CDW Documentation

Image Text Detection to Translation Demo

Image Text Detection to Translation Demo

Resources to Create

Note: For the resources that are being used, you will need to allow access to your IP via the Networking tab for each resource.

Prerequisites

1. Key Vault created to store the API keys of both the Computer Vision and Language AI Resources.
2. AI Foundry Resource created.
3. Key Vault Administrator access granted to yourself to store secrets.

1. Deploying Azure Computer Vision Resource

Step 1: Sign in to Azure Portal

- Go to: https://portal.azure.com
- Sign in with your Microsoft account.

Step 2: Create a Computer Vision Resource

1. Click **“Create a resource”** from the left sidebar.
2. Search for **“Computer Vision”** and select it.
3. Click **“Create”**.

Step 3: Configure the Resource

Fill out the following fields:

- **Subscription**: Choose your Azure subscription.
- **Resource group**: Create a new one or select an existing resource group.
- **Region**: Choose a supported region (e.g., West US, East US, etc.).
- **Name**: Enter a unique name for your resource.
- **Pricing tier**: Choose the appropriate pricing tier (usually “F0” for free tier or “S1” for standard).
- Click **“Review + create”**, then **“Create”** after validation passes.

Step 4: Access Keys and Endpoint

Once deployment is complete:

1. Go to the resource.
2. Under the **“Keys and Endpoint”** section, you’ll find:
    - **Key 1 / Key 2** (API keys)
    - **Endpoint** URL

These will be used in your application to call the Computer Vision API.

2. Deploying Azure Translator Resource

NOTE: One thing to note is that you might have an issue if you have multiple translator resources in the same region. I’d recommend creating your translating resource in a separate region than existing ones.

Step 1: Create a Translator Resource

1. From the Azure Portal dashboard, click **“Create a resource”**.
2. Search for **“Translator”** and select **“Translator (Cognitive Service)”**.
3. Click **“Create”**.

Step 2: Configure the Translator Resource

Fill out the necessary fields:

- **Subscription**: Your Azure subscription.
- **Resource group**: Use the same or a new one.
- **Region**: Translator supports fewer regions (e.g., Global, North Europe).
- **Name**: Unique name for your Translator resource.
- **Pricing tier**: Select “F0” for free tier if available, otherwise choose a paid tier.
- Click **“Review + create”**, then **“Create”**.

Step 3: Access Keys and Endpoint

Once deployed:

1. Navigate to your Translator resource.
2. Go to **“Keys and Endpoint”**.
    - **Key 1 / Key 2**
    - **Endpoint** (usually something like https://<region>.api.cognitive.microsoft.com/)

Deploying and Running Python Script

Prerequisites

1. Need to have python, OpenAI, Azure CLI, and python tk installed on your local terminal.

2. Need to change parameters in Python script to match your resources.

1. Make Changes to the Below Code to Match Parameters

```
import os
import time
import requests
import tkinter as tk
from tkinter import filedialog, scrolledtext, messagebox
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient

# — CONFIGURATION —
VISION_SECRET_NAME = “YOUR-VISION-SECRET-NAME ”
TRANSLATOR_SECRET_NAME = “YOUR-TRANSLATOR-SECRET-NAME”
KEY_VAULT_URL = “https://YOUR-KEY-VAULT.vault.azure.net/”

VISION_ENDPOINT = “https://ben-vision-service.cognitiveservices.azure.com”
READ_API_URL = f“{VISION_ENDPOINT}/vision/v3.2/read/analyze”

TRANSLATOR_ENDPOINT = “https://api.cognitive.microsofttranslator.com”
TRANSLATION_TARGET_LANG = “en”
TRANSLATOR_REGION = “REGION-OF-TRANSLATOR-RESOURCE”

# — FUNCTIONS —

def get_secret_from_keyvault(vault_url, secret_name):
credential = DefaultAzureCredential()
client = SecretClient(vault_url=vault_url, credential=credential)
```

```
secret = client.get_secret(secret_name)
return secret.value


def submit_image_for_ocr(image_path, api_key):
headers = {
“Ocp-Apim-Subscription-Key”: api_key,
“Content-Type”: “application/octet-stream”
}
with open(image_path, “rb”) as image_file:
response = requests.post(READ_API_URL, headers=headers, data=image_file)
if response.status_code != 202:
raise Exception(f“OCR request failed: {response.status_code} {response.text}”)
return response.headers[“operation-location”]


def get_ocr_result(operation_url, api_key):
headers = {“Ocp-Apim-Subscription-Key”: api_key}
while True:
response = requests.get(operation_url, headers=headers)
if response.status_code != 200:
raise Exception(f“OCR result failed: {response.status_code} {response.text}”)
result = response.json()
status = result.get(“status”)
if status == “succeeded”:
return result
elif status == “failed”:
raise Exception(“OCR processing failed.”)
else:
time.sleep(1)
```

```
def extract_text_lines(ocr_json):
lines = []
read_results = ocr_json.get(“analyzeResult”, {}).get(“readResults”, [])
for page in read_results:
for line in page.get(“lines”, []):
lines.append(line.get(“text”, “”))
return lines

def translate_text(lines, translator_key, translator_endpoint, to_language=“en”):
url = f“{translator_endpoint}/translate?api-version=3.0&to={to_language}”
headers = {
'Ocp-Apim-Subscription-Key': translator_key,
'Ocp-Apim-Subscription-Region': TRANSLATOR_REGION,
'Content-type': 'application/json',
}
body = [{'Text': line} for line in lines]
response = requests.post(url, headers=headers, json=body)
response.raise_for_status()
translations = response.json()
return [item['translations'][0]['text'] for item in translations]

def run_ocr_translation(image_path):
vision_key = get_secret_from_keyvault(KEY_VAULT_URL, VISION_SECRET_NAME)
translator_key = get_secret_from_keyvault(KEY_VAULT_URL, TRANSLATOR_SECRET_NAME)

operation_url = submit_image_for_ocr(image_path, vision_key)
ocr_result = get_ocr_result(operation_url, vision_key)
```

```
lines = extract_text_lines(ocr_result)


if not lines:

return [], []


translated = translate_text(lines, translator_key, TRANSLATOR_ENDPOINT,
TRANSLATION_TARGET_LANG)

return lines, translated


# — GUI —


def select_image():

file_path = filedialog.askopenfilename(filetypes=[(“Image files”, “*.png *.jpg *.jpeg *.bmp”)])

if not file_path:

return


try:

original_text.delete(1.0, tk.END)

translated_text.delete(1.0, tk.END)

originals, translated = run_ocr_translation(file_path)

original_text.insert(tk.END, “\n”.join(originals))

translated_text.insert(tk.END, “\n”.join(translated))

except Exception as e:

messagebox.showerror(“Error”, str(e))


# — Build GUI Window —


root = tk.Tk()
```

root.title(“Azure OCR + Translator”)

frame = tk.Frame(root)

frame.pack(padx=10, pady=10)

btn = tk.Button(frame, text=“Select Image”, command=select_image)

btn.grid(row=0, column=0, columnspan=2, pady=5)

tk.Label(frame, text=“Original Text:”).grid(row=1, column=0, sticky='w')

tk.Label(frame, text=“Translated Text:”).grid(row=1, column=1, sticky='w')

original_text = scrolledtext.ScrolledText(frame, width=50, height=20)

translated_text = scrolledtext.ScrolledText(frame, width=50, height=20)

original_text.grid(row=2, column=0, padx=5)

translated_text.grid(row=2, column=1, padx=5)

root.mainloop()

Once the changes to the parameters have been made in the above code, please save the code as image-detection-translator.py

2. Create Virtual Environment and Run Python Script

Step 1: Create a Virtual Environment

Navigate to your project folder or create a new one:

bash

mkdir my_project

cd my_project

Create the virtual environment:

bash

python -m venv venv

This creates a folder named venv that contains the isolated Python environment.

Step 2: Activate the Virtual Environment

- **Windows:**

bash

.\venv\Scripts\activate

- **macOS/Linux:**

bash

source venv/bin/activate

Once activated, you should see the environment name (e.g., (venv)) in your terminal prompt.

Step 3: Install Required Packages

Install any dependencies your script needs using pip. For example:

bash

pip install requests azure-cognitiveservices-vision-computervision

You can also install from a requirements.txt file:

bash

pip install -r requirements.txt

Step 4: Run Your Python Script

Place your script (e.g., main.py) in the project directory, then run:

bash

python image-detection-translator.py

Step 5: Upload Images to Translate

Once the script has been run, select your local image files (png, jpeg, etc.) and it will pull the text from those images and translate it to English.