Custom text classification

Azure AI Language provides several NLP capabilities, including the key phrase identification, text summarization, and sentiment analysis. The Language service also provides custom features like custom question answering and custom text classification.

To test the custom text classification of the Azure AI Language service, we'll configure the model using Language Studio then use a small command-line application that runs in the Cloud Shell to test it. The same pattern and functionality used here can be followed for real-world applications.

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. Additionally, use custom text classification, you need to enable the **Custom text classification & extraction** feature.

1. In a browser, open the Azure portal at `https://portal.azure.com`, and sign in with your Microsoft account.
2. Select the search field at the top of the portal, search for `Azure AI services`, and create a **Language Service** resource.
3. Select the box that includes **Custom text classification**. Then select **Continue to create your resource**.
4. Create a resource with the following settings:
    - **Subscription**: *Your Azure subscription*.
    - **Resource group**: *Select 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.
    - **Storage account**: New storage account
        - **Storage account name**: *Enter a unique name*.
        - **Storage account type**: Standard LRS
    - **Responsible AI notice**: Selected.
5. Select **Review + create,** then select **Create** to provision the resource.
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.

Upload sample articles

Once you've created the Azure AI Language service and storage account, you'll need to upload example articles to train your model later.

1. In a new browser tab, download sample articles from `https://aka.ms/classification-articles` and extract the files to a folder of your choice.
2. In the Azure portal, navigate to the storage account you created, and select it.
3. In your storage account select **Configuration**, located below **Settings**. In the Configuration screen enable the option to **Allow Blob anonymous access** then select **Save**.
4. Select **Containers** in the left menu, located below **Data storage**. On the screen that appears, select **+ Container**. Give the container the name `articles`, and set **Anonymous access level** to **Container (anonymous read access for containers and blobs)**.

   > **NOTE**: When you configure a storage account for a real solution, be careful to assign the appropriate access level. To learn more about each access level, see the Azure Storage documentation.

5. After you've created the container, select it then select the **Upload** button. Select **Browse for files** to browse for the sample articles you downloaded. Then select **Upload**.

Create a custom text classification project

After configuration is complete, create a custom text classification project. This project provides a working place to build, train, and deploy your model.

> **NOTE**: This lab utilizes **Language Studio**, but you can also create, build, train, and deploy your model through the REST API.

1. In a new browser tab, open the Azure AI Language Studio portal at `https://language.cognitive.azure.com/` and sign in using the Microsoft account associated with your Azure subscription.
2. If prompted to choose a Language resource, select the following settings:
   - **Azure Directory**: The Azure directory containing your subscription.
   - **Azure subscription**: Your Azure subscription.
   - **Resource type**: Language.
   - **Language resource**: The Azure AI Language resource you created previously.

   If you are not prompted to choose a language resource, it may be because you have multiple Language resources in your subscription; in which case:

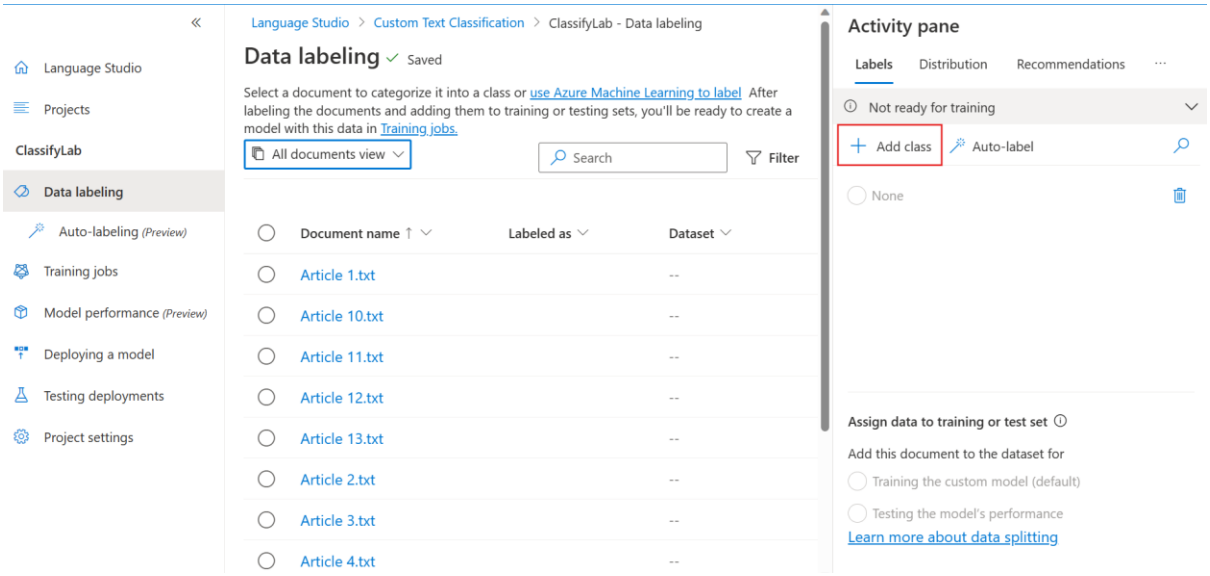
e. On the bar at the top if the page, select the **Settings (⚙)** button.

    f. On the **Settings** page, view the **Resources** tab.
    g. Select the language resource you just created, and click **Switch resource**.
    h. At the top of the page, click **Language Studio** to return to the Language Studio home page
3. At the top of the portal, in the **Create new** menu, select **Custom text classification**.
4. The **Connect storage** page appears. All values will already have been filled. So select **Next**.
5. On the **Select project type** page, select **Single label classification**. Then select **Next**.
6. On the **Enter basic information** pane, set the following:
    - **Name**: `ClassifyLab`
    - **Text primary language**: English (US)
    - **Description**: `Custom text lab`
7. Select **Next**.
8. On the **Choose container** page, set the **Blob store container** dropdown to your *articles* container.
9. Select the **No, I need to label my files as part of this project** option. Then select **Next**.
10. Select **Create project**.

Label your data

Now that your project is created, you need to label, or tag, your data to train your model how to classify text.

1. On the left, select **Data labeling**, if not already selected. You'll see a list of the files you uploaded to your storage account.
2. On the right side, in the **Activity** pane, select **+ Add class**. The articles in this lab fall into four classes you'll need to create: `Classifieds`, `Sports`, `News`, and `Entertainment`.



3. After you've created your four classes, select **Article 1** to start. Here you can read the article, define which class this file is, and which dataset (training or testing) to assign it to.
4. Assign each article the appropriate class and dataset (training or testing) using the **Activity** pane on the right. You can select a label from the list of labels on the right, and set each article to **training** or **testing** using the options at the bottom of the Activity pane. You select **Next document** to move to the next document. For the purposes of this lab, we'll define which are to be used for training the model and testing the model:

Article	Class	Dataset
Article 1	Sports	Training
Article 10	News	Training
Article 11	Entertainment	Testing
Article 12	News	Testing
Article 13	Sports	Testing
Article 2	Sports	Training
Article 3	Classifieds	Training
Article 4	Classifieds	Training

Article	Class	Dataset
Article 5	Entertainment	Training
Article 6	Entertainment	Training
Article 7	News	Training
Article 8	News	Training
Article 9	Entertainment	Training

5. **NOTE** Files in Language Studio are listed alphabetically, which is why the above list is not in sequential order. Make sure you visit both pages of documents when labeling your articles.

6. Select **Save labels** to save your labels.

Train your model

After you've labeled your data, you need to train your model.

1. Select **Training jobs** on the left side menu.
2. Select **Start a training job**.
3. Train a new model named `ClassifyArticles`.
4. Select **Use a manual split of training and testing data**.

   TIP In your own classification projects, the Azure AI Language service will automatically split the testing set by percentage which is useful with a large dataset. With smaller datasets, it's important to train with the right class distribution.

5. Select **Train**

IMPORTANT Training your model can sometimes take several minutes. You'll get a notification when it's complete.

Evaluate your model

In real world applications of text classification, it's important to evaluate and improve your model to verify it's performing as you expect.

1. Select **Model performance**, and select your **ClassifyArticles** model. There you can see the scoring of your model, performance metrics, and when it was trained. If the scoring of your model isn't 100%, it means that one of the

documents used for testing didn't evaluate to what it was labeled. These failures can help you understand where to improve.

2. Select **Test set details** tab. If there are any errors, this tab allows you to see the articles you indicated for testing and what the model predicted them as and whether that conflicts with their test label. The tab defaults to show incorrect predictions only. You can toggle the **Show mismatches only** option to see all the articles you indicated for testing and what they each of them predicted as.

Deploy your model

When you're satisfied with the training of your model, it's time to deploy it, which allows you to start classifying text through the API.

1. On the left panel, select **Deploying model**.
2. Select **Add deployment**, then enter `articles` in the **Create a new deployment name** field, and select **ClassifyArticles** in the **Model** field.
3. Select **Deploy** to deploy your model.
4. Once your model is deployed, leave that page open. You'll need your project and deployment name in the next step.

Prepare to develop an app in Visual Studio Code

To test the custom text classification capabilities of the Azure AI Language service, you'll develop a simple console application in Visual Studio Code.

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/04-text-classification** folder and expand the **CSharp** or **Python** folder depending on your language preference and the **classify-text** 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 classification functionality.
2. Right-click the **classify-text** 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:

   C#:

   CodeCopy

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

   Python:

   CodeCopy

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

3. In the **Explorer** pane, in the **classify-text** 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). The fil should already contain the project and deployment names for your text classification model.
5. Save the configuration file.

Add code to classify documents

Now you're ready to use the Azure AI Language service to classify documents.

1. Expand the **articles** folder in the **classify-text** folder to view the text articles that your application will classify.
2. In the **classify-text** folder, open the code file for the client application:
    - **C#**: Program.cs
    - **Python**: classify-text.py
3. 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: classify-text.py

    CodeCopy

```
# 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 and the project and deployment names 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

    C#Copy

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

    Python: classify-text.py

CodeCopy

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

5. in the **Main** function, note that the existing code reads all of the files in the **articles** folder and creates a list containing their contents. Then find the comment **Get Classifications** and add the following code:

   C#: Program.cs

   C#Copy

```
 // Get Classifications
 ClassifyDocumentOperation operation = await
aiClient.SingleLabelClassifyAsync(WaitUntil.Completed, batchedDocuments,
projectName, deploymentName);

 int fileNo = 0;
 await foreach (ClassifyDocumentResultCollection documentsInPage in
operation.Value)
 {

     foreach (ClassifyDocumentResult documentResult in documentsInPage)
     {
         Console.WriteLine(files[fileNo].Name);
         if (documentResult.HasError)
         {
             Console.WriteLine($"  Error!");
             Console.WriteLine($"  Document error code:
{documentResult.Error.ErrorCode}");
             Console.WriteLine($"  Message:
{documentResult.Error.Message}");
             continue;
         }

         Console.WriteLine($"  Predicted the following class:");
         Console.WriteLine();

         foreach (ClassificationCategory classification in
documentResult.ClassificationCategories)
         {
             Console.WriteLine($"  Category: {classification.Category}");
             Console.WriteLine($"  Confidence score:
{classification.ConfidenceScore}");
             Console.WriteLine();
         }
         fileNo++;
     }
 }
```

Python: classify-text.py

CodeCopy

```
# Get Classifications
operation = ai_client.begin_single_label_classify(
    batchedDocuments,
    project_name=project_name,
    deployment_name=deployment_name
)

document_results = operation.result()

for doc, classification_result in zip(files, document_results):
    if classification_result.kind == "CustomDocumentClassification":
        classification = classification_result.classifications[0]
        print("{} was classified as '{}' with confidence score
{}.".format(
            doc, classification.category, classification.confidence_score)
        )
    elif classification_result.is_error is True:
        print("{} has an error with code '{}' and message '{}'".format(
            doc, classification_result.error.code,
classification_result.error.message)
        )
```

6. Save the changes to your code file.

Test your application

Now your application is ready to test.

1. In the integrated terminal for the **classify-text** folder, and enter the following command to run the program:
    - **C#**: `dotnet run`
    - **Python**: `python classify-text.py`

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

2. Observe the output. The application should list a classification and confidence score for each text file.

Clean up

When you don’t need your project anymore, you can delete if from your **Projects** page in Language Studio. You can also remove the Azure AI Language service and associated storage account in the Azure portal.