Translate Text

The **Translator** service is a cognitive service that enables you to translate text between languages.

For example, suppose a travel agency wants to examine hotel reviews that have been submitted to the company's web site, standardizing on English as the language that is used for analysis. By using the Translator service, they can determine the language each review is written in, and if it is not already English, translate it from whatever source language it was written in into English.

Clone the repository for this course

If you have already cloned **AI-102-AIEngineer** code repository to the environment where you're working on this lab, open it in Visual Studio Code; otherwise, follow these steps to clone it now.

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/AI-102-AIEngineer` 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**.

Provision a Cognitive Services resource

If you don't already have one in your subscription, you'll need to provision a **Cognitive Services** resource.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. Select the **+ Create a resource** button, search for *cognitive services*, and create a **Cognitive Services** resource with the following settings:

   - **Subscription**: *Your Azure subscription*
   - **Resource group**: *Choose or create a resource group (if you are using a restricted subscription, you may not have permission to create a new resource group - use the one provided)*
   - **Region**: *Choose any available region*
   - **Name**: *Enter a unique name*
   - **Pricing tier**: Standard S0
3. Select the required checkboxes and create the resource.
4. Wait for deployment to complete, and then view the deployment details.
5. When the resource has been deployed, go to it and view its **Keys and Endpoint** page. You will need one of the keys and the location in which the service is provisioned from this page in the next procedure.

Prepare to use the Translator service

In this exercise, you'll complete a partially implemented client application that uses the Translator REST API to translate hotel reviews.

> **Note**: You can choose to use the API from either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **06-translate-text** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.
2. View the contents of the **text-translation** folder, and note that it contains a file for configuration settings:

   - **C#**: appsettings.json

- **Python**: .env

Open the configuration file and update the configuration values it contains to include an authentication **key** for your cognitive services resource, and the **location** where it is deployed (not the endpoint) - you should copy both of these from the **keys and Endpoint** page for your cognitive services resource. Save your changes.

3. Note that the **text-translation** folder contains a code file for the client application:

   - **C#**: Program.cs
   - **Python**: text-translation.py

   Open the code file and examine the code it contains.
4. In the **Main** function, note that code to load the cognitive services key and region from the configuration file has already been provided. The endpoint for the translation service is also specified in your code.
5. Right-click the **text-translation** folder, open an integrated terminal, and enter the following command to run the program:

   C#

   Code Copy

   ```
   dotnet run
   ```

   Python

   Code Copy

   ```
   python text-translation.py
   ```
6. Observe the output as the code should run without error, displaying the contents of each review text file in the **reviews** folder. The application currently doesn't make use of the Translator service. We'll fix that in the next procedure.

Detect language

The Translator service can automatically detect the source language of text to be translated, but it also enables you to explicitly detect the language in which text is written.

1. In your code file, find the **GetLanguage** function, which currently returns "en" for all text values.
2. In the **GetLanguage** function, under the comment **Use the Translator detect function**, add the following code to use the Translator's REST API to detect the language of the specified text, being careful not to replace the code at the end of the function that returns the language:

C#

Code Copy

```
// Use the Translator detect function
object[] body = new object[] { new { Text = text } };
var requestBody = JsonConvert.SerializeObject(body);
using (var client = new HttpClient())
{
    using (var request = new HttpRequestMessage())
    {
        // Build the request
        string path = "/detect?api-version=3.0";
        request.Method = HttpMethod.Post;
        request.RequestUri = new Uri(translatorEndpoint + path);
        request.Content = new StringContent(requestBody, Encoding.UTF8, "application/json");
        request.Headers.Add("Ocp-Apim-Subscription-Key", cogSvcKey);
        request.Headers.Add("Ocp-Apim-Subscription-Region", cogSvcRegion);

        // Send the request and get response
        HttpResponseMessage response = await client.SendAsync(request).ConfigureAwait(false);
        // Read response as a string
        string responseContent = await response.Content.ReadAsStringAsync();

        // Parse JSON array and get language
        JArray jsonResponse = JArray.Parse(responseContent);
        language = (string)jsonResponse[0]["language"];
    }
}
```

Python

Code

Copy

```
# Use the Translator detect function
path = '/detect'
url = translator_endpoint + path

# Build the request
params = {
    'api-version': '3.0'
}

headers = {
'Ocp-Apim-Subscription-Key': cog_key,
'Ocp-Apim-Subscription-Region': cog_region,
'Content-type': 'application/json'
}

body = [{
    'text': text
}]

# Send the request and get response
request = requests.post(url, params=params, headers=headers, json=body)
response = request.json()

# Parse JSON array and get language
language = response[0]["language"]
```

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

   C#

   Code

   ```
   dotnet run
   ```

   Python

   Code

   ```
   python text-translation.py
   ```

2. Observe the output, noting that this time the language for each review is identified.

Translate text

Now that your application can determine the language in which reviews are written, you can use the Translator service to translate any non-English reviews into English.

1. In your code file, find the **Translate** function, which currently returns and empty string for all text values.
2. In the **Translate** function, under the comment **Use the Translator translate function**, add the following code to use the Translator’s REST API to translate the specified text from its source language into English, being careful not to replace the code at the end of the function that returns the translation:

C#

Code

```
// Use the Translator translate function
object[] body = new object[] { new { Text = text } };
var requestBody = JsonConvert.SerializeObject(body);
using (var client = new HttpClient())
{
    using (var request = new HttpRequestMessage())
    {
        // Build the request
        string path = "/translate?api-version=3.0&from=" + sourceLanguage + "&to=en" ;
        request.Method = HttpMethod.Post;
        request.RequestUri = new Uri(translatorEndpoint + path);
        request.Content = new StringContent(requestBody, Encoding.UTF8, "application/json");
        request.Headers.Add("Ocp-Apim-Subscription-Key", cogSvcKey);
        request.Headers.Add("Ocp-Apim-Subscription-Region", cogSvcRegion);

        // Send the request and get response
        HttpResponseMessage response = await client.SendAsync(request).ConfigureAwait(false);
        // Read response as a string
        string responseContent = await response.Content.ReadAsStringAsync();

        // Parse JSON array and get translation
        JArray jsonResponse = JArray.Parse(responseContent);
        translation = (string)jsonResponse[0]["translations"][0]["text"];
    }
}
```

Python

Code

```
# Use the Translator translate function
path = '/translate'
url = translator_endpoint + path

# Build the request
params = {
    'api-version': '3.0',
    'from': source_language,
    'to': ['en']
}

headers = {
    'Ocp-Apim-Subscription-Key': cog_key,
    'Ocp-Apim-Subscription-Region': cog_region,
    'Content-type': 'application/json'
}

body = [{
    'text': text
}]

# Send the request and get response
request = requests.post(url, params=params, headers=headers, json=body)
response = request.json()

# Parse JSON array and get translation
translation = response[0]["translations"][0]["text"]
```

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

   C#

   Code

   ```
   dotnet run
   ```

   Python

   Code

   ```
   python text-translation.py
   ```

2. Observe the output, noting that non-English reviews are translated into English.

More information

For more information about using the **Translator** service, see the Translator documentation.