Analyze Text

Azure AI 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. In this exercise, you'll use the Azure AI Language Python SDK for text analytics to implement a simple hotel review application based on this example.

While this exercise is based on Python, you can develop text analytics applications using multiple language-specific SDKs; including:

- Azure AI Text Analytics client library for Python
- Azure AI Text Analytics client library for .NET
- Azure AI Text Analytics client library for JavaScript

This exercise takes approximately **30** minutes.

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. Select **Create a resource**.
3. In the search field, search for **Language service**. Then, in the results, select **Create** under **Language Service**.
4. Select **Continue to create your resource**.
5. 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.
6. Select **Review + create**, then select **Create** to provision the resource.
7. Wait for deployment to complete, and then go to the deployed resource.
8. View the **Keys and Endpoint** page in the **Resource Management** section. You will need the information on this page later in the exercise.

Clone the repository for this course

You'll develop your code using Cloud Shell from the Azure Portal. The code files for your app have been provided in a GitHub repo.

1. In the Azure Portal, use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal.

> **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

2. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.

3. In the PowerShell pane, enter the following commands to clone the GitHub repo for this exercise:

```
rm -r mslearn-ai-language -f
git clone https://github.com/microsoftlearning/mslearn-ai-language
```

> **Tip**: As you enter commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

4. After the repo has been cloned, navigate to the folder containing the application code files:

```
cd mslearn-ai-language/Labfiles/01-analyze-text/Python/text-analysis
```

Configure your application

1. In the command line pane, run the following command to view the code files in the **text-analysis** folder:

```
ls -a -l
```

   The files include a configuration file (**.env**) and a code file (**text-analysis.py**). The text your application will analyze is in the **reviews** subfolder.

2. Create a Python virtual environment and install the Azure AI Language Text Analytics SDK package and other required packages by running the following command:

```
python -m venv labenv
./labenv/bin/Activate.ps1
pip install -r requirements.txt azure-ai-textanalytics==5.3.0
```

3. Enter the following command to edit the application configuration file:

```
code .env
```

   The file is opened in a code editor.

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. After you've replaced the placeholders, within the code editor, use the **CTRL+S** command or **Right-click > Save** to save your changes and then use the **CTRL+Q** command or **Right-click > Quit** to close the code editor while keeping the cloud shell command line open.

Add code to connect to your Azure AI Language resource

1. Enter the following command to edit the application code file:

```
code text-analysis.py
```

2. Review the existing code. You will add code to work with the AI Language Text Analytics SDK.

> **Tip**: As you add code to the code file, be sure to maintain the correct indentation.

3. At the top of the code file, under the existing namespace references, find the comment **Import namespaces** and add the following code to import the namespaces you will need to use the Text Analytics SDK:

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

4. 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:

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

5. Save your changes (CTRL+S), then enter the following command to run the program (you maximize the cloud shell pane and resize the panels to see more text in the command line pane):

```
python text-analysis.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 successfully creates a client for the Text Analytics API but doesn't make use of it. We'll fix that in the next section.

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 code editor, find the comment **Get language**. Then add the code necessary to detect the language in each review document:

```
# 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 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 code editor, find the comment **Get sentiment**. Then add the code necessary to detect the sentiment of each review document:

Code Copy

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

2. Save your changes. Then close the code editor 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 code editor, find the comment **Get key phrases**. Then add the code necessary to detect the key phrases in each review document:

Code Copy

```
# 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 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 code editor, find the comment **Get entities**. Then, add the code necessary to identify entities that are mentioned in each review:

Code Copy

```
# 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 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 code editor, find the comment **Get linked entities**. Then, add the code necessary to identify linked entities that are mentioned in each review:

Code Copy

```
# 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 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. Close the Azure cloud shell pane
2. In the Azure portal, 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.

More information

For more information about using **Azure AI Language**, see the documentation.