger models tend to be more capable than smaller ones). These older models are what the Embeddings are based on.

Multimodal

This family is a collection of all the multimodal models, such as Whisper or Text-to-speech. There is some overlap with the GPT models, such as GPT-4o which has some multimodal functionality depending on the model

Embeddings

Embeddings-based models make use of embeddings (a specialized form of data encoding for text) to optimize specific tasks (evaluating text for similarities, searching text, and searching code)

Images

DALL-E is a model for image-generation based on natural language prompts. **DALL-E 3 is generally available for use with the REST APIs. DALL-E 2 and DALL-E 3 with client SDKs are in preview.**

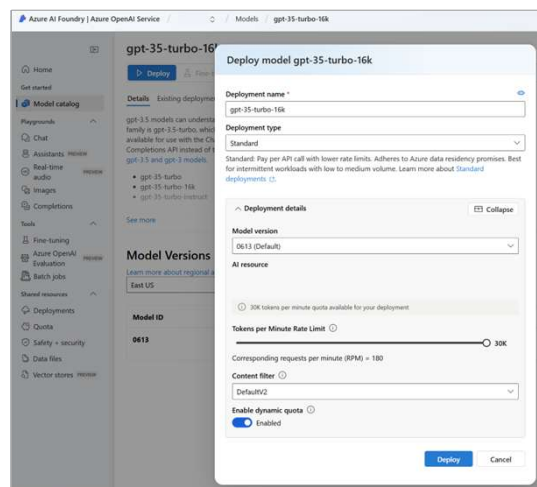
Not included here are models like o1 for advanced reasoning – these are limited preview so you can discuss if asked, but don't need to include it quite yet.

Deploying generative AI models

Deploy a model in Azure AI Foundry to use it

- You can deploy one or more instances of each available model
- The number of deployments depends on your quota, which you can see in the portal
- Alternatively, use the Azure CLI

```
az cognitiveservices account deployment create \
  -g myResourceGroup \
  -n MyOpenAIResource \
  --deployment-name my-gpt-model \
  --model-name gpt-35-turbo-16k \
  --model-version "0613" \
  --model-format OpenAI \
  --sku Standard \
  --sku-capacity 10
```



© Copyright Microsoft Corporation. All rights reserved.

Using prompts to get completions from models

Task	Prompt	Completion
Classifying content	Tweet: I enjoyed the training course. Sentiment:	Positive
Generating new content	Write a poem about databases	Databases, oh databases, You keep our information safe, From the small to the large, You store our data in a place.
Transformation/Translation	English: Hello French:	Bonjour
Summarization	Scotland is [long description of Scotland...] Summarize the previous text	Scotland is [summarized description...]
Continuation	One way to grow tomatoes is to	start with seeds...
Question answering	How many moons does Earth have?	Earth has one moon.
Chat	Setup, followed by messages...	A sequence of relevant responses

© Copyright Microsoft Corporation. All rights reserved.

Här är en genomgång av de olika uppgifterna (tasks) i bilden och vad de innebär:

1. Classifying content (Klassificering av innehåll)

1. **Exempel:** "Tweet: I enjoyed the training course. Sentiment:"
2. **Slutresultat:** "Positive"
3. Här används en modell för att analysera och kategorisera innehållet, t.ex. sentimentanalys (positiv, neutral, negativ).

2. Generating new content (Generera nytt innehåll)

1. **Exempel:** "Write a poem about databases"
2. **Slutresultat:** Ett kort poem om databaser.
3. AI genererar text från en given uppmaning (prompt), i detta fall en dikt om databaser.

3. Transformation/Translation (Transformation/Översättning)

1. **Exempel:** "English: Hello, French:"
2. **Slutresultat:** "Bonjour"
3. AI omvandlar eller översätter text från ett språk till ett annat.

4. Summarization (Sammanfattning)

1. **Exempel:** "Scotland is [long description of Scotland...]"
2. **Slutresultat:** "Scotland is [summarized description...]"
3. AI tar en längre text och kondenserar den till en kortare och mer överskådlig form.

5. Continuation (Fortsättning av text)

1. **Exempel:** "One way to grow tomatoes is to"
2. **Slutresultat:** "start with seeds..."
3. AI fortsätter en påbörjad mening eller text.

6. Question answering (Frågor och svar)

1. **Exempel:** "How many moons does Earth have?"
2. **Slutresultat:** "Earth has one moon."

3. AI besvarar faktabaserade frågor baserat på kunskap eller tillgängliga data.

1.Chat (Chattbaserad interaktion)

1. **Exempel:** "Setup, followed by messages..."
2. **Slutresultat:** "A sequence of relevant responses"
3. AI upprätthåller en konversation genom att svara på meddelanden på ett sammanhängande sätt.

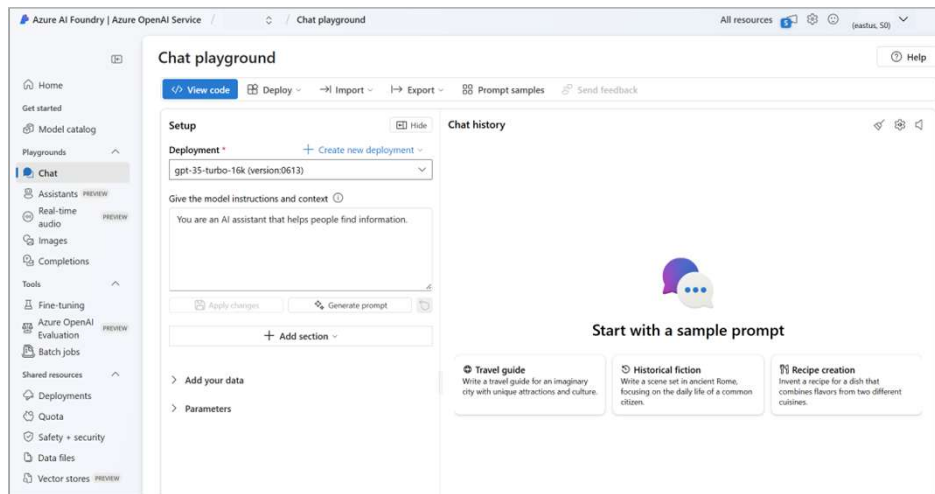
Sammanfattning

Bilden visar hur olika AI-modeller används för att generera text, klassificera innehåll, sammanfatta, översätta, besvara frågor och upprätthålla konversationer. Varje typ av uppgift har sina specifika användningsområden, från sentimentanalys till maskinöversättning och frågesvarssystem.

The core way to interact with a model in Azure OpenAI is to provide a natural language *prompt* as input. Models then respond with a *completion*. The completion depends on a number of factors, including the text in the prompt, the way the prompt is presented (including any *prompt engineering* techniques applied), and the *parameters* that are set for the interaction (discussed on the next slide). This slide shows some examples of common types of task that you can accomplish with a generative model in Azure OpenAI service.

Chat is a more advanced scenario that is supported by more recent models (GPT-35-turbo and later), in which a *setup* statement is used to define the chat context and then the interaction consists of conversational "turns" in which the user submits messages to which the model returns relevant responses..

Testing models in Azure AI Foundry playground



© Copyright Microsoft Corporation. All rights reserved.

Självklart! Här är en enkel förklaring av varje del:

Playground (Testmiljön)

Playground är ett verktyg där du kan skriva in kommandon (prompts) och justera inställningar för hur AI-modellen ska svara.

Viktiga inställningar du kan göra:

1. Maximalt svar (Max response)

1. Sätter en gräns för hur långt svaret kan vara.
2. Modellen kan hantera totalt 4000 "tokens" (ungefär 4 tecken per token för vanlig engelsk text), inklusive både fråga och svar.

2. Slumpmässighet (Temperature)

1. **Låg temperatur (t.ex. 0.2)** → Mer förutsägbara och upprepande svar.
2. **Hög temperatur (t.ex. 0.8)** → Mer kreativa och oväntade svar.
3. Rekommendation: Justera **antingen** Temperature eller Top P, inte båda.

3. Top P

1. Liknar Temperature men fungerar på ett annat sätt.
2. **Låg Top P** → Modellen väljer mer säkra och vanliga ord.
3. **Hög Top P** → Modellen kan välja mer ovanliga ord.

4. Stop-sekvenser

1. Låter dig bestämma **när** AI ska sluta generera text.
2. Exempel: Du kan ange att svaret ska sluta vid en punkt, ny rad eller vid ett visst ord.

5. Frekvensstraff (Frequency penalty)

1. Minskar sannolikheten att modellen **upprepar samma ord eller fras** för ofta.

6. Närvarostraff (Presence penalty)

1. Gör att modellen **pratar om nya ämnen** istället för att upprepa sig.

7. Tidigare meddelanden (Past messages included)

1. Anger hur många tidigare meddelanden som ska inkluderas för att modellen ska komma ihåg

kontext.

2. Exempel: Sätter du den till 10, så sparas 5 frågor och 5 svar för att ge mer sammanhängande konversationer.

Sammanfattning

- Vill du ha exakta svar? → Sänk **Temperature** och **Top P**.
- Vill du ha kreativa svar? → Höj **Temperature** eller **Top P**.
- Vill du undvika upprepningar? → Justera **Frequency penalty** och **Presence penalty**.
- Vill du att AI:n ska "komma ihåg" samtalet? → Öka **Past messages included**.

Det är bara att testa sig fram och se vad som fungerar bäst! 😊

The playground enables you to enter prompts and to set *parameters* that control the completions returned by the model.

When using the Chat playground, you can set the context or define a system message for the chat session; and also set the following parameters:

- **Max response:** Set a limit on the number of tokens per model response. The API supports a maximum of 4000 tokens shared between the prompt (including system message, examples, message history, and user query) and the model's response. One token is roughly four characters for typical English text.
- **Temperature:** Controls randomness. Lowering the temperature means that the model produces more repetitive and deterministic responses. Increasing the temperature results in more unexpected or creative responses. Try adjusting temperature or Top P but not both.
- **Top P:** Similar to temperature, this controls randomness but uses a different method. Lowering Top P narrows the model's token selection to likelier tokens. Increasing Top P lets the model choose from tokens with both high and low likelihood. Try adjusting temperature or Top P but not both.
- **Stop sequences:** Make responses stop at a desired point, such as the end of a sentence or list. Specify up to four sequences where the model will stop generating further tokens in a response. The returned text won't contain the stop sequence.
- **Frequency penalty:** 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.
- **Presence penalty:** 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.
- **Past messages included:** Select the number of past messages to include in each new API request. Including past messages helps give the model context for new user queries. Setting this number to 10 will include five user queries and five system responses.

Demo: Get started with Azure OpenAI Service



Deploy Azure OpenAI resource and model

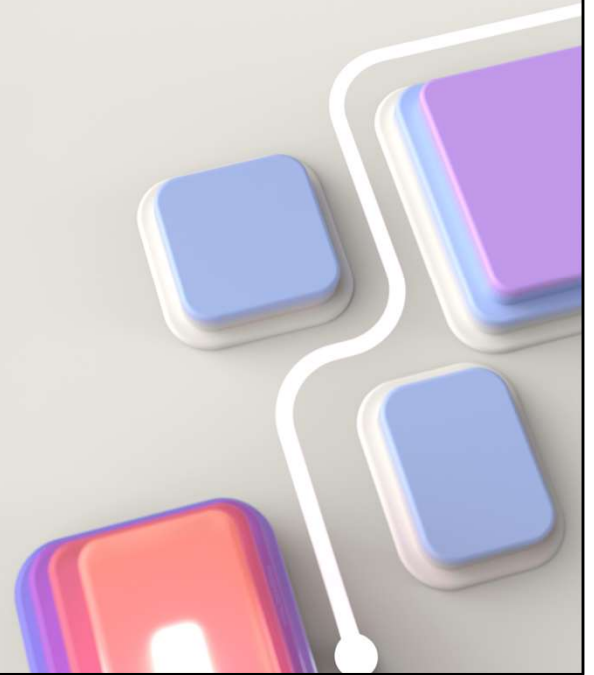
Test deployed model in playground

© Copyright Microsoft Corporation. All rights reserved.

IMPORTANT: Show how to decrease quota when deploying your model, as students will be doing so in their labs.

Also, if you or the students find the model they're told to deploy does not appear available, have them delete the Azure OpenAI resource and recreate it in one of the regions that has the correct model. Model availability can be found here: [Azure OpenAI Service models - Azure OpenAI | Microsoft Learn](#)

Develop apps with Azure OpenAI Service



The goal of this section is to teach developers how to integrate Azure OpenAI models into their applications by using the REST APIs and Python and C# SDKs

The estimated time to present these slides is 40 minutes, plus 30 minutes to complete the interactive exercise.

Learning Objectives

After completing this module, you will be able to:

- 1 Integrate Azure OpenAI into your app
- 2 Use the REST API
- 3 Use language specific SDKs

© Copyright Microsoft Corporation. All rights reserved.

Integrating Azure OpenAI into your app

Applications submit prompts to deployed models. Responses are completions.

Three REST API endpoints:

- **Completion** - model takes an input prompt, and generates one or more predicted completions
- **Embeddings** - model takes input and returns a vector representation of that input
- **ChatCompletion** - model takes input in the form of a chat conversation (where roles are specified with the message they send), and the next chat completion is generated

ChatCompletion will be the endpoint we focus on for this course

Use **Completion** with GPT-3 based models
 Use **Embeddings** with text-embedding models
 Use **ChatCompletion** with GPT-3.5-Turbo and later models

© Copyright Microsoft Corporation. All rights reserved.

Den här bilden handlar om hur du kan använda **Azure OpenAI** i en app genom att skicka frågor (prompts) till AI-modeller och få svar.

Tre olika API-endpoints

1.Completion – Skickar in en text och får tillbaka en eller flera förslag på fortsättning.

Används med GPT-3-baserade modeller.

2.Embeddings – Skickar in en text och får en vektor som representerar texten.

Används för att jämföra texter, söka i dokument eller analysera likheter.

3.ChatCompletion – Skickar in ett chattsamtal med rollspecifika meddelanden och får ett svar.

Används med GPT-3.5-Turbo och nyare modeller (som GPT-4o).

Fokus på ChatCompletion

Bilden betonar att **ChatCompletion** är den API-metod som används i kursen, eftersom den är optimerad för konversationer.

Sammanfattning

•Vill du bara ha en textkomplettering? → Använd **Completion**.

•Vill du analysera textlikhet? → Använd **Embeddings**.

•Vill du ha en interaktiv chatt? → Använd **ChatCompletion**.

Det här är grunden för att bygga AI-drivna appar i Azure! 🚀

The multiple REST endpoints reflect the evolution of OpenAI models.

In most cases for new applications, you should use a GPT-3.5-Turbo or later model and use the **ChatCompletion** endpoint.

GPT-4 model only support the **ChatCompletion** endpoint.

Using the Azure OpenAI REST API

Completion Endpoint

<https://endpoint.openai.azure.com/openai/deployments/deployment/completions>

```
{
  "prompt": "Your favorite Shakespeare
            play is",
  "max_tokens": 5
}
```



```
{
  "id": "1234...",
  "object": "text_completion",
  "created": 1679001781,
  "model": "gpt-35-turbo",
  "choices": [
    {
      "text": "Macbeth",
      "index": 0,
      "logprobs": null,
      "finish_reason": "stop"
    }
  ]
}
```

© Copyright Microsoft Corporation. All rights reserved.

Förfrågan (Request) – Vänster sida

Detta är den JSON-struktur vi skickar till API:t:

{ "prompt": "Your favorite Shakespeare play is", "max_tokens": 5 } **Beskrivning av varje ord:**

- **"prompt"** → En nyckel i JSON som innehåller den fråga eller text vi skickar till AI-modellen.
- **"Your favorite Shakespeare play is"** → Den faktiska texten vi skickar till AI:n som input. Här försöker vi få modellen att nämna en pjäs av Shakespeare.
- **"max_tokens"** → Anger det maximala antalet tokens (textfragment) som AI:n får använda i sitt svar.
- **5** → Sätter en gräns på 5 tokens för att förhindra att svaret blir för långt.

Svaret (Response) – Höger sida

Detta är den JSON-struktur som API:t returnerar:

{ "id": "1234...", "object": "text_completion", "created": 1679001781, "model": "gpt-35-turbo", "choices": [{ "text": "Macbeth", "index": 0, "logprobs": null, "finish_reason": "stop" }] } **Beskrivning av varje ord:**

- **"id"** → Ett unikt identifieringsnummer för denna API-förfrågan.
- **"object"** → Typen av svar som API:t returnerar. Här är det en "text_completion", vilket betyder att modellen har fyllt i en text.
- **"created"** → En tidsstämpel (Unix-tid) som visar när svaret genererades.
- **"model"** → Namnet på AI-modellen som användes, här "gpt-35-turbo", en snabb version av GPT-3.5.
- **"choices"** → En lista (array) där olika möjliga svar kan finnas.

Detaljer i choices (där själva svaret finns)

- **"text"** → AI:ns genererade svar. Här är det "Macbeth" – en pjäs av Shakespeare.
- **"index"** → Indikator för att det är det första (och enda) svaret i listan, därför "0".
- **"logprobs"** → (Log probabilities) Normalt används detta för att se hur sannolikt modellen trodde att just detta svar var. Här är det "null", vilket betyder att loggarna inte är inkluderade i svaret.
- **"finish_reason"** → Anledningen till att modellen slutade generera text. "stop" betyder att modellen ansåg att svaret var färdigt inom den angivna token-begränsningen.

Sammanfattning

1. Vi skickar in en fråga ("prompt": "Your favorite Shakespeare play is") med en begränsning på 5 tokens.
2. AI:n svarar med "Macbeth", eftersom det är en känd pjäs av Shakespeare.
3. API:t returnerar en JSON-struktur med information om svaret, vilken modell som användes, och varför texten stoppades.

Det här är grunden för att arbeta med Azure OpenAI:s **Completion API** i en app! 🚀

Using the Azure OpenAI REST API

Embedding Endpoint

<https://endpoint.openai.azure.com/openai/deployments/deployment/embeddings>

```
{
  "input": "The food was delicious and
           the waiter was very
           friendly..."
}
```



```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "embedding": [
        0.0172990688066482523,
        ....
        0.0134544348834753042,
      ],
      "index": 0
    }
  ],
  "model": "text-embedding-ada:002"
}
```

© Copyright Microsoft Corporation. All rights reserved.

Självklart! Här är en detaljerad genomgång av koden i bilden, där jag förklarar varje del.

Förfrågan (Request) – Vänster sida

Detta är JSON-strukturen vi skickar till API:t:

`{ "input": "The food was delicious and the waiter was very friendly..." }` **Beskrivning av varje ord:**

- **"input"** → Nyckeln som innehåller texten vi vill bearbeta.
- **"The food was delicious and the waiter was very friendly..."** → Den text som vi skickar till modellen för att omvandlas till en numerisk representation (embedding).

Svaret (Response) – Höger sida

Detta är JSON-strukturen som API:t returnerar:

`{ "object": "list", "data": [{ "object": "embedding", "embedding": [0.0172990688066482523, ..., 0.0134544348834753042], "index": 0 },], "model": "text-embedding-ada-002" }` **Beskrivning av varje ord:**

- **"object": "list"** → API:t returnerar en lista med resultat.
- **"data"** → En lista som innehåller embedding-värdena.
- Detaljer i data (där den faktiska embeddingen finns)**
 - **"object": "embedding"** → Indikerar att objektet är en embedding.
 - **"embedding"** → En lista med decimaltal som representerar texten i en matematisk form.
 - **[0.0172990688066482523, ..., 0.0134544348834753042]** → En lista av numeriska värden (vektorer) som beskriver textens betydelse.
 - **"index": 0** → Indikerar vilken position detta embedding-objekt har i svaret.

Övriga fält

- **"model": "text-embedding-ada-002"** → Anger vilken AI-modell som användes för att skapa embeddingarna. Här används "text-embedding-ada-002", en modell specialiserad på att omvandla text till numeriska representationer.

Vad gör detta API?

- Den tar en text (t.ex. en kundrecension) och omvandlar den till en vektor (en lång lista med siffror).
- Denna vektor kan sedan användas för **sökningar, jämförelser eller maskininlärning**.
- **Exempel:** Man kan använda detta för att hitta texter som liknar varandra, även om de inte har exakt samma ord.

Sammanfattning

- 1.Vi skickar in en text ("The food was delicious...").
 - 2.AI:n omvandlar texten till en embedding – en numerisk representation.
 - 3.API:t returnerar en lista av siffror som beskriver textens innehåll.
 - 4.Denna embedding kan användas för att jämföra och analysera text på ett smart sätt!
- Det här är grunden för AI-driven sökning och textanalys i Azure OpenAI! 🚀

Using the Azure OpenAI REST API

ChatCompletion Endpoint

<https://endpoint.openai.azure.com/openai/deployments/deployment/chat/completions>

```
{
  "messages": [
    { "role": "system",
      "content": "You are an assistant
that teaches people about AI." },
    { "role": "user",
      "content": "Does Azure OpenAI
support multiple languages?" },
    { "role": "assistant",
      "content": "Yes, Azure OpenAI
supports several languages." },
    { "role": "user",
      "content": "Do other Cognitive
Services support translation?" }
  ]
}
```



```
{
  "id": "unique_id", "object": "chat.completion",
  "created": 1679001781, "model": "gpt-35-turbo",
  "usage": { "prompt_tokens": 95,
             "completion_tokens": 84, "total_tokens": 179 },
  "choices": [
    { "message":
      { "role": "assistant",
        "content": "Yes, other Azure Cognitive
Services also support translation..." },
      "finish_reason": "stop",
      "index": 0 }
  ]
}
```

© Copyright Microsoft Corporation. All rights reserved.

Ja! I **ChatCompletion API** används olika roller för att styra hur modellen ska bete sig i en konversation. Här är rollerna i exemplet:

1. **system** – Ger modellen instruktioner om hur den ska bete sig.

1. **Exempel i bilden:** {"role": "system", "content": "You are an assistant that teaches people about AI."}
2. **Funktion:** Sätter ramen för samtalet – i detta fall instrueras modellen att agera som en AI-lärare.

2. **user** – Representerar användarens inmatning (frågor eller kommandon).

1. **Exempel i bilden:** {"role": "user", "content": "Does Azure OpenAI support multiple languages?"}
2. **Funktion:** Användaren ställer frågor eller ger kommandon.

3. **assistant** – AI-modellens svar.

1. **Exempel i bilden:** {"role": "assistant", "content": "Yes, Azure OpenAI supports several languages."}
2. **Funktion:** Genererar ett svar baserat på användarens fråga och systeminstruktionen.

Sammanfattning

- **system** = Sätter regler för modellen.
- **user** = Ställer frågor eller ger input.
- **assistant** = Genererar svar baserat på användarens frågor.

Det här formatet hjälper modellen att hålla sig inom rätt kontext och svara mer naturligt i en chatt! 🚀

The side shows an example input and response for the **ChatCompletion** API endpoint.

Note that the input can include the previous messages in the conversation to provide additional context for the completion.

Mention that this is what we'll be focusing on as the newest models only work with ChatCompletions. If they want to learn more about embeddings, suggest going on Learn – that's too specific to be covered in this course.

Using language specific SDKs

Language specific SDKs are available for use in your applications, such as Azure OpenAI's .NET library and OpenAI's Python library.

Code structure follows a similar pattern for both languages.

Parameters such as *Max Response* and *Temperature* are defined in the chat options.

Both synchronous and asynchronous API versions are available.

För exempel, se nästa bild!

Pseudo code structure:

```
<include library>

<create client>

<define chat messages and options>

<send request>

<extract response content>
```

© Copyright Microsoft Corporation. All rights reserved.

Den här bilden handlar om hur man använder **språksspecifika SDK:er** (Software Development Kits) för att integrera Azure OpenAI i en applikation. SDK:er finns för flera språk, exempelvis **.NET** och **Python**, och de följer en liknande kodstruktur.

Vad gör SDK:erna?

- De förenklar anrop till OpenAI:s API genom att erbjuda färdiga funktioner.
- De stöder både **synkrona** och **asynkrona** API-anrop.
- Man kan definiera inställningar som **Max Response** och **Temperature** för att styra AI:ns beteende.

Kodexempel i Python

Här är ett exempel på hur man använder **OpenAI:s Python-SDK** för att göra ett **ChatCompletion-anrop**:

```
import openai # Skapa en klient med API-nyckeln client = openai.AzureOpenAI( api_key="din-api-nyckel",
azure_endpoint="https://ditt-endpoint.openai.azure.com/", api_version="2023-05-15" ) # Definiera chat-
meddelanden messages = [ {"role": "system", "content": "You are a helpful assistant."}, {"role": "user", "content":
"What is the capital of Sweden?"} ] # Skicka förfrågan response = client.chat.completions.create( model="gpt-35-
turbo", messages=messages, max_tokens=50, temperature=0.7 ) # Skriv ut svaret
print(response.choices[0].message.content)
```

Kodexempel i C# (.NET)

Om du använder **Azure OpenAI med .NET SDK**, kan koden se ut så här:

```
using Azure; using Azure.AI.OpenAI; using System; using System.Threading.Tasks; class Program { static async
Task Main() { var client = new OpenAIClient( new Uri("https://ditt-endpoint.openai.azure.com/"), new
AzureKeyCredential("din-api-nyckel") ); var chatCompletionsOptions = new ChatCompletionsOptions() {
Messages = { new ChatMessage(ChatRole.System, "You are a helpful assistant."), new
ChatMessage(ChatRole.User, "What is the capital of Sweden?" ) }, MaxTokens = 50, Temperature = 0.7 }; var
response = await client.GetChatCompletionsAsync("gpt-35-turbo", chatCompletionsOptions);
Console.WriteLine(response.Value.Choices[0].Message.Content); } }
```

Sammanfattning
1. Python och .NET SDK:erna gör det enkelt att anropa OpenAI:s API.

2. Strukturen är densamma i båda språken:

1. Skapa klient
2. Definiera chattmeddelanden
3. Skicka förfrågan

4. Hämta och visa svar

1. Man kan anpassa modellen genom att justera inställningar som **Temperature** (slumpmässighet) och **MaxTokens** (maximalt antal ord i svaret).

This pseudocode shows the general structure of using the SDK. This will vary slightly with SDK versions.

Highly recommend trainers show actual code samples in class, but they are no longer included here due to frequent changes.

Python-code



```
import openai

# Skapa en klient med API-nyckeln
client = openai.AzureOpenAI(
    api_key="din-api-nyckel",
    azure_endpoint="https://ditt-endpoint.openai.azure.com/",
    api_version="2023-05-15"
)

# Definiera chat-meddelanden
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "What is the capital of Sweden?"}
]

# Skicka förfrågan
response = client.chat.completions.create(
    model="gpt-35-turbo",
    messages=messages,
    max_tokens=50,
    temperature=0.7
)

# Skriv ut svaret
print(response.choices[0].message.content)
```

© Copyright Microsoft Corporation. All rights reserved.

The exercise will take a minimum of 30 minutes to complete.

Not all students work at the same pace, so you should allow additional time as necessary for your class.

IMPORTANT: make sure students know to decrease their model quota, as indicated in the instructions. It is important to make sure all students can deploy a model in the lab environment, and they should never hit issues with models deployed with lower quota.

Also, if students find the model they're told to deploy does not appear available, have them delete the Azure OpenAI resource and recreate it in one of the regions that has the correct model. Model availability can be found here: [Azure OpenAI Service models - Azure OpenAI | Microsoft Learn](#)

Exercise: Integrate Azure OpenAI into your app



Configure your application to connect to Azure OpenAI

Test your app by sending a request to your deployed model

Use the hosted lab environment if provided, or view the lab instructions at the link below:

<https://aka.ms/mslearn-azure-openai-api>

© Copyright Microsoft Corporation. All rights reserved.

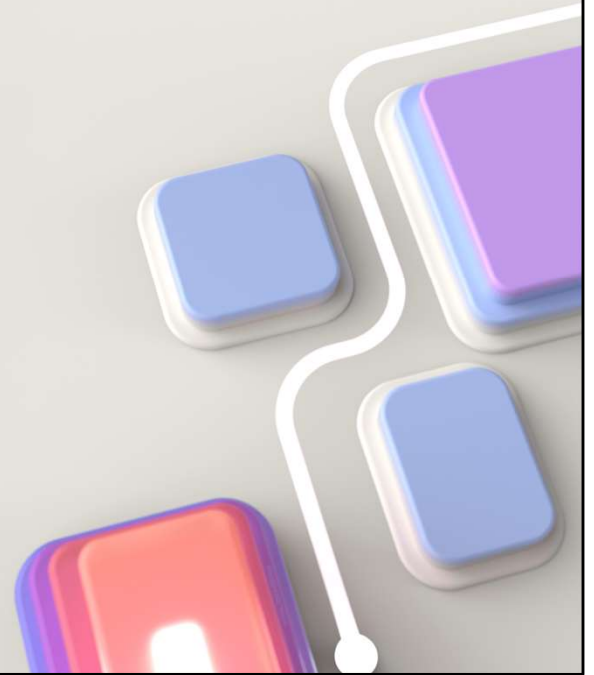
The exercise will take a minimum of 30 minutes to complete.

Not all students work at the same pace, so you should allow additional time as necessary for your class.

IMPORTANT: make sure students know to decrease their model quota, as indicated in the instructions. It is important to make sure all students can deploy a model in the lab environment, and they should never hit issues with models deployed with lower quota.

Also, if students find the model they're told to deploy does not appear available, have them delete the Azure OpenAI resource and recreate it in one of the regions that has the correct model. Model availability can be found here: [Azure OpenAI Service models - Azure OpenAI | Microsoft Learn](#)

Apply prompt engineering with Azure OpenAI Service



The goal of this section is to teach developers how to apply prompt engineering techniques with Azure OpenAI to improve the completions returned by models.

The estimated time to present these slides is 40 minutes, plus 30 minutes to complete the interactive exercise.

Learning Objectives

After completing this module, you will be able to:

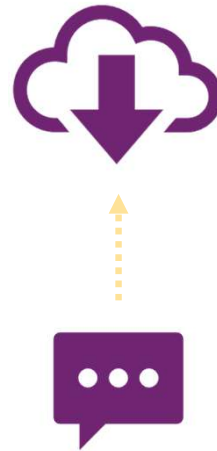
- 1 Understand what prompt engineering is
- 2 Understand considerations for different endpoints
- 3 Explore different techniques of prompt engineering

© Copyright Microsoft Corporation. All rights reserved.

What is Prompt Engineering?

Constructing prompts to:

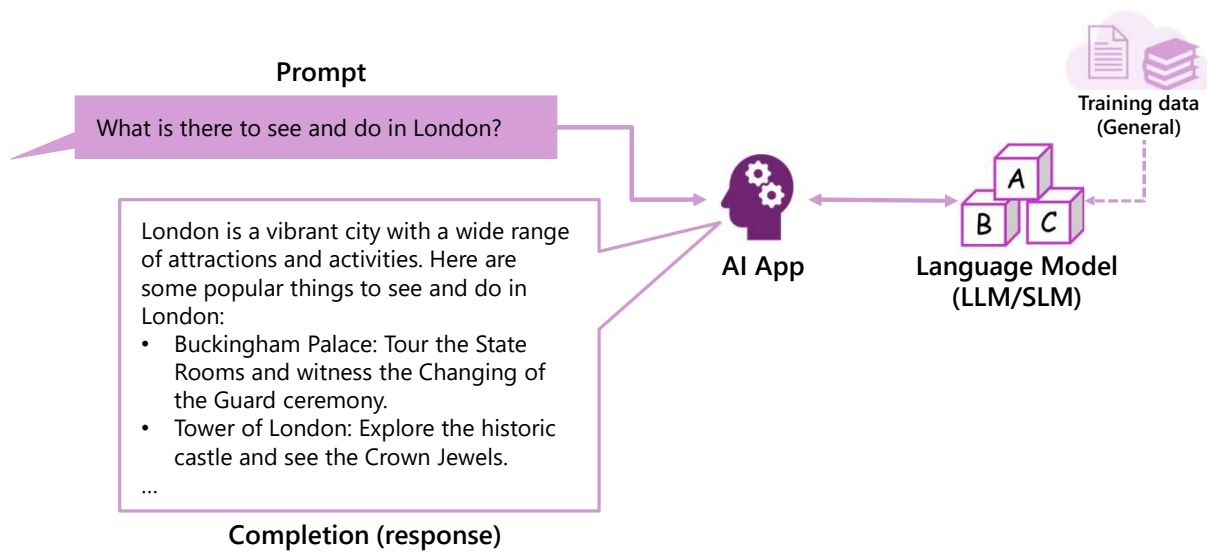
- Maximize relevancy and accuracy of completions
- Specify formatting and style of completions
- Provide conversational context
- Mitigate bias and improve fairness



© Copyright Microsoft Corporation. All rights reserved.

Prompt engineering is the term used to describe various techniques used to design a prompt for a large language model, such as a GPT model in Azure OpenAI service, to maximize the usefulness of the response.

Using a language model “out of the box”



Make note that the training data is just the general data used to train the model, nothing custom.

Using prompt engineering

Prompt

System: You are an AI travel agent that advises on sightseeing activities in major cities.
User: Give me ideas for things to see and do in London to help me plan a trip there. Include a numbered list of the top 10 places to visit in the city.

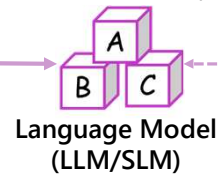
Certainly! Here's are the top 10 places to visit in London:

1. Buckingham Palace: Tour the State Rooms and witness the Changing of the Guard ceremony.
2. Tower of London: Explore the historic castle and see the Crown Jewels.
- ...

Completion (response)



AI App



Language Model (LLM/SLM)



Training data (General)

Explain that prompt engineering is simply designing the prompt to get a better response. Being more thoughtful and thorough can make a huge difference for the performance

Providing clear instructions

Write a product description for a new water bottle



Introducing the latest addition to our product line - the innovative and eco-friendly water bottle. Made from high-quality, BPA-free materials, this bottle is designed to keep your beverages at the perfect temperature for hours. Its sleek and stylish design makes it the perfect accessory for any outdoor activity or workout session. With its convenient carrying handle and leak-proof lid, you can take it with you wherever you go, without worrying about spills or leaks. Plus, its easy-to-clean design ensures that you can use it again and again, making it the perfect choice for anyone looking for a reliable and sustainable water bottle.

Write a product description for a new water bottle that is 100% recycled. Be sure to include that it comes in natural colors with no dyes, and each purchase removes 10 pounds of plastic from our oceans

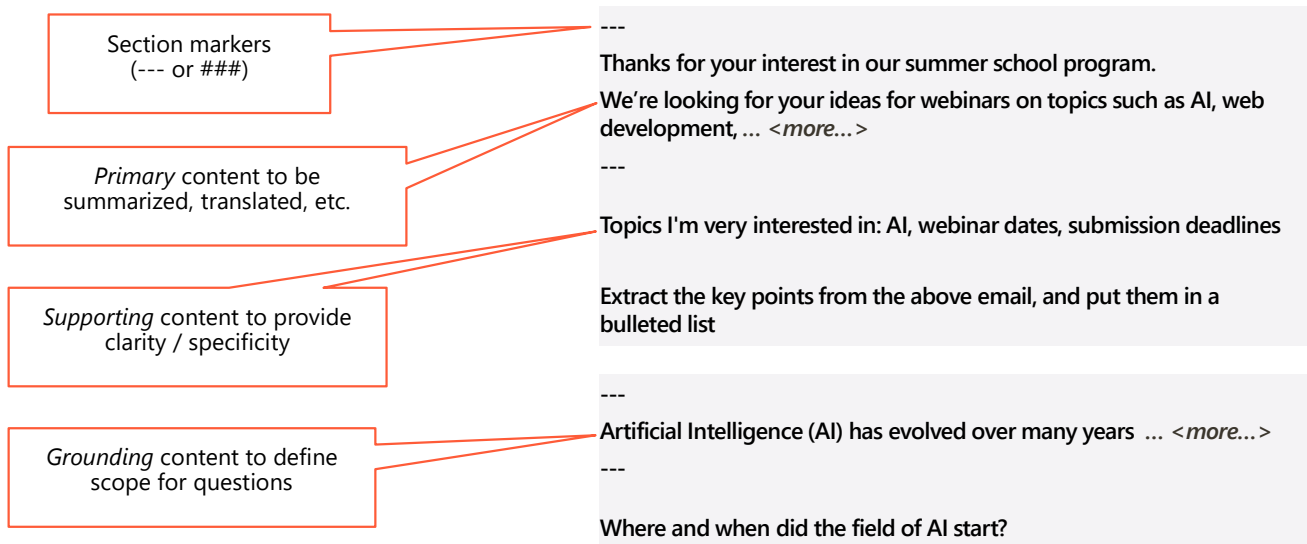


Introducing our newest water bottle, the eco-friendly choice for the conscious consumer. Made from 100% recycled materials and available in natural colors with no dyes, this bottle is not only stylish but also sustainable. Each purchase of this bottle removes 10 pounds of plastic from our oceans, making it a responsible choice for those who care about the environment. Its durable and leak-proof design makes it perfect for outdoor activities, while its easy-to-carry handle makes it convenient to take on the go. Join us in our mission to reduce plastic waste and choose this eco-friendly water bottle for your daily hydration needs.

© Copyright Microsoft Corporation. All rights reserved.

Compare the two examples on the slide. The example on the right uses a more descriptive prompt to ensure that the resulting completion is more relevant and includes the information required.

Primary, supporting, and grounding content



© Copyright Microsoft Corporation. All rights reserved.

Den här bilden förklarar hur **textinnehåll** kan delas upp i tre kategorier vid AI-bearbetning:

1.Primary content – Huvudinnehållet som ska sammanfattas, översättas eller bearbetas.

2.Supporting content – Ytterligare information som ger sammanhang och tydlighet.

3.Grounding content – Information som definierar ramarna för frågor och svar.

Bilden visar också att **sektioner markeras** med --- eller ### för att skilja olika typer av innehåll åt. Detta hjälper AI att förstå vad som är viktigt, vad som ger stöd och vad som sätter kontext för frågor. 🚀

Grounding content differs from primary content as it's the source of information to answer the prompt query, instead of the content being operated on for things like summarization or translation. If you need more reliable and current responses, or you need to reference unpublished or specific information, grounding content is highly recommended.

Cues

- Cues provide a starting point on which the completion builds
- Use cues to set the scope and format for the output
- Cues are particularly useful when generating code. For example:
 - Provide the first keyword in a SQL SELECT statement
 - Define a name and parameters for a Python function

```
---  
Review: The writing was good but the movie was too long.  
Review: The special effects were terrible.  
<more>...  
---  
Summarize the reviews above.  
Most common complaints:  
-
```



- The movie was too long.
- The special effects were terrible.

© Copyright Microsoft Corporation. All rights reserved.

Cues are leading words for the model to build upon, and often help shape the response in the right direction.

Requesting output composition

Write a table in markdown with 6 animals in it, with their genus and species



Animal	Genus	Species
Red Panda	Ailurus	fulgens
African Elephant	Loxodonta	africana
Snow Leopard	Panthera	uncia
Arctic Fox	Vulpes	lagopus
Chimpanzee	Pan	troglydytes
Siberian Tiger	Panthera	tigris altaica

© Copyright Microsoft Corporation. All rights reserved.

Specifying the structure of your output can have a large impact on your results. This could include something like asking the model to cite their sources, write the response as an email, format the response as a SQL query, classify sentiment into a specific structure, and so on.

Using a system message

When using the ChatCompletions API, a system message sets context that can affect completions

Kortfattning	Vad det gör	Exempel
Grounding	Hämtar fakta från externa källor för att säkerställa riktighet.	AI söker i en produktdatabas innan den svarar.
System Message	Bestämmer AI:ns beteende och roll.	"You are a helpful assistant."
User Message	Användarens fråga eller input.	"What is the capital of Sweden?"

```
{ "role": "system", "content": "You are a casual, helpful assistant.
  You will talk like an American old western film character." },
{ "role": "user", "content": "Can you direct me to the library?" }
```



```
{ "role": "assistant", "content": "Well howdy there, stranger! The library, huh?
  Y'all just head down the main road till you hit the town
  square. Once you're there, take a left and follow the street
  for a spell. You'll see the library on your right, can't
  miss it. Happy trails!" }
```

© Copyright Microsoft Corporation. All rights reserved.

Grounding, System Message och User Message

Koncept	Vad det gör	Exempel
Grounding	Hämtar fakta från externa källor för att säkerställa riktighet.	AI söker i en produktdataas innan den svarar.
System Message	Bestämmer AI:ns beteende och roll.	"You are a helpful assistant."
User Message	Användarens fråga eller input.	"What is the capital of Sweden?"

```
[
  {"role": "system", "content": "You are a customer service agent for a travel company."},
  {"role": "user", "content": "What is the cheapest flight from Stockholm to London?"},
  {"role": "assistant", "content": "Let me check for you..."}
]
```

© Copyright Microsoft Corporation. All rights reserved.

Conversation history and *few shot* learning

- Including previous messages helps retain the context and style of a conversation
- Starting a conversation with user-defined messages and responses (*few shot* learning) sets examples for the expected responses

```
{ "role": "system", "content": "You are an assistant that evaluates the sentiment of customer feedback." },  
{ "role": "user", "content": "That was an awesome experience" },  
{ "role": "assistant", "content": "positive" },  
{ "role": "user", "content": "I won't do that again" },  
{ "role": "assistant", "content": "negative" },  
{ "role": "user", "content": "That was not worth my time" },  
{ "role": "assistant", "content": "negative" },  
{ "role": "user", "content": "You can't miss this" }
```



```
{ "role": "assistant", "content": "positive" },
```

© Copyright Microsoft Corporation. All rights reserved.

Den här bilden handlar om **konversationshistorik** och **"few-shot learning"** i AI-modeller, särskilt hur man kan använda tidigare exempel för att styra modellens beteende.

Huvudpunkter:

1. Konversationshistorik

1. Genom att inkludera tidigare meddelanden i en chatt kan AI:n **behålla sammanhanget** och förstå samtalets stil.
2. Det gör att modellen kan svara mer konsekvent baserat på tidigare utbyten.

2. Few-shot learning

1. Istället för att träna om modellen kan man **ge exempel i konversationen** så att den lär sig hur den ska svara.
2. I detta fall får AI:n exempel på hur den ska analysera sentiment (positivt eller negativt) i användarfeedback.

JSON-exemplet på bilden:

• **Systemmeddelande** instruerar AI:n:

• `{ "role": "system", "content": "You are an assistant that evaluates the sentiment of customer feedback." }` → Modellen får rollen som en sentimentanalys-AI.

• **Användaren ger feedback**

• `{ "role": "user", "content": "That was an awesome experience" }` → Positivt omdöme.

• **AI:n lär sig svara rätt**

• `{ "role": "assistant", "content": "positive" }`

• **Negativa omdömen får rätt etikett**

• `{ "role": "user", "content": "I won't do that again" }` `{ "role": "assistant", "content": "negative" }`

• **Ny input: "You can't miss this"**

→ Baserat på tidigare exempel gissar AI:n att sentimentet är **"positive"**.

Sammanfattning:

• **Konversationshistorik** hjälper AI att minnas sammanhang.

- Few-shot learning** används för att lära AI att klassificera sentiment **utan omträning**.
- Resultatet:** AI kan nu analysera sentiment på nya meningar mer exakt.
Perfekt för sentimentanalys i kundfeedback, chattbotar och liknande! 🚀

Note that you can maintain the context and tone of a conversation by including previous messages in each prompt. You can also initialize a new conversation with some pre-prepared messages and responses to provide examples of the kind of output you want the model to generate; which is known as *few shot* learning.

Chain of thought

Ask a model to break down its response and explain its reasoning

What sport is easiest to learn but hardest to master? Give a step by step approach of your thoughts, ending in your answer

Step 1: Identify the criteria for "easy to learn" and "hard to master" sports.

For a sport to be considered easy to learn, it should have simple rules and require minimal equipment. For a sport to be considered hard to master, it should require years of practice to perfect and have a large variety of techniques and strategies...

Step 2: Consider different sports that fit these criteria.

Some sports that could be considered easy to learn but hard to master include:

- Tennis: Simple rules, easy to pick up, but requires years of practice to master all the different shots and strategies.
- Golf: Basic swing mechanics are easy to learn but perfecting them takes a lifetime of practice.
- ...

Step 3: Evaluate each sport based on the criteria.

Tennis and golf both require expensive equipment, which could make them less accessible to beginners. In terms of skill ceiling and time required to master, tennis and golf both have a high skill ceiling and require years of practice to perfect....

Step 4: Make a decision based on the evaluation.

Based on the above criteria and evaluation, I would say that tennis is the sport that is easiest to learn but hardest to master. While it may require expensive equipment, it is still accessible to many people. The basic rules and techniques are easy to learn, but mastering all the different shots and strategies takes years of practice...

© Copyright Microsoft Corporation. All rights reserved.

Ja, enligt bilden används Chain of Thought (CoT) för att få modellen att bryta ner sitt resonemang steg för steg.

Hur fungerar Chain of Thought?

1. Man ber modellen att tänka i flera steg

1. Istället för att bara ge ett direkt svar, får modellen instruktionen att **analysera frågan** och **stegvis bygga upp sitt resonemang**.

2. Den identifierar kriterier

1. I exemplet definierar modellen först vad som menas med "**lätt att lära, svårt att bemästra**".

3. Den jämför olika alternativ

1. Den listar möjliga sporter och **utvärderar dem** baserat på de definierade kriterierna.

4. Den drar en slutsats

1. Efter att ha gått igenom sina egna resonemang kommer den fram till att **tennis** passar bäst för definitionen.

Skickar man in alla steg på en gång?

Ja, oftast skickar man in **hela strukturen som en prompt**. Det kan göras på två sätt:

1. Explicit instruktion i prompten

1. Man ber modellen att **själv resonera stegvis** genom att lägga till något som: Tänk högt och bryt ner ditt svar i steg innan du ger en slutsats.

2. Ge exempel på en tidigare Chain of Thought

1. Man kan ge ett **färdigt exempel** på hur modellen ska tänka, och den lär sig att följa det mönstret.
2. Exempel: Q: Vilken sport är lätt att lära men svår att bemästra? A: Steg 1: Identifiera kriterier för "lätt att lära" och "svårt att bemästra"... Steg 2: Lista möjliga sporter... Steg 3: Utvärdera dem... Steg 4: Ge ett slutligt svar.

Varför är Chain of Thought användbart?

- Bättre och mer logiska svar
- Hjälper AI:n att undvika felaktiga gissningar
- Användbart i komplexa frågor, t.ex. matematik, programmering och beslutsfattande.

Så ja, hela resonemanget skickas in i en prompt, och modellen följer stegen istället för att bara gissa svaret direkt. 🚀

Asking a model to respond with the step by step process by which it determined the response is a helpful way to understand how the model is interpreting the prompt. By doing so, you can see where the model made an incorrect logical turn and better understand how to change your prompt to avoid the error. This technique can include asking it to cite its sources, like Bing chat does (which uses a GPT-4 generation model), and giving reasoning for why it determined its answer.

Exercise: Utilize prompt engineering in your app



Explore prompt engineering techniques

Connect your app to Azure OpenAI and test prompts with increasing prompt engineering

Use the hosted lab environment if provided, or view the lab instructions at the link below:

<https://aka.ms/openai-prompt-engineering-lab>

© Copyright Microsoft Corporation. All rights reserved.

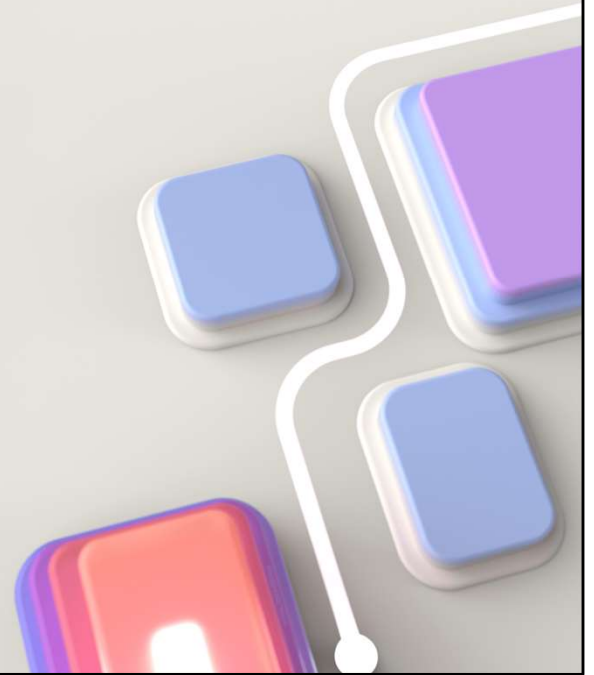
The exercise will take a minimum of 30 minutes to complete.

Not all students work at the same pace, so you should allow additional time as necessary for your class.

IMPORTANT: make sure students know to decrease their model quota, as indicated in the instructions. It is important to make sure all students can deploy a model in the lab environment, and they should never hit issues with models deployed with lower quota.

Also, if students find the model they're told to deploy does not appear available, have them delete the Azure OpenAI resource and recreate it in one of the regions that has the correct model. Model availability can be found here: [Azure OpenAI Service models - Azure OpenAI | Microsoft Learn](#)

Implement Retrieval Augmented Generation (RAG) with Azure OpenAI Service



Before delivering this presentation, review the associated module on Microsoft Learn (<https://aka.ms/mslearn-openai-own-data-module>).

The goal of this presentation is to teach developers how to integrate your own data into responses from Azure OpenAI models. We cover creating and connecting to a data source in the **Azure AI Foundry portal**, and using it in their applications by using the REST APIs and Python and C# SDKs

The estimated time to present these slides is 30 minutes, plus 30 minutes to complete the hands-on exercise.

Learning Objectives

After completing this module, you will be able to:

- 1 Understand how RAG using your own data works
- 2 Use the REST API
- 3 Use language specific SDKs

© Copyright Microsoft Corporation. All rights reserved.

RAG: Retrieval Augmented Generation

Prompt

System: You are an AI travel agent that advises on sightseeing activities in major cities.
User: Give me ideas for things to see and do in London to help me plan a trip there. Include a numbered list of the top 10 places to visit in the city.

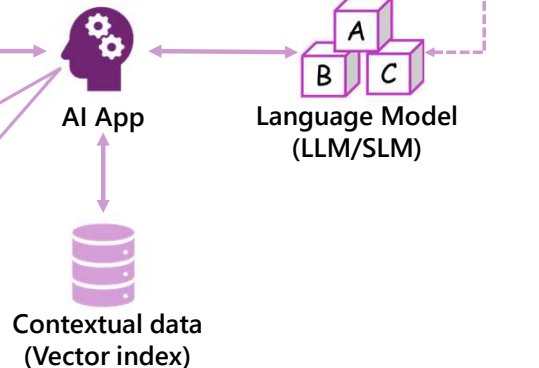
Certainly! Here's are the top 10 activities offered by Margie's Travel:

1. Tour of the Tower of London¹.
2. Open top bus tour of the City of Westminster²
- ...

1. Margie's London brochure
 2. Margie's city tours pamphlet

Completion (response)

© Copyright Microsoft Corporation. All rights reserved.



Den här bilden förklarar **Retrieval-Augmented Generation (RAG)**, en metod där AI kombinerar **generativ textmodellering** med **extern informationshämtning**. Detta förbättrar svarens relevans och korrekthet genom att hämta uppdaterad och domänspecifik data.

Vad visar bilden?

Den illustrerar **hur en AI-app använder RAG** för att svara på en fråga om turistaktiviteter i London.

1. Prompten skickas till AI-appen

1. **Systemmeddelande:** "Du är en AI-reseagent..."
2. **Användaren frågar:** "Ge mig idéer om saker att göra i London..."

2. AI-appen hanterar frågan

1. AI-appen fungerar som en mellanhand och skickar frågan till en **språkmodell (LLM/SLM)**.

3. LLM/SLM genererar ett svar baserat på träning

1. Språkmodellen (LLM = Large Language Model, SLM = Small Language Model) har tränats på **generella data**, vilket ger den bred kunskap men ingen specifik information om t.ex. "Margie's Travel".

4. AI-appen hämtar extern kontextuell data

1. Eftersom språkmodellen **inte alltid har uppdaterad information**, hämtar AI-appen ytterligare **relevant data från en vektorbas** (en form av databas för snabb informationsökning).
2. I detta exempel används **Margie's London brochure och city tour pamphlet** som källor.

5. AI-appen kombinerar extern data och generativ AI för att skapa svaret

1. Svaret blir en kombination av språkmodellens generella kunskap och **specifik information från databasen**.
2. Exempel på svar: "Here's the top 10 activities offered by Margie's Travel:
 1. Tour of the Tower of London¹.
 2. Open top bus tour of Westminster²."

6. Svar med källhänvisningar (Completion)

1. Svarstexten innehåller **footnoter** (¹ och ²) som visar var informationen hämtades ifrån.

2. Det gör att användaren **kan verifiera svaret**, vilket är en stor fördel jämfört med en ren generativ AI som bara skapar text utan att ge källor.

Varför är RAG bättre än en vanlig LLM?

1. Bättre precision och uppdaterad information

1. Vanliga språkmodeller har en **fast "cutoff"** i sin träning, medan RAG kan **hämta uppdaterade data** vid varje fråga.

2. Färre hallucineringar (felaktiga svar)

1. Eftersom AI:n hämtar **verifierbar information**, minskar risken för att den hittar på fakta.

3. Bättre anpassning till specifika domäner

1. Om AI behöver svara på medicinska eller juridiska frågor kan den **hämta specialiserad information**, istället för att enbart använda generell kunskap.

Praktiska användningsområden för RAG

- **Sökmotorer** – AI som söker i interna dokument innan det svarar.
- **Kundsupport** – En chatbot som hämtar uppdaterade produktmanualer.
- **Medicinska AI-assistenter** – Genererar svar baserat på vetenskapliga artiklar.
- **Företagsintern AI** – Letar upp information i företagsdokument för att svara på frågor.

Sammanfattning

RAG är en kraftfull teknik som **kombinerar generativ AI med informationshämtning**. Den hjälper AI att ge **mer exakta och verifierbara svar** genom att dra in kontextuell data från en vektorbas. På så sätt kan AI:n **hålla sig uppdaterad och mer pålitlig!** 🚀

How Azure OpenAI can use your data



Set up your data source

- Use an existing data source, such as an Azure AI Search resource
- Use the Azure AI Foundry portal to create that data source, if you don't already have one
- When creating the data source, you can use data already in your account such as blob storage



Configure the Azure AI Foundry or your app to connect to that data source

- In the Azure AI Foundry portal, set up the connection by pointing it to the data source
- In your app, specify the data source in the prompt parameters
- Both configurations allow the search resource to augment the prompt



Use the Azure OpenAI model, which now uses your data for grounding

- Chat with the AI models like normal
- If the data source has relevant information about the prompt, it will use that data
- You can specify if the AI model is limited to just your data source

© Copyright Microsoft Corporation. All rights reserved.

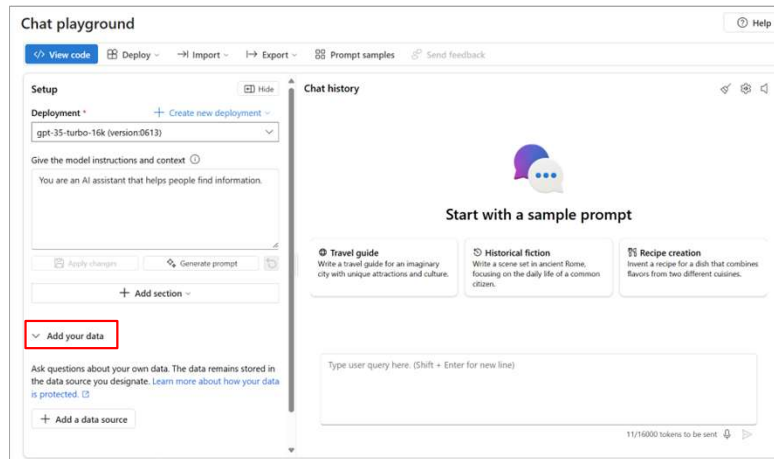
Fine-tuning vs. using your own data

Fine-tuning is a technique used to create a custom model by training an existing foundational model such as gpt-35-turbo with a dataset of additional training data. Fine-tuning can result in higher quality requests that prompt engineering alone, customize the model on examples larger than can fit in a prompt, and allow the user to provide fewer examples to get the same high quality response. However, the process for fine-tuning is both costly and time intensive, and should only be used for use cases where it's necessary.

Azure OpenAI on your data still uses the stateless API to connect to the model, which removes the requirement of training a custom model with your data and simplifies the interaction with the AI model. AI Search first finds the useful information to answer the prompt, and Azure OpenAI forms the response based on that information.

Connect to your data source

- Add your data source in the Chat playground, under **Add your data**
- Use an existing data source, or use that wizard to create a new one
- Once connected, a new chat session will start. Chat like normal, and see how the AI model references that data



© Copyright Microsoft Corporation. All rights reserved.

After this slide is a great time to switch to a demo in your portal and see the way it references the grounding data.

If you're using your own index that wasn't created through the Azure AI Foundry portal, one of the pages allows you to specify your column mapping. It's important to provide accurate fields, to enable the model to provide a better response, especially for **Content data**.

If you're uploading or using files already in a storage account, Azure OpenAI on your data supports .md, .txt, .html, .pdf, and Microsoft Word or PowerPoint files. If any of these files contain graphics or images, the response quality depends on how well text can be extracted from the visual content.

When uploading data or connecting to files in a storage account, it's recommended to use the Azure AI Foundry portal to create the search resource and index. Adding data this way allows the appropriate chunking to happen when inserting into the index, yielding better responses. If you're using large text files or forms, you should use the available data preparation script to improve the AI model's accuracy.

<https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/use-your-data#ingesting-your-data-into-azure-cognitive-search>

Enabling semantic search for your AI Search service can improve the result of searching your data index and you're likely to receive higher quality responses and citations. However, enabling semantic search may increase the cost of the search service.

Using the Azure OpenAI REST API

Using your own data

`https://endpoint.openai.azure.com/openai/deployments/deployment/chat/completions?api-version=version`

- With each call, you need to specify the data source values, along with the messages array and any other parameters
- Authentication in the data source definition is for your search resource, not your Azure OpenAI resource

```
{
  "data_sources": [
    {
      "type": "azure_search",
      "parameters": {
        "endpoint": "<your_search_endpoint>",
        "index_name": "<your_search_index>",
        "authentication": {
          "type": "system_assigned_managed_identity"
        }
      }
    }
  ],
  "messages": [
    ...
  ]
}
```

© Copyright Microsoft Corporation. All rights reserved.

The slide shows an example endpoint and input body for using your own data.

The API version is limited to what works, so advise checking the docs.

Starting in version **2024-02-15-preview**, breaking changes were introduced. Notes about that here: [Azure OpenAI On Your Data Python & REST API reference - Azure OpenAI | Microsoft Learn](#). Of note, the REST endpoint changed, as well as some parameter naming. The newest version changes are reflected here.

Using the Azure OpenAI SDKs

Language specific SDKs are available for use in your applications, in both C# and Python.

Code structure follows a similar pattern for both languages.

Current supported data sources are:

- Azure AI Search
- Azure Cosmos DB for MangoDB vCore
- Plus others in preview, soon to be released GA

Pseudo code structure:

```
<include library>

<create client>

<define chat messages and options>

<define data source object to include with request>

<send request>

<extract response content>
```

© Copyright Microsoft Corporation. All rights reserved.

This pseudocode shows the general structure of using the SDK. This will vary slightly with SDK versions.

Highly recommend trainers show actual code samples in class, but they are no longer included here due to frequent changes.

Kodexempel, sidan 1

```
import openai

# Skapa en klient med API-nyckeln
client = openai.AzureOpenAI(
    api_key="din-api-nyckel",
    azure_endpoint="https://ditt-endpoint.openai.azure.com/",
    api_version="2023-05-15"
)

# Definiera chat-meddelanden och inställningar
messages = [
    {"role": "system", "content": "You are a helpful assistant with access to AI Search."},
    {"role": "user", "content": "Find information about Azure OpenAI services."}
]
```

© Copyright Microsoft Corporation. All rights reserved.

This pseudocode shows the general structure of using the SDK. This will vary slightly with SDK versions.

Highly recommend trainers show actual code samples in class, but they are no longer included here due to frequent changes.

Kodexempel, sidan 2

```
# Definiera en dataförankring (grounding) med Azure AI Search
data_source = {
    "type": "azure_search",
    "parameters": {
        "index_name": "documents-index",
        "query": "Azure OpenAI services",
        "top_k": 5
    }
}

# Skicka förfrågan med dataförankring
response = client.chat.completions.create(
    model="gpt-35-turbo",
    messages=messages,
    max_tokens=100,
    temperature=0.7,
    extra_body={"data_sources": [data_source]} # Kopplar AI Search till GPT
)

# Skriv ut svaret
print(response.choices[0].message.content)
```

© Copyright Microsoft Corporation. All rights reserved.

This pseudocode shows the general structure of using the SDK. This will vary slightly with SDK versions.

Highly recommend trainers show actual code samples in class, but they are no longer included here due to frequent changes.

Exercise: Implement Retrieval Augmented Generation (RAG) with Azure OpenAI Service



Set up and connect your data in the chat playground

Configure your app to use your own data for augmenting the prompt

Use the hosted lab environment if provided, or view the lab instructions at the link below:

<https://aka.ms/mslearn-openai-own-data>

© Copyright Microsoft Corporation. All rights reserved.

The exercise will take a minimum of 30 minutes to complete.

Not all students work at the same pace, so you should allow additional time as necessary for your class.

IMPORTANT: make sure students know to decrease their model quota, as indicated in the instructions. It is important to make sure all students can deploy a model in the lab environment, and they should never hit issues with models deployed with lower quota.

Also, if students find the model they're told to deploy does not appear available, have them delete the Azure OpenAI resource and recreate it in one of the regions that has the correct model. Model availability can be found here: [Azure OpenAI Service models - Azure OpenAI | Microsoft Learn](#)

Extended interactive exercises



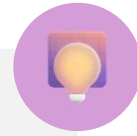
Generate code

Generate images

<https://aka.ms/develop-azure-openai>

© Copyright Microsoft Corporation. All rights reserved.

Knowledge check



- 1 What is the purpose of providing conversation history to an AI model?**
 - ☐ Providing conversation history to an AI model is irrelevant and has no effect on the AI's performance.
 - ☐ To limit the number of input tokens used by the model
 - ☒ To enable the model to continue responding in a similar way and allow the user to reference previous content in subsequent queries
- 2 Which parameter could you adjust to change the randomness or creativeness of completions?**
 - ☒ Temperature
 - ☐ Frequency penalty
 - ☐ Stop sequence
- 3 You plan to implement a multi-turn conversation with Azure OpenAI. Which endpoint should you use?**
 - ☐ Completions
 - ☒ Chat
 - ☐ Embeddings

© Copyright Microsoft Corporation. All rights reserved.

1 c
2 a
3 b

Här är en förklaring av de olika svarsalternativen i bilden:

Fråga 1: Vad är syftet med att ge en AI-modell konversationshistorik?


- ☐ Providing conversation history to an AI model is irrelevant and has no effect on the AI's performance.
 - ☒ *Fel.* AI-modeller som GPT är *kontextberoende*, vilket innebär att de behöver konversationshistorik för att ge sammanhängande svar.
- ☐ To limit the number of input tokens used by the model.
 - ☒ *Fel.* Historiken ökar snarare antalet tokens, vilket kan påverka kostnaden och prestandan, men det är inte det primära syftet.
- ☒ To enable the model to continue responding in a similar way and allow the user to reference previous content in subsequent queries.
 - ☒ *Rätt svar.* Konversationshistorik gör att AI:n kan hålla kontext mellan interaktioner, så att den "kommer ihåg" tidigare meddelanden och ger mer relevanta svar.

✚ **Sammanfattning:** Att ge AI:n konversationshistorik gör den bättre på att hålla en naturlig dialog och förstå referenser till tidigare meddelanden.

Fråga 2: Vilken parameter påverkar slumpmässigheten eller kreativiteten i svaren?

- ☒ Temperature
 - ☒ *Rätt svar.* *Temperature* styr slumpmässigheten i AI:s svar:
 - **Låg temperatur (t.ex. 0.1):** Mer förutsägbara och konservativa svar.
 - **Hög temperatur (t.ex. 1.0):** Mer kreativa och varierade svar.
- ☐ Frequency penalty
 - ☒ *Fel.* *Frequency penalty* används för att minska upprepningar i AI:s svar, men den påverkar inte kreativiteten direkt.

- ☐ Stop sequence

-  *Fel.* Stop sequence används för att definiera en textsekvens där AI:n ska sluta generera svar, inte för att styra slumpmässigheten.

 **Sammanfattning:** *Temperature* är den parameter som styr hur kreativa eller slumpmässiga svaren blir – högre värde ger mer varierade svar.

Fråga 3: Vilken endpoint ska du använda för en konversation med flera turer?

- ☐ Completions

-  *Fel.* Completions-endpointen används främst för att generera enstaka svar från en given prompt, inte för att hålla en längre konversation.

- ☒ Chat

-  *Rätt svar.* Chat-endpointen (t.ex. chat.completions.create) är speciellt byggd för att hantera flerledade samtal där AI:n kan hålla kontext och svara baserat på tidigare meddelanden.

- ☐ Embeddings

-  *Fel.* Embeddings används för att representera text som numeriska vektorer för semantisk sökning och likhetsanalys, inte för att föra konversationer.

 **Sammanfattning:** För att hantera en **flerturssamtal** med Azure OpenAI ska du använda **Chat-endpointen**, eftersom den stödjer konversationshistorik.

Övergripande slutsats:

- AI behöver **konversationshistorik** för att hålla en sammanhängande dialog.
- **Temperature** justerar slumpmässigheten i AI:s svar.
- **Chat-endpointen** är det bästa valet för flerturssamtal.

 **Sammanfattning i en mening:** För en naturlig och kontextmedveten chattupplevelse, använd **Chat-endpointen** och justera *Temperature* för att styra kreativiteten!

Learning Recap

In this learning path, we:

- Created and deployed Azure OpenAI resources
- Integrated Azure OpenAI into your application through REST APIs and SDKs
- Explored prompt engineering techniques to improve model responses
- Connected your own data for grounding an Azure OpenAI model

© Copyright Microsoft Corporation. All rights reserved.



© Copyright Microsoft Corporation. All rights reserved.