Translate Speech

Azure AI Speech includes a speech translation API that you can use to translate spoken language. For example, suppose you want to develop a translator application that people can use when traveling in places where they don't speak the local language. They would be able to say phrases such as "Where is the station?" or "I need to find a pharmacy" in their own language, and have it translate them to the local language.

NOTE This exercise requires that you are using a computer with speakers/headphones. For the best experience, a microphone is also required. Some hosted virtual environments may be able to capture audio from your local microphone, but if this doesn't work (or you don't have a microphone at all), you can use a provided audio file for speech input. Follow the instructions carefully, as you'll need to choose different options depending on whether you are using a microphone or the audio file.

Provision an *Azure AI Speech* resource

If you don't already have one in your subscription, you'll need to provision an **Azure AI Speech** resource.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. In the search field at the top, search for **Azure AI services** and press **Enter**, then select **Create** under **Speech service** in the results.
3. Create a resource with the following settings:
    - **Subscription**: *Your Azure subscription*
    - **Resource group**: *Choose or create a resource group*
    - **Region**: *Choose any available region*
    - **Name**: *Enter a unique name*
    - **Pricing tier**: Select **F0** (*free*), or **S** (*standard*) if F is not available.
    - **Responsible AI Notice**: Agree.
4. Select **Review + create**.
5. Wait for deployment to complete, and then go to the deployed resource.
6. View the **Keys and Endpoint** page. You will need the information on this page later in the exercise.

Prepare to develop an app in Visual Studio Code

You'll develop your speech app using Visual Studio Code. The code files for your app have been provided in a GitHub repo.

Tip: If you have already cloned the **mslearn-ai-language** repo, open it in Visual Studio code. Otherwise, follow these steps to clone it to your development environment.

1. Start Visual Studio Code.

2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/mslearn-ai-language` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo.
   Note: If you are prompted to add required assets to build and debug, select **Not Now**.

Configure your application

Applications for both C# and Python have been provided. Both apps feature the same functionality. First, you'll complete some key parts of the application to enable it to use your Azure AI Speech resource.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Labfiles/08-speech-translation** folder and expand the **CSharp** or **Python** folder depending on your language preference and the **translator** folder it contains. Each folder contains the language-specific code files for an app into which you're you're going to integrate Azure AI Speech functionality.
2. Right-click the **translator** folder containing your code files and open an integrated terminal. Then install the Azure AI Speech SDK package by running the appropriate command for your language preference:

   C#

   CodeCopy

   ```
   dotnet add package Microsoft.CognitiveServices.Speech --version 1.30.0
   ```

   Python

   CodeCopy

   ```
   pip install azure-cognitiveservices-speech==1.30.0
   ```

3. In the **Explorer** pane, in the **translator** folder, open the configuration file for your preferred language
   - **C#**: appsettings.json
   - **Python**: .env
4. Update the configuration values to include the **region** and a **key** from the Azure AI Speech resource you created (available on the **Keys and Endpoint** page for your Azure AI Speech resource in the Azure portal).

NOTE: Be sure to add the *region* for your resource, not the endpoint!

5. Save the configuration file.

Add code to use the Speech SDK

1. Note that the **translator** folder contains a code file for the client application:
   - **C#**: Program.cs
   - **Python**: translator.py

   Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Azure AI Speech SDK:

   C#: Program.cs

   C#Copy

```
// Import namespaces
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using Microsoft.CognitiveServices.Speech.Translation;
```

   Python: translator.py

   CodeCopy

```
# Import namespaces
import azure.cognitiveservices.speech as speech_sdk
```

2. In the **Main** function, note that code to load the Azure AI Speech service key and region from the configuration file has already been provided. You must use these variables to create a **SpeechTranslationConfig** for your Azure AI Speech resource, which you will use to translate spoken input. Add the following code under the comment **Configure translation**:

   C#: Program.cs

   C#Copy

```
// Configure translation
```

```
 translationConfig = SpeechTranslationConfig.FromSubscription(aiSvcKey,
aiSvcRegion);
 translationConfig.SpeechRecognitionLanguage = "en-US";
 translationConfig.AddTargetLanguage("fr");
 translationConfig.AddTargetLanguage("es");
 translationConfig.AddTargetLanguage("hi");
 Console.WriteLine("Ready to translate from " +
translationConfig.SpeechRecognitionLanguage);
```

Python: translator.py

CodeCopy

```
 # Configure translation
 translation_config =
speech_sdk.translation.SpeechTranslationConfig(ai_key, ai_region)
 translation_config.speech_recognition_language = 'en-US'
 translation_config.add_target_language('fr')
 translation_config.add_target_language('es')
 translation_config.add_target_language('hi')
 print('Ready to translate
from',translation_config.speech_recognition_language)
```

3. You will use the **SpeechTranslationConfig** to translate speech into text, but you will also use a **SpeechConfig** to synthesize translations into speech. Add the following code under the comment **Configure speech**:

   C#: Program.cs

   C#Copy

   ```
   // Configure speech
   speechConfig = SpeechConfig.FromSubscription(aiSvcKey, aiSvcRegion);
   ```

   Python: translator.py

   CodeCopy

   ```
   # Configure speech
   speech_config = speech_sdk.SpeechConfig(ai_key, ai_region)
   ```

4. Save your changes and return to the integrated terminal for the **translator** folder, and enter the following command to run the program:

   C#

CodeCopy

```
dotnet run
```

Python

CodeCopy

```
python translator.py
```

5. If you are using C#, you can ignore any warnings about using the **await** operator in asynchronous methods - we'll fix that later. The code should display a message that it is ready to translate from en-US and prompt you for a target language. Press ENTER to end the program.

Implement speech translation

Now that you have a **SpeechTranslationConfig** for the Azure AI Speech service, you can use the Azure AI Speech translation API to recognize and translate speech.

IMPORTANT: This section includes instructions for two alternative procedures. Follow the first procedure if you have a working microphone. Follow the second procedure if you want to simulate spoken input by using an audio file.

If you have a working microphone

1. In the **Main** function for your program, note that the code uses the **Translate** function to translate spoken input.
2. In the **Translate** function, under the comment **Translate speech**, add the following code to create a **TranslationRecognizer** client that can be used to recognize and translate speech using the default system microphone for input.

C#: Program.cs

C#Copy

```
// Translate speech
using AudioConfig audioConfig = AudioConfig.FromDefaultMicrophoneInput();
using TranslationRecognizer translator = new
TranslationRecognizer(translationConfig, audioConfig);
Console.WriteLine("Speak now...");
```

```
 TranslationRecognitionResult result = await
translator.RecognizeOnceAsync();
 Console.WriteLine($"Translating '{result.Text}'");
 translation = result.Translations[targetLanguage];
 Console.OutputEncoding = Encoding.UTF8;
 Console.WriteLine(translation);
```

Python: translator.py

CodeCopy

```
 # Translate speech
 audio_config = speech_sdk.AudioConfig(use_default_microphone=True)
 translator =
speech_sdk.translation.TranslationRecognizer(translation_config,
audio_config = audio_config)
 print("Speak now...")
 result = translator.recognize_once_async().get()
 print('Translating "{}"'.format(result.text))
 translation = result.translations[targetLanguage]
 print(translation)
```

NOTE The code in your application translates the input to all three languages in a single call. Only the translation for the specific language is displayed, but you could retrieve any of the translations by specifying the target language code in the **translations** collection of the result.

3. Now skip ahead to the **Run the program** section below.

Alternatively, use audio input from a file

1. In the terminal window, enter the following command to install a library that you can use to play the audio file:

   C#: Program.cs

   C#Copy

   ```
   dotnet add package System.Windows.Extensions --version 4.6.0
   ```

   Python: translator.py

   CodeCopy

```
pip install playsound==1.3.0
```

2. In the code file for your program, under the existing namespace imports, add the following code to import the library you just installed:

   C#: Program.cs

   C#Copy

```
using System.Media;
```

   Python: translator.py

   CodeCopy

```
from playsound import playsound
```

3. In the **Main** function for your program, note that the code uses the **Translate** function to translate spoken input. Then in the **Translate** function, under the comment **Translate speech**, add the following code to create a **TranslationRecognizer** client that can be used to recognize and translate speech from a file.

   C#: Program.cs

   C#Copy

```
// Translate speech
string audioFile = "station.wav";
SoundPlayer wavPlayer = new SoundPlayer(audioFile);
wavPlayer.Play();
using AudioConfig audioConfig = AudioConfig.FromWavFileInput(audioFile);
using TranslationRecognizer translator = new
TranslationRecognizer(translationConfig, audioConfig);
Console.WriteLine("Getting speech from file...");
TranslationRecognitionResult result = await
translator.RecognizeOnceAsync();
Console.WriteLine($"Translating '{result.Text}'");
translation = result.Translations[targetLanguage];
Console.OutputEncoding = Encoding.UTF8;
Console.WriteLine(translation);
```

   Python: translator.py

CodeCopy

```
 # Translate speech
 audioFile = 'station.wav'
 playsound(audioFile)
 audio_config = speech_sdk.AudioConfig(filename=audioFile)
 translator =
speech_sdk.translation.TranslationRecognizer(translation_config,
audio_config = audio_config)
 print("Getting speech from file...")
 result = translator.recognize_once_async().get()
 print('Translating "{}"'.format(result.text))
 translation = result.translations[targetLanguage]
 print(translation)
```

Run the program

1. Save your changes and return to the integrated terminal for the **translator** folder, and enter the following command to run the program:

   C#

   CodeCopy

   ```
   dotnet run
   ```

   Python

   CodeCopy

   ```
   python translator.py
   ```

2. When prompted, enter a valid language code (*fr*, *es*, or *hi*), and then, if using a microphone, speak clearly and say "where is the station?" or some other phrase you might use when traveling abroad. The program should transcribe your spoken input and translate it to the language you specified (French, Spanish, or Hindi). Repeat this process, trying each language supported by the application. When you're finished, press ENTER to end the program.

   The TranslationRecognizer gives you around 5 seconds to speak. If it detects no spoken input, it produces a "No match" result. The translation to Hindi may

not always be displayed correctly in the Console window due to character encoding issues.

NOTE: The code in your application translates the input to all three languages in a single call. Only the translation for the specific language is displayed, but you could retrieve any of the translations by specifying the target language code in the **translations** collection of the result.

Synthesize the translation to speech

So far, your application translates spoken input to text; which might be sufficient if you need to ask someone for help while traveling. However, it would be better to have the translation spoken aloud in a suitable voice.

1. In the **Translate** function, under the comment **Synthesize translation**, add the following code to use a **SpeechSynthesizer** client to synthesize the translation as speech through the default speaker:

   C#: Program.cs

   C#Copy

```
// Synthesize translation
var voices = new Dictionary<string, string>
                {
                    ["fr"] = "fr-FR-HenriNeural",
                    ["es"] = "es-ES-ElviraNeural",
                    ["hi"] = "hi-IN-MadhurNeural"
                };
speechConfig.SpeechSynthesisVoiceName = voices[targetLanguage];
using SpeechSynthesizer speechSynthesizer = new
SpeechSynthesizer(speechConfig);
SpeechSynthesisResult speak = await
speechSynthesizer.SpeakTextAsync(translation);
if (speak.Reason != ResultReason.SynthesizingAudioCompleted)
{
    Console.WriteLine(speak.Reason);
}
```

   Python: translator.py

   CodeCopy

```
# Synthesize translation
voices = {
        "fr": "fr-FR-HenriNeural",
        "es": "es-ES-ElviraNeural",
        "hi": "hi-IN-MadhurNeural"
}
```

```
speech_config.speech_synthesis_voice_name = voices.get(targetLanguage)
speech_synthesizer = speech_sdk.SpeechSynthesizer(speech_config)
speak = speech_synthesizer.speak_text_async(translation).get()
if speak.reason != speech_sdk.ResultReason.SynthesizingAudioCompleted:
    print(speak.reason)
```

2. Save your changes and return to the integrated terminal for the **translator** folder, and enter the following command to run the program:

   C#

   CodeCopy

   ```
   dotnet run
   ```

   Python

   CodeCopy

   ```
   python translator.py
   ```

3. When prompted, enter a valid language code (*fr*, *es*, or *hi*), and then speak clearly into the microphone and say a phrase you might use when traveling abroad. The program should transcribe your spoken input and respond with a spoken translation. Repeat this process, trying each language supported by the application. When you're finished, press **ENTER** to end the program.

NOTE *In this example, you've used a* ***SpeechTranslationConfig*** *to translate speech to text, and then used a* ***SpeechConfig*** *to synthesize the translation as speech. You can in fact use the* ***SpeechTranslationConfig*** *to synthesize the translation directly, but this only works when translating to a single language, and results in an audio stream that is typically saved as a file rather than sent directly to a speaker.*