Classifying_Images_Using_Watson_API

August 22, 2019

Lab - Classifying Images using IBM Watson Visual Recognition in Python

Introduction

Welcome! This lab is about how to operate the Watson Visual Recognition API and OpenCV using the Python Programming Language. The advantage of using the Watson Visual Recognition API over the Graphic User Interface on the Browser that you did earlier in this course is because you can automate the training, and testing of your Visual Recognition model.

So instead of logging in to your IBM Cloud account so that you can upload a picture that you want to classify, you can upload an image to your Visual Recognition model by running a piece a piece of python code.

Click on the links to go to the following sections:

IBM Watson Package

In order to run this lab we need to import two packages.

IBM Watson: which allows access to the Watson Visual Recognition API

OpenCV: a package that will help us with image processing

The code below will install Watson Developer Cloud and OpenCV.

To run, click on the code cell below and press “shift + enter”.

NOTE - The Watson Devloper Cloud Package has been deprecated and has been replaced by the IBM Watson Package

[1]:
```
! pip install --upgrade ibm-watson opencv-python
```

```
Collecting ibm-watson
  Downloading https://files.pythonhosted.org/packages/ad/d4/4881fec032f846
37546492ec7a68ae228197feb92ae512d0728b1f2b931e/ibm-watson-3.3.0.tar.gz (276kB)
    || 286kB 23.9MB/s eta 0:00:01
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/7b/d2/a2dbf83d4553ca
6b3701d91d75e42fe50aea97acdc00652dca515749fb5d/opencv_python-4.1.0.25-cp36-cp36m
-manylinux1_x86_64.whl (26.6MB)
    || 26.6MB 31.4MB/s eta 0:00:01
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.0 in
```

/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from ibm-watson) (2.22.0)
Requirement already satisfied, skipping upgrade: python_dateutil>=2.5.3 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from ibm-watson) (2.8.0)
Collecting websocket-client==0.48.0 (from ibm-watson)
  Downloading https://files.pythonhosted.org/packages/8a/a1/72ef9aa26cfe1a75cee09fc1957e4723add9de098c15719416a1ee89386b/websocket_client-0.48.0-py2.py3-none-any.whl (198kB)
    || 204kB 30.6MB/s eta 0:00:01
Collecting ibm_cloud_sdk_core>=0.5.1 (from ibm-watson)
  Downloading https://files.pythonhosted.org/packages/f9/60/5c3e62a0c93dc66e179d79984c4d75d7bc9a866f8b320c087f2e68aec077/ibm-cloud-sdk-core-0.5.1.tar.gz
Requirement already satisfied, skipping upgrade: numpy>=1.11.3 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from opencv-python) (1.15.4)
Requirement already satisfied, skipping upgrade: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from requests<3.0,>=2.0->ibm-watson) (1.24.3)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from requests<3.0,>=2.0->ibm-watson) (3.0.4)
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from requests<3.0,>=2.0->ibm-watson) (2.8)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from requests<3.0,>=2.0->ibm-watson) (2019.6.16)
Requirement already satisfied, skipping upgrade: six>=1.5 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from python_dateutil>=2.5.3->ibm-watson) (1.12.0)
Collecting PyJWT>=1.7.1 (from ibm_cloud_sdk_core>=0.5.1->ibm-watson)
  Downloading https://files.pythonhosted.org/packages/87/8b/6a9f14b5f781697e51259d81657e6048fd31a113229cf346880bb7545565/PyJWT-1.7.1-py2.py3-none-any.whl
Building wheels for collected packages: ibm-watson, ibm-cloud-sdk-core
  Building wheel for ibm-watson (setup.py) ... done
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/0a/7f/88/ce3eb54dd29720a32ef8a071e38b47f2a2abd324061252747e
  Building wheel for ibm-cloud-sdk-core (setup.py) ... done
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/4d/42/cd/c4cdd4cc793679714e817ecfcfaa77dfbacf2af34c7d27ffd0
Successfully built ibm-watson ibm-cloud-sdk-core
Installing collected packages: websocket-client, PyJWT, ibm-cloud-sdk-core, ibm-watson, opencv-python
Successfully installed PyJWT-1.7.1 ibm-cloud-sdk-core-0.5.1 ibm-watson-3.3.0 opencv-python-4.1.0.25 websocket-client-0.48.0

What (or who) do you see in the following image?

URL:
https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/CV0101/Images/Donald_Tru
02-13_02.jpg

Plotting images in Jupyter Notebooks

Let's use a function to help us display images from a URL: The function below with the name plt_image grabs the image from the internet provided that you supply the web address of the image.

URL stands for Uniform Resource Locator, which in this case the the web address of our image.

[2]:
```
import cv2
import urllib.request
from matplotlib import pyplot as plt
from pylab import rcParams

def plt_image(image_url, size = (10,8)):

    # Downloads an image from a URL, and displays it in the notebook
    urllib.request.urlretrieve(image_url, "image.jpg") # downloads file as "image.jpg"
    image = cv2.imread("image.jpg")

    # If image is in color, then correct color coding from BGR to RGB
    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    rcParams['figure.figsize'] = size[0], size[1] #set image display size

    plt.axis("off")
    plt.imshow(image, cmap="Greys_r")
    plt.show()
```

Lets grab the image above from the internet and plot it out.

[3]:
```
image_url = 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/
↪CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13_02.jpg'
plt_image(image_url)
```

Classify images with IBM Watson API

Setting the API key for IBM Watson Visual Recognition

In order for you to use the IBM Watson Visual Recognition API, you will need the API key of the Visual Recognition instance that you have created in the previous sections.

Log into your IBM Cloud Account with the following link.

https://cloud.ibm.com

Click on Services

Under Services, click on your Watson Visual Recognition Instance

Copy the API Key and past it in the code cell below

Then press "ctrl + enter" to run the code cell.

[7]:
```
# Paste your API key for IBM Watson Visual Recognition below:
my_apikey = '**********************************************************'
```

Initialize Watson Visual Recognition

Lets create your own Watson Visual Recognition instance, it will allow you to make calls to the Watson Visual Recognition API.

[8]:
```
from ibm_watson import VisualRecognitionV3

visrec = VisualRecognitionV3(version = '2019-01-01',
                             iam_apikey = my_apikey)
```

Identifying Objects in the Image We can see that there are two persons in the picture above. But does the computer knows this?

Lets call the classify method from the Watson Image Reconition API to see what objects our Image Recognition Model can identify from this picture.

[9]:
```
import json

image_url = 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/
 ↪CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13_02.jpg'


# threshold is set to 0.6, that means only classes that has a confidence score of 0.6 or greater␣
 ↪will be shown
classes = visrec.classify(url=image_url,
                          threshold='0.6',
                          classifier_ids='default').get_result()

plt_image(image_url)
print(json.dumps(classes, indent=2))
```


```
{
  "images": [
    {
      "classifiers": [
        {
```

```
        "classifier_id": "default",
        "name": "default",
        "classes": [
          {
            "class": "official",
            "score": 0.789,
            "type_hierarchy": "/person/official"
          },
          {
            "class": "person",
            "score": 0.82
          },
          {
            "class": "beige color",
            "score": 0.782
          }
        ]
      }
    ],
    "source_url": "http://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-data/CognitiveClass/CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13
_02.jpg",
    "resolved_url": "http://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-
data/CognitiveClass/CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13_02.jpg"
  }
 ],
 "images_processed": 1,
 "custom_classes": 0
}
```


Under the field classes you should see the class person, and other classes with their corresponding confidence score. You might get other classes other than person depending on your Visual Recognition model.

Detecting Faces with Watson Visual Recognition

Since there are faces in the picture, we can use a Watson Visual Recognition API that detects faces.

The detect_faces method from the Watson Image Reconition API can help use detect faces in this picture.

[10]:
```
import json

image_url = 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/
 ↪CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13_02.jpg'

classes = visrec.detect_faces(url=image_url,
                              threshold='0.6',
                              classifier_ids='default').get_result()
```

```
plt_image(image_url)
print(json.dumps(classes, indent=2))
```


```
{
  "images": [
    {
      "faces": [
        {
          "age": {
            "min": 59,
            "max": 62,
            "score": 0.75315815
          },
          "face_location": {
            "height": 84,
            "width": 87,
            "left": 546,
            "top": 145
          },
          "gender": {
            "gender": "MALE",
            "gender_label": "male",
```

```
          "score": 0.9172811
        }
      },
      {
        "age": {
          "min": 31,
          "max": 36,
          "score": 0.5343688
        },
        "face_location": {
          "height": 77,
          "width": 68,
          "left": 220,
          "top": 125
        },
        "gender": {
          "gender": "MALE",
          "gender_label": "male",
          "score": 0.99950147
        }
      }
    ],
    "source_url": "http://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-data/CognitiveClass/CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13
_02.jpg",
    "resolved_url": "http://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-
data/CognitiveClass/CV0101/Images/Donald_Trump_Justin_Trudeau_2017-02-13_02.jpg"
  }
 ],
 "images_processed": 1
}
```


The the API call we got the age, face_location and gender of the faces in this picture.

Getting Watson Visual Recognition results as a dataframe

The problem with the classify and detect_faces method is that it gave an output that is extremely confusing to look at. The output is in a format called JSON which stands for JavaScript Object Notation, we can cleanup the presentation of our output by using the a datastructure called dataframe in the pandas library.

In the code cell below we use a function called getdf_visrec which uses a dataframe can help us easily sort the classified labels by confidence score in descending order.

[11]:
```
from pandas.io.json import json_normalize

def getdf_visrec(url, apikey = my_apikey):

    json_result = visrec.classify(url=url,
                                  threshold='0.6',
```

```
                                classifier_ids='default').get_result()

    json_classes = json_result['images'][0]['classifiers'][0]['classes']

    df = json_normalize(json_classes).sort_values('score', ascending=False).
↪reset_index(drop=True)

    return df
```

[12]:
```
url = 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/CV0101/
↪Images/76011_MAIN._AC_SS190_V1446845310_.jpg'
plt_image(url)
getdf_visrec(url)
```

[12]:

	class	score	type_hierarchy
0	food	0.877	NaN
1	cat food	0.779	/food/feed/cat food
2	feed	0.779	NaN
3	toiletry	0.772	NaN
4	pale yellow color	0.697	NaN
5	CD-R	0.681	/memory device/CD-R
6	memory device	0.681	NaN
7	beige color	0.679	NaN
8	canned food	0.627	/food/food product/canned food
9	food product	0.627	NaN

[13]:
```
url = 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/CV0101/
↪Images/2880px-Kyrenia_01-2017_img04_view_from_castle_bastion.jpg'
plt_image(url)
getdf_visrec(url)
```

[13]:

	class	score	type_hierarchy
0	nature	0.933	NaN
1	shore	0.895	NaN
2	seashore	0.869	/nature/shore/seashore
3	sea green color	0.864	NaN
4	blue color	0.858	NaN
5	natural elevation	0.794	NaN
6	slope	0.782	NaN

```
7          beach  0.612                  /nature/beach
8        seaside  0.601  /nature/shore/seashore/seaside
```

Exercises

Question 1

Watson Visual Recognition also has an API to convert image to text, the function image_to_text below converts image to text. Your job is to use the function below by providing it your api key and an image with text to extract the text in the image.

[17]:
```
# press Shift+Enter to execute
import requests

def image_to_text(apikey, url):
    request = 'https://gateway.watsonplatform.net/visual-recognition/api/v3/recognize_text?
 ↪url={url}&version=2018-03-19'.format(url=url)
    r = requests.get(request , auth=('apikey', apikey))
    return r.json()
```

[33]:
```
my_apikey = '**********************************************************'
image_to_text(my_apikey, 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/
 ↪CognitiveClass/CV0101/Images/76011_MAIN._AC_SS190_V1446845310_.jpg')
```

[33]:
```
{'images': [{'source_url': 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-
data/CognitiveClass/CV0101/Images/76011_MAIN._AC_SS190_V1446845310_.jpg',
   'resolved_url': 'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-
data/CognitiveClass/CV0101/Images/76011_MAIN._AC_SS190_V1446845310_.jpg',
   'text': 'fancy\nfeast',
   'words': [{'word': 'fancy',
     'location': {'height': 28, 'width': 91, 'left': 50, 'top': 78},
     'score': 1,
     'line_number': 0},
    {'word': 'feast',
     'location': {'height': 25, 'width': 73, 'left': 68, 'top': 103},
     'score': 0.9779,
     'line_number': 1}]}],
 'images_processed': 1}
```

Double-click here for the solution.

Question 2

Make a function that uses a pandas dataframe to convert the result into a table format.

[37]:
```
from pandas.io.json import json_normalize

def getdf_text(apikey, url):

    result = image_to_text(apikey, url)['images'][0]['words']

    df = json_normalize(result).reset_index(drop=True)
    return df
```

```
getdf_text(my_apikey,'http://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/
↪CognitiveClass/CV0101/Images/76011_MAIN._AC_SS190_V1446845310_.jpg')# Write␣
↪your code below and press Shift+Enter to execute
```

[37]:

	word	score	line_number	location.height	location.width	location.left	location.top
0	fancy	1.0000	0	28	91	50	78
1	feast	0.9779	1	25	73	68	103

Double-click here for the solution.

Thank you for completing this notebook

You can read more about Watson Visual Recognition APIs from the following link. https://cloud.ibm.com/apidocs/visual-recognition

Get IBM Watson Studio free of charge!

<p><a href="https://cocl.us/NotebooksPython101bottom"><img src="https://s3-api.us-geo.objectstorage.softl

About the Authors:

This notebook was written by Yi Yao.

Yi Yao is a Data Scientist and Software Engineer at IBM, and holds a Masters in Statistics. His research focused on Cloud Computing, Machine Learning and Computer Vision.