AI-LEARN-02d OCR (Learn)

57 Minutes Remaining

Instructions Resources Help  100%

**Tip**: As you follow the instructions in this pane, whenever you see a icon, you can use it to copy text from the instruction pane into the virtual machine interface. This is particularly useful to copy code; but bear in mind you may need to modify the pasted code to fix indent levels or formatting before running it!

Log into Windows as **Student** account with the password Pa55w.rd.

Read Text in Images

Optical character recognition (OCR) is a subset of computer vision that deals with reading text in images and documents. The **Azure AI Vision** service provides an API for reading text, which you'll explore in this exercise.

Clone the repository for this course

If you haven't already done so, you must clone the code repository for this course:

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-vision 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**. If you are prompted with the Message *Detected an Azure Function Project in folder*, you can safely close that message.

Provision an Azure AI Services resource

If you don't already have one in your subscription, you'll need to provision an **Azure AI 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. In the top search bar, search for *Azure AI services*, select **Azure AI Services**, and create an Azure AI services multi-service account 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 from East US, France Central, Korea Central, North Europe, Southeast Asia, West Europe, West US, or East Asia\**
   * **Name**: *Enter a unique name*
   * **Pricing tier**: Standard S0

\*Azure AI Vision 4.0 features are currently only available in these regions.

1. Select the required checkboxes and create the resource.
2. Wait for deployment to complete, and then view the deployment details.
3. When the resource has been deployed, go to it and view its **Keys and Endpoint** page. You'll need the endpoint and one of the keys from this page in the next procedure.

Prepare to use the Azure AI Vision SDK

In this exercise, you'll complete a partially implemented client application that uses the Azure AI Vision SDK to read text.

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

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Labfiles\05-ocr** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.
2. Right-click the **read-text** folder and open an integrated terminal. Then install the Azure AI Vision SDK package by running the appropriate command for your language preference:

**C#**

csharp

dotnet add package Azure.AI.Vision.ImageAnalysis -v 0.15.1-beta.1

**Note**: If you are prompted to install dev kit extensions, you can safely close the message.

**Python**

python

pip install azure-ai-vision==0.15.1b1

1. View the contents of the **read-text** 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 Azure AI services resource. Save your changes.

Use the Azure AI Vision SDK to read text from an image

one of the features of the **Azure AI Vision SDK** is to read text from an image. In this exercise, you'll complete a partially implemented client application that uses the Azure AI Vision SDK to read text from an image.

1. The **read-text** folder contains a code file for the client application:
   * **C#**: Program.cs
   * **Python**: read-text.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'll need to use the Azure AI Vision SDK:

**C#**

C#

// Import namespaces

using Azure.AI.Vision.Common;

using Azure.AI.Vision.ImageAnalysis;

**Python**

Python

# Import namespaces

import azure.ai.vision as sdk

1. In the code file for your client application, in the **Main** function, the code to load the configuration settings has been provided. Then find the comment **Authenticate Azure AI Vision client**. Then, under this comment, add the following language-specific code to create and authenticate an Azure AI Vision client object:

**C#**

C#

// Authenticate Azure AI Vision client

var cvClient = new VisionServiceOptions(

aiSvcEndpoint,

new AzureKeyCredential(aiSvcKey));

**Python**

Python

# Authenticate Azure AI Vision client

cv\_client = sdk.VisionServiceOptions(ai\_endpoint, ai\_key)

1. In the **Main** function, under the code you just added, note that the code specifies the path to an image file and then passes the image path to the **GetTextRead** function. This function isn't yet fully implemented.
2. Let's add some code to the body of the **GetTextRead** function. Find the comment **Use Analyze image function to read text in image**. Then, under this comment, add the following language-specific code:

**C#**

C#

// Use Analyze image function to read text in image

Console.WriteLine($"\nReading text in {imageFile}\n");

using (var imageData = File.OpenRead(imageFile))

{

var analysisOptions = new ImageAnalysisOptions()

{

// Specify features to be retrieved

};

using var imageSource = VisionSource.FromFile(imageFile);

using var analyzer = new ImageAnalyzer(serviceOptions, imageSource, analysisOptions);

var result = analyzer.Analyze();

if (result.Reason == ImageAnalysisResultReason.Analyzed)

{

// get image captions

if (result.Text != null)

{

Console.WriteLine($"Text:");

// Prepare image for drawing

System.Drawing.Image image = System.Drawing.Image.FromFile(imageFile);

Graphics graphics = Graphics.FromImage(image);

Pen pen = new Pen(Color.Cyan, 3);

foreach (var line in result.Text.Lines)

{

// Return the text detected in the image

}

// Save image

String output\_file = "text.jpg";

image.Save(output\_file);

Console.WriteLine("\nResults saved in " + output\_file + "\n");

}

}

}

**Python**

Python

# Use Analyze image function to read text in image

print('Reading text in {}\n'.format(image\_file))

analysis\_options = sdk.ImageAnalysisOptions()

features = analysis\_options.features = (

# Specify the features to be retrieved

)

# Get image analysis

image = sdk.VisionSource(image\_file)

image\_analyzer = sdk.ImageAnalyzer(cv\_client, image, analysis\_options)

result = image\_analyzer.analyze()

if result.reason == sdk.ImageAnalysisResultReason.ANALYZED:

# Get image captions

if result.text is not None:

print("\nText:")

# Prepare image for drawing

image = Image.open(image\_file)

fig = plt.figure(figsize=(image.width/100, image.height/100))

plt.axis('off')

draw = ImageDraw.Draw(image)

color = 'cyan'

for line in result.text.lines:

# Return the text detected in the image

# Save image

plt.imshow(image)

plt.tight\_layout(pad=0)

outputfile = 'text.jpg'

fig.savefig(outputfile)

print('\n Results saved in', outputfile)

1. Now that the body of the **GetTextRead** function has been added, under the comment **Specify features to be retrieved**, add the following code to specify that you want to retrieve text:

**C#**

C#

// Specify features to be retrieved

Features =

ImageAnalysisFeature.Text

**Python**

Python

# Specify features to be retrieved

sdk.ImageAnalysisFeature.TEXT

1. In the code file in Visual Studio Code, find the **GetTextRead** function, and under the **Return the text detected in the image** comment, add the following code (this code prints the image text to the console and generates the image **text.jpg** which highlights the image's text):

**C#**

C#

// Return the text detected in the image

Console.WriteLine(line.Content);

var drawLinePolygon = true;

// Return each line detected in the image and the position bounding box around each line

// Return each word detected in the image and the position bounding box around each word with the confidence level of each word

// Draw line bounding polygon

if (drawLinePolygon)

{

var r = line.BoundingPolygon;

Point[] polygonPoints = {

new Point(r[0].X, r[0].Y),

new Point(r[1].X, r[1].Y),

new Point(r[2].X, r[2].Y),

new Point(r[3].X, r[3].Y)

};

graphics.DrawPolygon(pen, polygonPoints);

}

**Python**

Python

# Return the text detected in the image

print(line.content)

drawLinePolygon = True

r = line.bounding\_polygon

bounding\_polygon = ((r[0], r[1]),(r[2], r[3]),(r[4], r[5]),(r[6], r[7]))

# Return each line detected in the image and the position bounding box around each line

# Return each word detected in the image and the position bounding box around each word with the confidence level of each word

# Draw line bounding polygon

if drawLinePolygon:

draw.polygon(bounding\_polygon, outline=color, width=3)

1. In the **read-text/images** folder, select **Lincoln.jpg** to view the file that your code processes.
2. In the code file for your application, in the **Main** function, examine the code that runs if the user selects menu option **1**. This code calls the **GetTextRead** function, passing the path to the *Lincoln.jpg* image file.
3. Save your changes and return to the integrated terminal for the **read-text** folder, and enter the following command to run the program:

**C#**

dotnet run

**Python**

python read-text.py

1. When prompted, enter **1** and observe the output, which is the text extracted from the image.
2. In the **read-text** folder, select the **text.jpg** image and noticed how there's a polygon around each *line* of text.
3. Return to the code file in Visual Studio Code, and find the comment **Return each line detected in the image and the position bounding box around each line**. Then, under this comment, add the following code:

**C#**

C#

// Return each line detected in the image and the position bounding box around each line

string pointsToString = "{" + string.Join(',', line.BoundingPolygon.Select(pointsToString => pointsToString.ToString())) + "}";

Console.WriteLine($" Line: '{line.Content}', Bounding Polygon {pointsToString}");

**Python**

Python

# Return each line detected in the image and the position bounding box around each line

print(" Line: '{}', Bounding Polygon: {}".format(line.content, bounding\_polygon))

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

**C#**

dotnet run

**Python**

python read-text.py

1. When prompted, enter **1** and observe the output, which should be each line of text in the image with their respective position in the image.
2. Return to the code file in Visual Studio Code, and find the comment **Return each word detected in the image and the position bounding box around each word with the confidence level of each word**. Then, under this comment, add the following code:

**C#**

C#

// Return each word detected in the image and the position bounding box around each word with the confidence level of each word

foreach (var word in line.Words)

{

pointsToString = "{" + string.Join(',', word.BoundingPolygon.Select(pointsToString => pointsToString.ToString())) + "}";

Console.WriteLine($" Word: '{word.Content}', Bounding polygon {pointsToString}, Confidence {word.Confidence:0.0000}");

// Draw word bounding polygon

drawLinePolygon = false;

var r = word.BoundingPolygon;

Point[] polygonPoints = {

new Point(r[0].X, r[0].Y),

new Point(r[1].X, r[1].Y),

new Point(r[2].X, r[2].Y),

new Point(r[3].X, r[3].Y)

};

graphics.DrawPolygon(pen, polygonPoints);

}

**Python**

Python

# Return each word detected in the image and the position bounding box around each word with the confidence level of each word

for word in line.words:

r = word.bounding\_polygon

bounding\_polygon = ((r[0], r[1]),(r[2], r[3]),(r[4], r[5]),(r[6], r[7]))

print(" Word: '{}', Bounding Polygon: {}, Confidence: {}".format(word.content, bounding\_polygon,word.confidence))

# Draw word bounding polygon

drawLinePolygon = False

draw.polygon(bounding\_polygon, outline=color, width=3)

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

**C#**

dotnet run

**Python**

python read-text.py

1. When prompted, enter **1** and observe the output, which should be each word of text in the image with their respective position in the image. Notice how the confidence level of each word is also returned.
2. In the **read-text** folder, select the **text.jpg** image and noticed how there's a polygon around each *word*.

Use the Azure AI Vision SDK to read handwritten text from an image

In the previous exercise, you read well defined text from an image, but sometimes you might also want to read text from handwritten notes or papers. The good news is that the **Azure AI Vision SDK** can also read handwritten text with the same exact code you used to read well defined text. We'll use the same code from the previous exercise, but this time we'll use a different image.

1. In the **read-text/images** folder, select on **Note.jpg** to view the file that your code processes.
2. In the code file for your application, in the **Main** function, examine the code that runs if the user selects menu option **2**. This code calls the **GetTextRead** function, passing the path to the *Note.jpg* image file.
3. From the integrated terminal for the **read-text** folder, enter the following command to run the program:

**C#**

dotnet run

**Python**

python read-text.py

1. When prompted, enter **2** and observe the output, which is the text extracted from the note image.
2. In the **read-text** folder, select the **text.jpg** image and noticed how there's a polygon around each *word* of the note.

Clean up resources

If you're not using the Azure resources created in this lab for other training modules, you can delete them to avoid incurring further charges. 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. In the top search bar, search for *Azure AI services multi-service account*, and select the Azure AI services multi-service account 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 the **Azure AI Vision** service to read text, see the [Azure AI Vision documentation](https://learn.microsoft.com/azure/ai-services/computer-vision/overview-ocr).