Azure OpenAI speech to speech chat (Python)

Prerequisites

- Azure subscription - Create one for free
- Create a Microsoft Azure OpenAI Service resource in the Azure portal.
- Deploy a model in your Azure OpenAI resource. For more information about model deployment, see the Azure OpenAI resource deployment guide.
- Get the Azure OpenAI resource key and endpoint. After your Azure OpenAI resource is deployed, select **Go to resource** to view and manage keys. For more information about Azure AI services resources, see Get the keys for your resource.
- Create a Speech resource in the Azure portal.
- Get the Speech resource key and region. After your Speech resource is deployed, select **Go to resource** to view and manage keys. For more information about Azure AI services resources, see Get the keys for your resource.

Set up the environment

The Speech SDK for Python is available as a Python Package Index (PyPI) module. The Speech SDK for Python is compatible with Windows, Linux, and macOS.

- Install the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017, 2019, and 2022 for your platform. Installing this package for the first time might require a restart.
- On Linux, you must use the x64 target architecture.

Install a version of Python from 3.7 or later. First check the SDK installation guide for any more requirements.

Install the following Python libraries: os, requests, json.

Set environment variables

This example requires environment variables named OPEN_AI_KEY, OPEN_AI_ENDPOINT, OPEN_AI_DEPLOYMENT_NAME, SPEECH_KEY, and SPEECH_REGION.

Your application must be authenticated to access Azure AI services resources. For production, use a secure way of storing and accessing your credentials. For example, after you get a key for your Speech resource, write it to a new environment variable on the local machine running the application.

Tip

Don't include the key directly in your code, and never post it publicly. See **Azure AI services security** for more authentication options like **Azure Key Vault**.

To set the environment variables, open a console window, and follow the instructions for your operating system and development environment.

- To set the OPEN_AI_KEY environment variable, replace your-openai-key with one of the keys for your resource.
- To set the OPEN_AI_ENDPOINT environment variable, replace your-openai-endpoint with one of the regions for your resource.
- To set the OPEN_AI_DEPLOYMENT_NAME environment variable, replace your-openai-deployment-name with one of the regions for your resource.
- To set the SPEECH_KEY environment variable, replace your-speech-key with one of the keys for your resource.
- To set the SPEECH_REGION environment variable, replace your-speech-region with one of the regions for your resource.

Windows

Console

setx OPEN_AI_KEY your-openai-key

setx OPEN_AI_ENDPOINT your-openai-endpoint

setx OPEN_AI_DEPLOYMENT_NAME your-openai-deployment-name

setx SPEECH_KEY your-speech-key

setx SPEECH_REGION your-speech-region

Note

If you only need to access the environment variable in the current running console, set the environment variable with set instead of setx.

After you add the environment variables, you might need to restart any running programs that need to read the environment variable, including the console window. For example, if Visual Studio is your editor, restart Visual Studio before running the example.

Recognize speech from a microphone

Follow these steps to create a new console application.

1. Open a command prompt window in the folder where you want the new project. Open a command prompt where you want the new project, and create a new file named openai-speech.py.

2. Run this command to install the Speech SDK:

Console

pip install azure-cognitiveservices-speech

3. Run this command to install the OpenAI SDK:

Console

pip install openai

4. Create a file named *openai-speech.py*. Copy the following code into that file:

Python

```
import os
import azure.cognitiveservices.speech as speechsdk
from openai import AzureOpenAI

# This example requires environment variables named "OPEN_AI_KEY",
"OPEN_AI_ENDPOINT" and "OPEN_AI_DEPLOYMENT_NAME"
# Your endpoint should look like the following
https://YOUR_OPEN_AI_RESOURCE_NAME.openai.azure.com/
client = AzureOpenAI(
azure_endpoint=os.environ.get('OPEN_AI_ENDPOINT'),
api_key=os.environ.get('OPEN_AI_KEY'),
api_version="2023-05-15"
)

# This will correspond to the custom name you chose for your deployment when
you deployed a model.
deployment_id=os.environ.get('OPEN_AI_DEPLOYMENT_NAME')

# This example requires environment variables named "SPEECH_KEY" and
"SPEECH_REGION"
speech_config =
speechsdk.SpeechConfig(subscription=os.environ.get('SPEECH_KEY'),
region=os.environ.get('SPEECH_REGION'))
audio_output_config =
speechsdk.audio.AudioOutputConfig(use_default_speaker=True)
audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)

# Should be the locale for the speaker's language.
speech_config.speech_recognition_language="en-US"
speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config,
audio_config=audio_config)
```

```
# The language of the voice that responds on behalf of Azure OpenAI.
speech_config.speech_synthesis_voice_name='en-US-JennyMultilingualNeural'
speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config,
audio_config=audio_output_config)
# tts sentence end mark
tts_sentence_end = [ ".", "!", "?", ";", "。", "！", "？", "；", "\n" ]

# Prompts Azure OpenAI with a request and synthesizes the response.
def ask_openai(prompt):
    # Ask Azure OpenAI in streaming way
    response = client.chat.completions.create(model=deployment_id,
max_tokens=200, stream=True, messages=[
        {"role": "user", "content": prompt}
    ])
    collected_messages = []
    last_tts_request = None

    # iterate through the stream response stream
    for chunk in response:
        if len(chunk.choices) > 0:
            chunk_message = chunk.choices[0].delta.content  # extract the
message
            if chunk_message is not None:
                collected_messages.append(chunk_message)  # save the message
                if chunk_message in tts_sentence_end: # sentence end found
                    text = ''.join(collected_messages).strip() # join the
recieved message together to build a sentence
                    if text != '': # if sentence only have \n or space, we
could skip
                        print(f"Speech synthesized to speaker for: {text}")
                        last_tts_request =
speech_synthesizer.speak_text_async(text)
                        collected_messages.clear()
    if last_tts_request:
        last_tts_request.get()

# Continuously listens for speech input to recognize and send as text to Azure
OpenAI
def chat_with_open_ai():
    while True:
        print("Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end
the conversation.")
        try:
            # Get audio from the microphone and then send it to the TTS
service.
            speech_recognition_result =
speech_recognizer.recognize_once_async().get()

            # If speech is recognized, send it to Azure OpenAI and listen for
the response.
            if speech_recognition_result.reason ==
speechsdk.ResultReason.RecognizedSpeech:
                if speech_recognition_result.text == "Stop.":
                    print("Conversation ended.")
                    break
```

```
                print("Recognized speech:
{}".format(speech_recognition_result.text))
                ask_openai(speech_recognition_result.text)
            elif speech_recognition_result.reason ==
speechsdk.ResultReason.NoMatch:
                print("No speech could be recognized:
{}".format(speech_recognition_result.no_match_details))
                break
            elif speech_recognition_result.reason ==
speechsdk.ResultReason.Canceled:
                cancellation_details =
speech_recognition_result.cancellation_details
                print("Speech Recognition canceled:
{}".format(cancellation_details.reason))
                if cancellation_details.reason ==
speechsdk.CancellationReason.Error:
                    print("Error details:
{}".format(cancellation_details.error_details))
        except EOFError:
            break

# Main

try:
    chat_with_open_ai()
except Exception as err:
    print("Encountered exception. {}".format(err))
```

5. To increase or decrease the number of tokens returned by Azure OpenAI, change the MaxTokens property in the ChatCompletionsOptions class instance.
6. Run your new console application to start speech recognition from a microphone:

Console

python openai-speech.py

Important

Make sure that you set
the OPEN_AI_KEY, OPEN_AI_ENDPOINT, OPEN_AI_DEPLOYMENT_NAME, SPEECH_KEY and SPEECH_REGION **environment variables** as described. If you don't set these variables, the sample will fail with an error message.

Speak into your microphone when prompted. The console output includes the prompt for you to begin speaking, then your request as text, and then the response from Azure OpenAI as text. The response from Azure OpenAI should be converted from text to speech and then output to the default speaker.

Remarks

Here are some more considerations:

- To change the speech recognition language, replace en-US with another supported language. For example, es-ES for Spanish (Spain). The default language is en-US.
- To change the voice that you hear, replace en-US-JennyMultilingualNeural with another supported voice. If the voice doesn't speak the language of the text returned from Azure OpenAI, the Speech service doesn't output synthesized audio.
- To use a different model, replace gpt-35-turbo-instruct with the ID of another deployment. The deployment ID isn't necessarily the same as the model name. You named your deployment when you created it in Azure OpenAI Studio.
- Azure OpenAI also performs content moderation on the prompt inputs and generated outputs. The prompts or responses might be filtered if harmful content is detected.