Analyze Text

Azure Language supports analysis of text, including language detection, sentiment analysis, key phrase extraction, and entity recognition.

For example, suppose a travel agency wants to process hotel reviews that have been submitted to the company's web site. By using the Azure AI Language, they can determine the language each review is written in, the sentiment (positive, neutral, or negative) of the reviews, key phrases that might indicate the main topics discussed in the review, and named entities, such as places, landmarks, or people mentioned in the reviews.

Provision an *Azure AI Language* resource

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

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**. Then, in the results, select **Create** under **Language Service**.
3. Select **Continue to create your resource**.
4. Provision the resource using 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.
5. Select **Review + create**.
6. Wait for deployment to complete, and then go to the deployed resource.
7. 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 text analytics 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, as well as a sample text file you'll use to test the summarization. Both apps feature the same functionality. First, you'll complete some key parts of the application to enable it to use your Azure AI Language resource.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Labfiles/01-analyze-text** folder and expand the **CSharp** or **Python** folder depending on your language preference and the **text-analytics** folder it contains. Each folder contains the language-specific files for an app into which you're you're going to integrate Azure AI Language text analytics functionality.
2. Right-click the **text-analytics** folder containing your code files and open an integrated terminal. Then install the Azure AI Language Text Analytics SDK package by running the appropriate command for your language preference. For the Python exercise, also install the `dotenv` package:

   C#:

   CodeCopy

   ```
   dotnet add package Azure.AI.TextAnalytics --version 5.3.0
   ```

   Python:

   CodeCopy

   ```
   pip install azure-ai-textanalytics==5.3.0
   pip install python-dotenv
   ```

3. In the **Explorer** pane, in the **text-analytics** folder, open the configuration file for your preferred language
    - **C#**: appsettings.json
    - **Python**: .env
4. Update the configuration values to include the **endpoint** and a **key** from the Azure Language resource you created (available on the **Keys and Endpoint** page for your Azure AI Language resource in the Azure portal)
5. Save the configuration file.
6. Note that the **text-analysis** folder contains a code file for the client application:
    - **C#**: Program.cs
    - **Python**: text-analysis.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 Text Analytics SDK:

    C#: Programs.cs

    C#Copy

```
// import namespaces
using Azure;
using Azure.AI.TextAnalytics;
```

    Python: text-analysis.py

    CodeCopy

```
# import namespaces
from azure.core.credentials import AzureKeyCredential
from azure.ai.textanalytics import TextAnalyticsClient
```

7. In the **Main** function, note that code to load the Azure AI Language service endpoint and key from the configuration file has already been provided. Then find the comment **Create client using endpoint and key**, and add the following code to create a client for the Text Analysis API:

    C#: Programs.cs

    CodeCopy

```
 // Create client using endpoint and key
 AzureKeyCredential credentials = new AzureKeyCredential(aiSvcKey);
 Uri endpoint = new Uri(aiSvcEndpoint);
 TextAnalyticsClient aiClient = new TextAnalyticsClient(endpoint,
credentials);
```

Python: text-analysis.py

CodeCopy

```
 # Create client using endpoint and key
 credential = AzureKeyCredential(ai_key)
 ai_client = TextAnalyticsClient(endpoint=ai_endpoint,
credential=credential)
```

8. Save your changes and return to the integrated terminal for the **text-analysis** folder, and enter the following command to run the program:
   - **C#**: `dotnet run`
   - **Python**: `python text-analysis.py`

   Tip: You can use the **Maximize panel size** (**^**) icon in the terminal toolbar to see more of the console text.

9. Observe the output as the code should run without error, displaying the contents of each review text file in the **reviews** folder. The application successfully creates a client for the Text Analytics API but doesn't make use of it. We'll fix that in the next procedure.

Add code to detect language

Now that you have created a client for the API, let's use it to detect the language in which each review is written.

1. In the **Main** function for your program, find the comment **Get language**. Then, under this comment, add the code necessary to detect the language in each review document:

   C#: Programs.cs

   C#Copy

```
 // Get language
 DetectedLanguage detectedLanguage = aiClient.DetectLanguage(text);
```

```
Console.WriteLine($"\nLanguage: {detectedLanguage.Name}");
```

Python: text-analysis.py

CodeCopy

```
# Get language
detectedLanguage = ai_client.detect_language(documents=[text])[0]
print('\nLanguage: {}'.format(detectedLanguage.primary_language.name))
```

> **Note**: *In this example, each review is analyzed individually, resulting in a separate call to the service for each file. An alternative approach is to create a collection of documents and pass them to the service in a single call. In both approaches, the response from the service consists of a collection of documents; which is why in the Python code above, the index of the first (and only) document in the response ([0]) is specified.*

2. Save your changes. Then return to the integrated terminal for the **text-analysis** folder, and re-run the program.
3. Observe the output, noting that this time the language for each review is identified.

Add code to evaluate sentiment

Sentiment analysis is a commonly used technique to classify text as *positive* or *negative* (or possible *neutral* or *mixed*). It's commonly used to analyze social media posts, product reviews, and other items where the sentiment of the text may provide useful insights.

1. In the **Main** function for your program, find the comment **Get sentiment**. Then, under this comment, add the code necessary to detect the sentiment of each review document:

   C#: Program.cs

   C#Copy

```
// Get sentiment
DocumentSentiment sentimentAnalysis = aiClient.AnalyzeSentiment(text);
Console.WriteLine($"\nSentiment: {sentimentAnalysis.Sentiment}");
```

   Python: text-analysis.py

CodeCopy

```
# Get sentiment
sentimentAnalysis = ai_client.analyze_sentiment(documents=[text])[0]
print("\nSentiment: {}".format(sentimentAnalysis.sentiment))
```

2. Save your changes. Then return to the integrated terminal for the **text-analysis** folder, and re-run the program.
3. Observe the output, noting that the sentiment of the reviews is detected.

Add code to identify key phrases

It can be useful to identify key phrases in a body of text to help determine the main topics that it discusses.

1. In the **Main** function for your program, find the comment **Get key phrases**. Then, under this comment, add the code necessary to detect the key phrases in each review document:

   C#: Program.cs

   C#Copy

```
// Get key phrases
KeyPhraseCollection phrases = aiClient.ExtractKeyPhrases(text);
if (phrases.Count > 0)
{
    Console.WriteLine("\nKey Phrases:");
    foreach(string phrase in phrases)
    {
        Console.WriteLine($"\t{phrase}");
    }
}
```

   Python: text-analysis.py

   CodeCopy

```
# Get key phrases
phrases = ai_client.extract_key_phrases(documents=[text])[0].key_phrases
if len(phrases) > 0:
    print("\nKey Phrases:")
    for phrase in phrases:
        print('\t{}'.format(phrase))
```

2. Save your changes. Then return to the integrated terminal for the **text-analysis** folder, and re-run the program.
3. Observe the output, noting that each document contains key phrases that give some insights into what the review is about.

Add code to extract entities

Often, documents or other bodies of text mention people, places, time periods, or other entities. The text Analytics API can detect multiple categories (and subcategories) of entity in your text.

1. In the **Main** function for your program, find the comment **Get entities**. Then, under this comment, add the code necessary to identify entities that are mentioned in each review:

   C#: Program.cs

   C#Copy

```
// Get entities
CategorizedEntityCollection entities = aiClient.RecognizeEntities(text);
if (entities.Count > 0)
{
    Console.WriteLine("\nEntities:");
    foreach(CategorizedEntity entity in entities)
    {
        Console.WriteLine($"\t{entity.Text} ({entity.Category})");
    }
}
```

   Python: text-analysis.py

   CodeCopy

```
# Get entities
entities = ai_client.recognize_entities(documents=[text])[0].entities
if len(entities) > 0:
    print("\nEntities")
    for entity in entities:
        print('\t{} ({})'.format(entity.text, entity.category))
```

2. Save your changes. Then return to the integrated terminal for the **text-analysis** folder, and re-run the program.
3. Observe the output, noting the entities that have been detected in the text.

Add code to extract linked entities

In addition to categorized entities, the Text Analytics API can detect entities for which there are known links to data sources, such as Wikipedia.

1. In the **Main** function for your program, find the comment **Get linked entities**. Then, under this comment, add the code necessary to identify linked entities that are mentioned in each review:

   C#: Program.cs

   C#Copy

```
 // Get linked entities
 LinkedEntityCollection linkedEntities =
aiClient.RecognizeLinkedEntities(text);
 if (linkedEntities.Count > 0)
 {
     Console.WriteLine("\nLinks:");
     foreach(LinkedEntity linkedEntity in linkedEntities)
     {
         Console.WriteLine($"\t{linkedEntity.Name} ({linkedEntity.Url})");
     }
 }
```

   Python: text-analysis.py

   CodeCopy

```
 # Get linked entities
 entities =
ai_client.recognize_linked_entities(documents=[text])[0].entities
 if len(entities) > 0:
     print("\nLinks")
     for linked_entity in entities:
         print('\t{} ({})'.format(linked_entity.name, linked_entity.url))
```

2. Save your changes. Then return to the integrated terminal for the **text-analysis** folder, and re-run the program.
3. Observe the output, noting the linked entities that are identified.

Clean up resources

If you're finished exploring the Azure AI Language service, you can delete the resources you created in this exercise. Here's how:

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. Browse to the Azure AI Language resource you created in this lab.
3. On the resource page, select **Delete** and follow the instructions to delete the resource.