

# Analyze Text

The **Language** service is a cognitive service that 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 Language service, 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.

## Clone the repository for this course

If you have not already cloned **AI-102-AIEngineer** code repository to the environment where you're working on this lab, follow these steps to do so. Otherwise, open the cloned folder in Visual Studio Code.

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 the endpoint and one of the keys from this page in the next procedure.

## Prepare to use the Language SDK for text analytics


In this exercise, you'll complete a partially implemented client application that uses the Language service text analytics SDK to analyze hotel reviews.

 **Note:** You can choose to use the SDK for 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 **05-analyze-text** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.

2. Right-click the **text-analysis** folder and open an integrated terminal. Then install the Text Analytics SDK package by running the appropriate command for your language preference:

#### C#

Code	 Copy
<pre>dotnet add package Azure.AI.TextAnalytics --version 5.1.0</pre>	

#### Python

Code	 Copy
<pre>pip install azure-ai-textanalytics==5.1.0</pre>	

3. View the contents of the **text-analysis** 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 reflect the **endpoint** and an authentication **key** for your cognitive services resource. Save your changes.

4. 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#


Code	 Copy
<pre>// import namespaces using Azure; using Azure.AI.TextAnalytics;</pre>	

#### Python


Code	 Copy
<pre># import namespaces from azure.core.credentials import AzureKeyCredential from azure.ai.textanalytics import TextAnalyticsClient</pre>	

5. In the **Main** function, note that code to load the cognitive services 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#


Code	 Copy
<pre>// Create client using endpoint and key AzureKeyCredential credentials = new AzureKeyCredential(cogSvcKey); Uri endpoint = new Uri(cogSvcEndpoint); TextAnalyticsClient CogClient = new TextAnalyticsClient(endpoint, credentials);</pre>	

## Python

Code	 Copy
<pre># Create client using endpoint and key credential = AzureKeyCredential(cog_key) cog_client = TextAnalyticsClient(endpoint=cog_endpoint, credential=credential)</pre>	

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

## C#

Code	 Copy
<pre>dotnet run</pre>	

## Python

Code	 Copy
<pre>python text-analysis.py</pre>	


7. 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.

## Detect language


Now that you have created a client for the Text Analytics 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#

Code	 Copy
<pre>// Get language DetectedLanguage detectedLanguage = CogClient.DetectLanguage(text); Console.WriteLine(\$"Language: {detectedLanguage.Name}");</pre>	

## Python

Code	 Copy
<pre># Get language detectedLanguage = cog_client.detect_language(documents=[text])[0] print('Language: {}'.format(detectedLanguage.primary_language.name))</pre>	


 **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 and return to the integrated terminal for the **text-analysis** folder, and enter the following command to run the program:

**C#**

Code	 Copy
<pre>dotnet run</pre>	

**Python**

Code	 Copy
<pre>python text-analysis.py</pre>	


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

## 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#**


Code	 Copy
<pre>// Get sentiment DocumentSentiment sentimentAnalysis = CogClient.AnalyzeSentiment(text); Console.WriteLine(\$"{Environment.NewLine}Sentiment: {sentimentAnalysis.Sentiment}");</pre>	

**Python**


Code	 Copy
<pre># Get sentiment sentimentAnalysis = cog_client.analyze_sentiment(documents=[text])[0] print(\$"{Environment.NewLine}Sentiment: {}".format(sentimentAnalysis.sentiment))</pre>	

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

**C#**

Code	 Copy
<pre>dotnet run</pre>	

**Python**

Code	 Copy
<pre>python text-analysis.py</pre>	


3. Observe the output, noting that the sentiment of the reviews is detected.

## 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#


Code	 Copy
<pre>// Get key phrases KeyPhraseCollection phrases = CogClient.ExtractKeyPhrases(text); if (phrases.Count &gt; 0) {     Console.WriteLine("\nKey Phrases:");     foreach(string phrase in phrases)     {         Console.WriteLine(\$"{t{phrase}");     } }</pre>	

### Python


Code	 Copy
<pre># Get key phrases phrases = cog_client.extract_key_phrases(documents=[text])[0].key_phrases if len(phrases) &gt; 0:     print("\nKey Phrases:")     for phrase in phrases:         print('{t}'.format(phrase))</pre>	

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

### C#

Code	 Copy
<pre>dotnet run</pre>	

### Python

Code	 Copy
<pre>python text-analysis.py</pre>	


3. Observe the output, noting that each document contains key phrases that give some insights into what the review is about.

## 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#


Code	 Copy
<pre>// Get entities CategorizedEntityCollection entities = CogClient.RecognizeEntities(text); if (entities.Count &gt; 0) {     Console.WriteLine("\nEntities:");     foreach(CategorizedEntity entity in entities)     {         Console.WriteLine(\$"{entity.Text} ({entity.Category})");     } }</pre>	

#### Python


Code	 Copy
<pre># Get entities entities = cog_client.recognize_entities(documents=[text])[0].entities if len(entities) &gt; 0:     print("\nEntities")     for entity in entities:         print('\t{} {}'.format(entity.text, entity.category))</pre>	

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

#### C#

Code	 Copy
<pre>dotnet run</pre>	

#### Python

Code	 Copy
<pre>python text-analysis.py</pre>	


3. Observe the output, noting the entities that have been detected in the text.

## 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#

Code	 Copy
------	--

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

### Python

Code

 Copy

```
# Get linked entities
entities = cog_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 return to the integrated terminal for the **text-analysis** folder, and enter the following command to run the program:

### C#

Code

 Copy

```
dotnet run
```

### Python

Code

 Copy

```
python text-analysis.py
```

3. Observe the output, noting the linked entities that are identified.

## More information

For more information about using the **Language** service, see the [Text Analytics documentation](#).

