Read Text in Images

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

Clone the repository for this course

If you have not 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/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 Computer Vision SDK

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

> **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 **20-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 Computer Vision SDK package by running the appropriate command for your language preference:

C#

```
Code
```

Copy

```
dotnet add package Microsoft.Azure.CognitiveServices.Vision.ComputerVision --version 6.0.0
```

Python

Code Copy

```
pip install azure-cognitiveservices-vision-computervision==0.7.0
```

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 cognitive services resource. Save your changes.

2. Note that 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 will need to use the Computer Vision SDK:

C#

Code Copy

```
// import namespaces
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision;
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision.Models;
```

Python

Code Copy

```
# import namespaces
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from azure.cognitiveservices.vision.computervision.models import OperationStatusCodes
from msrest.authentication import CognitiveServicesCredentials
```

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

C#

Code Copy

```
// Authenticate Computer Vision client
ApiKeyServiceClientCredentials credentials = new ApiKeyServiceClientCredentials(cogSvcKey);
cvClient = new ComputerVisionClient(credentials)
{
    Endpoint = cogSvcEndpoint
};
```

Python

Code Copy

```
# Authenticate Computer Vision client
credential = CognitiveServicesCredentials(cog_key)
cv_client = ComputerVisionClient(cog_endpoint, credential)
```

Use the OCR API

The **OCR** API is an optical character recognition API that is optimized for reading small to medium amounts of printed text in *.jpg*, *.png*, *.gif*, and *.bmp* format images. It supports a wide range of languages and in addition to reading text in the image it can determine the orientation of each text region and return information about the rotation angle of the text in relation to the image

1. 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 **GetTextOcr** function, passing the path to an image file.
2. In the **read-text/images** folder, open **Lincoln.jpg** to view the image that your code will process.
3. Back in the code file, find the **GetTextOcr** function, and under the existing code that prints a message to the console, add the following code:

C#

Code Copy

```
// Use OCR API to read text in image
using (var imageData = File.OpenRead(imageFile))
{
    var ocrResults = await cvClient.RecognizePrintedTextInStreamAsync(detectOrientation:false,
image:imageData);

    // Prepare image for drawing
    Image image = Image.FromFile(imageFile);
    Graphics graphics = Graphics.FromImage(image);
    Pen pen = new Pen(Color.Magenta, 3);

    foreach(var region in ocrResults.Regions)
    {
        foreach(var line in region.Lines)
        {
            // Show the position of the line of text
            int[] dims = line.BoundingBox.Split(",").Select(int.Parse).ToArray();
            Rectangle rect = new Rectangle(dims[0], dims[1], dims[2], dims[3]);
            graphics.DrawRectangle(pen, rect);

            // Read the words in the line of text
            string lineText = "";
            foreach(var word in line.Words)
            {
                lineText += word.Text + " ";
            }
            Console.WriteLine(lineText.Trim());
        }
    }

    // Save the image with the text locations highlighted
    String output_file = "ocr_results.jpg";
    image.Save(output_file);
    Console.WriteLine("Results saved in " + output_file);
}
```

Python

Code Copy

```
# Use OCR API to read text in image
with open(image_file, mode="rb") as image_data:
    ocr_results = cv_client.recognize_printed_text_in_stream(image_data)

# Prepare image for drawing
fig = plt.figure(figsize=(7, 7))
img = Image.open(image_file)
draw = ImageDraw.Draw(img)

# Process the text line by line
for region in ocr_results.regions:
    for line in region.lines:

        # Show the position of the line of text
        l,t,w,h = list(map(int, line.bounding_box.split(',')))
        draw.rectangle(((l,t), (l+w, t+h)), outline='magenta', width=5)

        # Read the words in the line of text
        line_text = ''
        for word in line.words:
            line_text += word.text + ' '
        print(line_text.rstrip())

# Save the image with the text locations highlighted
plt.axis('off')
plt.imshow(img)
outputfile = 'ocr_results.jpg'
fig.savefig(outputfile)
print('Results saved in', outputfile)
```

1. Examine the code you added to the **GetTextOcr** function. It detects regions of printed text from an image file, and for each region it extracts the lines of text and highlights there position on the image. It then extracts the words in each line and prints them.
2. Save your changes and return to the integrated terminal for the **read-text** folder, and enter the following command to run the program:

C#

Code Copy

```
dotnet run
```

*The C# output may display warnings about asynchronous functions now using the **await** operator. You can ignore these.*

Python

Code Copy

```
python read-text.py
```

1. When prompted, enter **1** and observe the output, which is the text extracted from the image.
2. View the **ocr_results.jpg** file that is generated in the same folder as your code file to see the annotated lines of text in the image.

Use the Read API

The **Read** API uses a newer text recognition model than the OCR API, and performs better for larger images that contain a lot of text. It also supports text extraction from *.pdf* files, and can recognize both printed text (in multiple languages) and handwritten text (in English).

The **Read** API uses an asynchronous operation model, in which a request to start text recognition is submitted; and the operation ID returned from the request can subsequently be used to check progress and retrieve results.

1. 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 a PDF document file.
2. In the **read-text/images** folder, right-click **Rome.pdf** and select **Reveal in File Explorer**. Then in File Explorer, open the PDF file to view it.
3. Back in the code file in Visual Studio Code, find the **GetTextRead** function, and under the existing code that prints a message to the console, add the following code:

C#

Code Copy

```
// Use Read API to read text in image
using (var imageData = File.OpenRead(imageFile))
{
    var readOp = await cvClient.ReadInStreamAsync(imageData);

    // Get the async operation ID so we can check for the results
    string operationLocation = readOp.OperationLocation;
    string operationId = operationLocation.Substring(operationLocation.Length - 36);

    // Wait for the asynchronous operation to complete
    ReadOperationResult results;
    do
    {
        Thread.Sleep(1000);
        results = await cvClient.GetReadResultAsync(Guid.Parse(operationId));
    }
    while ((results.Status == OperationStatusCodes.Running ||
            results.Status == OperationStatusCodes.NotStarted));

    // If the operation was successfuly, process the text line by line
    if (results.Status == OperationStatusCodes.Succeeded)
    {
        var textUrlFileResults = results.AnalyzeResult.ReadResults;
        foreach (ReadResult page in textUrlFileResults)
        {
            foreach (Line line in page.Lines)
            {
                Console.WriteLine(line.Text);
            }
        }
    }
}
```

Python

Code Copy

```
# Use Read API to read text in image
with open(image_file, mode="rb") as image_data:
    read_op = cv_client.read_in_stream(image_data, raw=True)

    # Get the async operation ID so we can check for the results
    operation_location = read_op.headers["Operation-Location"]
    operation_id = operation_location.split("/")[-1]

    # Wait for the asynchronous operation to complete
    while True:
        read_results = cv_client.get_read_result(operation_id)
        if read_results.status not in [OperationStatusCodes.running,
OperationStatusCodes.not_started]:
            break
        time.sleep(1)

    # If the operation was successfuly, process the text line by line
    if read_results.status == OperationStatusCodes.succeeded:
        for page in read_results.analyze_result.read_results:
            for line in page.lines:
                print(line.text)
```

1. Examine the code you added to the **GetTextRead** function. It submits a request for a read operation, and then repeatedly checks status until the operation has completed. If it was successful, the code processes the results by iterating through each page, and then through each line.
2. Save your changes and return to the integrated terminal for the **read-text** folder, and enter the following command to run the program:

C#

Code Copy

```
dotnet run
```

Python

Code Copy

```
python read-text.py
```

1. When prompted, enter **2** and observe the output, which is the text extracted from the document.

Read handwritten text

In addition to printed text, the **Read** API can extract handwritten text in English..

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

C#

Code Copy

```
dotnet run
```

Python

```
python read-text.py
```

1. When prompted, enter **3** and observe the output, which is the text extracted from the document.

More information

For more information about using the **Computer Vision** service to read text, see the Computer Vision documentation.