Watson API
Visual Recognition

IBM API 사용 가이드를 먼저 읽고 와주세요!

watson에서 제공하는 음식모델 식별자 사용 (코드)

```
import json
from ibm_watson import VisualRecognitionV3
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

authenticator = IAMAuthenticator('발급받은 API키 입력')
visual_recognition = VisualRecognitionV3(
    version='2018-03-19',
    authenticator=authenticator
)
# 우리 데이터를 얘네가 학습 시킨다고 쓸 수도 있음 이걸 방지해 주려면 아래 코드 true로 설정해 줘야함.
visual_recognition.set_default_headers({'x-watson-learning-opt-out': "true"})

visual_recognition.set_service_url('발급받은 URL 입력')

visual_recognition.set_disable_ssl_verification(True)          # 이렇게 SSL verification 꺼줘야함.
with open('C:/Users/img/pz.png, 'rb') as images_file:          # 상대경로 할 경우 상위 폴더는 파이썬 path라, 잘 모르겠다면
                                                               # 절대경로로 이미지를 설정해 주세요!
    classes = visual_recognition.classify(
        images_file=images_file,
        classifier_ids=["food"]).get_result()                  # 빨간색으로 표시한 부분이 watson에서 제공하는 식별자.
    print(json.dumps(classes, indent=2))
```

입력 이미지와 실행 결과

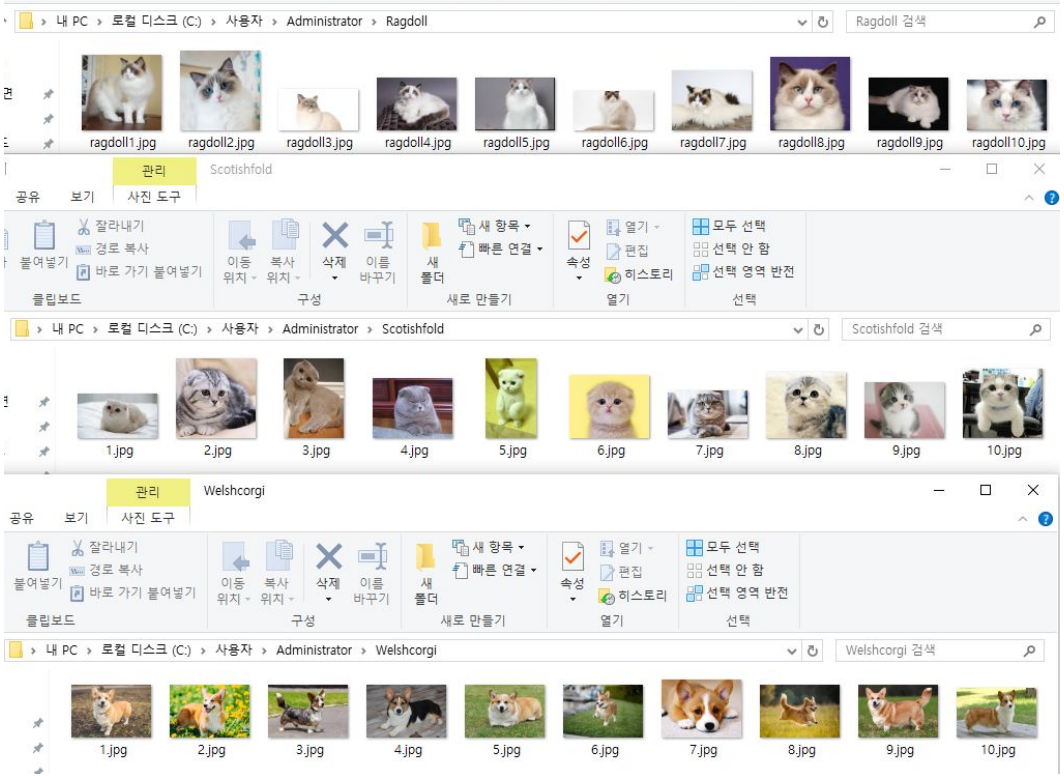
입력 이미지 pz.png


```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "food",
          "name": "food",
          "classes": [
            {
              "class": "pizza",
              "score": 0.798
            },
            {
              "class": "pepperoni pizza",
              "score": 0.5,
              "type_hierarchy": "/pizza/pepperoni pizza"
            }
          ]
        }
      ],
      "image": "pizza.png"
    }
  ],
  "images_processed": 1,
  "custom_classes": 0
}
```


실행 결과
페퍼로니 피자인것 까지 나왔습니다.

식별자를 직접 만들어서 실행!

학습 시킬 이미지를 모읍니다.



학습은 최소 몇 백장씩 이루어 져야하지만 시간 관계상 10개씩으로만 진행하였습니다.

커스텀 식별자를 쓰기 위해선 각 클래스(분류) 마다 10장씩의 이미지가 필요합니다. 부족할 경우 커스텀 식별자 상태 확인시 status = failed 와 함께 아래의 오류를 뱉습니다.

"explanation": "Cannot execute learning task. : Could not train classifier. Verify there are at least 10 positive training images for each class, and at least 10 other unique training images (inluding optional negative_examples). There is a minimum of 1 positive class. Not enough samples for training, class: scotishfold has only 5 samples",

식별자를 직접 만들어서 실행!

※ 식별자를 직접 만든다고해서 Watson의 학습 모델을 안쓴다는 것이 아닙니다.
커스텀 모델을 만들게 될 경우, 해당 custom-id의 모델을 먼저 살펴 본 후
연관되는 것이 없으면 Watson의 모델을 통해 식별을 진행하게 됩니다.

```
import json
from ibm_watson import VisualRecognitionV3
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
authenticator = IAMAuthenticator('발급받은 API키 입력')
visual_recognition = VisualRecognitionV3(
    version='2018-03-19',
    authenticator=authenticator
)
visual_recognition.set_default_headers({'x-watson-learning-opt-out': "true"})
visual_recognition.set_service_url('발급받은 URL입력')
visual_recognition.set_disable_ssl_verification(True)

with      open('./Ragdoll.zip', 'rb') as ragdoll, open('./Scotishfold.zip', 'rb') as scotishfold, open('./welshcorgi.zip', 'rb') as welshcorgi:   #사진 오픈
    model = visual_recognition.create_classifier(
        'cats',                                                                                                    #식별자 이름을 지정 -> 고양이
        positive_examples={'ragdoll': ragdoll, 'scotishfold': scotishfold},                                        #고양이 사진들을 넣습니다.
        negative_examples=welshcorgi).get_result()                                                                 #반례처럼 강아지 사진들을 넣습니다.
print(json.dumps(model, indent=2)
)
```

식별자를 직접 만든 결과


```
{
  "classifier_id": "custom_cats_494600106",  #추후 사용하기 위해 이 식별자 id는 기억해야 합니다.
  "name": "custom_cats",
  "status": "training",
  "owner": "3258dd12-c387-488c-a75d-7e9cf2391cd7",
  "created": "2020-08-19T06:53:16.818Z",
  "updated": "2020-08-19T06:53:16.818Z",
  "classes": [
    {
      "class": "scotishfold"
    },
    {
      "class": "ragdoll"
    }
  ],
  "rscnn_enabled": false,
  "core_ml_enabled": true
}
```

커스텀 식별자 사용전 상태 확인

아래의 코드를 통해 내가 만든 식별자들을 확인하고,
각 식별자의 상태를 확인 할 수 있다.

```
classifiers =
visual_recognition.list_classifiers(verbose=True).get_result()
print(json.dumps(classifiers, indent=2))
```

내가 만든 식별자로 분류를 하기 위해선 해당 식별자의 상태가
status : ready일때 사용 가능하다.

하나만 확인하고 싶다면

```
classifier = visual_recognition.get_classifier(
classifier_id='custom_cats_494600106').get_result()
print(json.dumps(classifier, indent=2))
```

를 사용하면 된다.

Response

Classifiers	A container for the list of classifiers.
classifiers * List[Classifier] Classifier	List of classifiers.
classifier_id * str	ID of a classifier identified in the image.
name * str	Name of the classifier.
owner str	Unique ID of the account who owns the classifier. Might not be returned by some requests.
status str	Training status of classifier. **Possible values:** [`ready` , `training` , `retraining` , `failed`]
core_ml_enabled bool	Whether the classifier can be downloaded as a Core ML model after the training status is `ready` .
explanation str	If classifier training has failed, this field might explain why.
created datetime	Date and time in Coordinated Universal Time (UTC) that the classifier was created.
classes List[Class] Class	Classes that define a classifier.
retrained datetime	Date and time in Coordinated Universal Time (UTC) that the classifier was updated. Might not be returned by some requests. Identical to `updated` and retained for backward compatibility.
updated datetime	Date and time in Coordinated Universal Time (UTC) that the classifier was most recently updated. The field matches either `retrained` or `created` . Might not be returned by some requests.

커스텀 식별자 사용전 상태 확인 결과

5분정도 기다리다 보면 아래와 같이 ready로 변한것을 알 수 있다. 이제 이 식별자를 통해 분류가 가능하다.


```
{
  "classifiers": [
    {
      "classifier_id": "custom_cats_494600106",
      "name": "custom_cats",
      "status": "ready",
      "owner": "3258dd12-c387-488c-a75d-7e9cf2391cd7",
      "created": "2020-08-19T06:53:16.818Z",
      "updated": "2020-08-19T06:53:16.818Z",
      "classes": [
        {
          "class": "scotishfold"
        },
        {
          "class": "ragdoll"
        }
      ],
      "rscnn_enabled": false,
      "core_ml_enabled": true
    }
  ]
}
```

직접 만든 식별자 테스트

```
import json
from ibm_watson import VisualRecognitionV3
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

authenticator = IAMAuthenticator('발급받은 API키 입력')
visual_recognition = VisualRecognitionV3(
    version='2018-03-19',
    authenticator=authenticator
)
visual_recognition.set_default_headers({'x-watson-learning-opt-out': "true"})

visual_recognition.set_service_url('발급받은 URL 입력')
visual_recognition.set_disable_ssl_verification(True)
with open('./cat_test1.jpg', 'rb') as images_file:
    classes = visual_recognition.classify(
        images_file=images_file,
        classifier_id="custom_cats_494600106").get_result() #직접 만든 식별자 사용 #classifier_id 대신 owner=[“me 사용가능”]
    print(json.dumps(classes, indent=2))


# classfier_id 를 사용할 경우 owner는 override됩니다.
# owner는 내가 만든 식별자들 모두에서 검색합니다.
```

직접 만든 식별자 테스트

테스트 이미지 cat_test1.jpg

우선 테스트 해 볼 사진은 같은 렉돌이지만,
학습 시킨 이미지에는 없는 이미지입니다.

직접 만든 식별자 테스트 결과


```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "custom_cats_494600106",
          "name": "custom_cats",
          "classes": [
            {
              "class": "ragdoll",
              "score": 0.897
            }
          ]
        }
      ],
      "image": "cat_test1.jpg"
    }
  ],
  "images_processed": 1,
  "custom_classes": 2
}
```


내가 만든 식별자의 분류 명인
custom_cats로 분석되었습니다.

또한, 89.7 % 의 확률로 레그돌이라 판별되었습니다.

images_processed 는 1개의 이미지가 처리되었음을 의미하고

custom_classes는 커스텀 식별자는 총 2개의 클래스만 있다를 의미합니다. (반례로 등록된 코기는 클래스로 치지 않습니다.)

직접 만든 식별자 테스트

이번엔 스코티쉬 폴드 이미지 중 학습시키지 않은 이미지를 넣고 돌려보겠습니다.

해당 파일명은 cat_test2.jpg입니다.

직접 만든 식별자 테스트 결과


```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "custom_cats_494600106",
          "name": "custom_cats",
          "classes": [
            {
              "class": "scotishfold",
              "score": 0.883
            }
          ]
        }
      ],
      "image": "cat_test2.jpg"
    }
  ],
  "images_processed": 1,
  "custom_classes": 2
}
```


내가 만든 식별자의 분류 명인
custom_cats로 분석되었습니다.

또한, 88.3 % 의 확률로 스코티쉬 폴드라 판별되었습니다.

직접 만든 식별자 테스트

이번엔 의도적으로 반례로 등록하였던
웰시코기로 테스트 해보겠습니다.

아마 반례로 등록되었기 때문에
custom_cat으로는 분류되지 않을 것입니다.

이미지 이름은 dog_test1.jpg입니다.

직접 만든 식별자 테스트 결과


```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "custom_cats_494600106",
          "name": "custom_cats",
          "classes": []
        }
      ],
      "image": "dog_test1.jpg"
    }
  ],
  "images_processed": 1,
  "custom_classes": 2
}
```


예상대로 아무런 고양이 classes만 있는 custom_classifier로는 분류를 할 수 없습니다.

직접 만든 식별자 테스트

이번엔 의도적으로 학습시키지 않은
클래스의 이미지를 넣어 보겠습니다.

해당 이미지의 개는 아프간 하운드입니다.

코기의 경우 반례로 등록하였기에 확실히
고양이가 아니다! 라고 할 수 있었습니다.

그럼 반례에 등록되지 않은 이 이미지는
어떻게 분류 될까요?

해당 이미지는 dog_test2.jpg입니다.

직접 만든 식별자 테스트 결과


```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "custom_cats_494600106",
          "name": "custom_cats",
          "classes": [
            {
              "class": "ragdoll",
              "score": 0.792
            }
          ]
        }
      ],
      "image": "dog_test2.jpg"
    }
  ],
  "images_processed": 1,
  "custom_classes": 2
}
```


예상과는 다르게 custom_cats -> ragdoll로 분류되었습니다.

이는 학습시킨 데이터가 현저히 부족하기 때문에 일어나는 현상입니다.