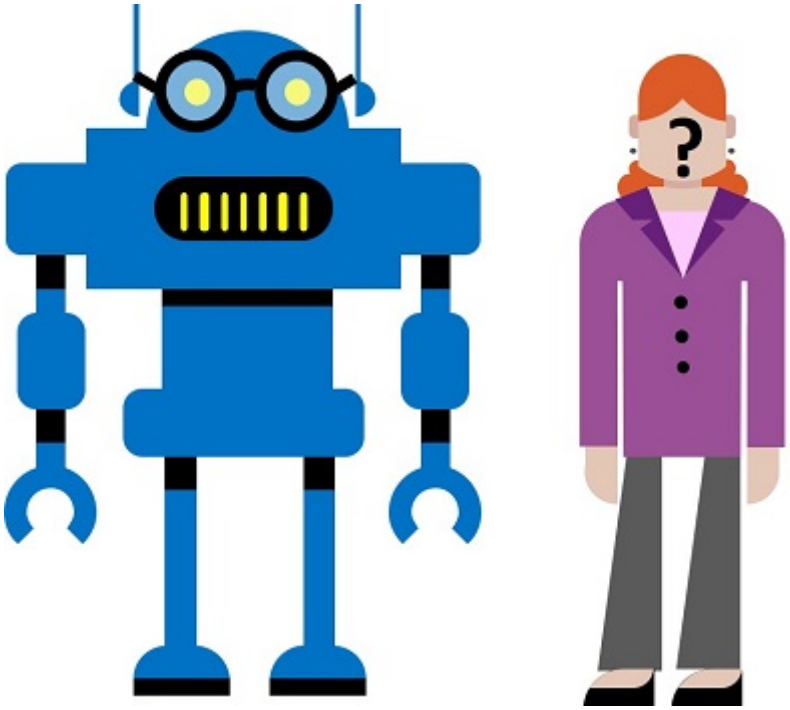
얼굴 감지 및 분석Detecting and Analyzing Faces

Computer Vision 솔루션은 종종 사람의 얼굴을 감지, 분석 또는 식별할 수 있는 인공지능(AI) 솔루션이 필요하다. 예를 들어, 소매업체 NorthWind Traders가 AI 서비스를 통해 매장을 모니터링하여 도움이 필요한 고객을 파악하고 직원들에게 도움을 지시하는 '스마트 스토어'를 구현하기로 결정했다고 가정합시다. 이를 달성하기 위한 한 가지 방법은 얼굴 감지 및 분석을 수행하는 것이다. 즉, 이미지에 얼굴이 있는지 확인하고, 얼굴의 특징을 분석하는 것이다.



Face 서비스를 통해 얼굴 감지 및 분석

Northwind Traders회사에서 원하는 스마트 시스템은 고객의 얼굴 특징을 검지하고 분석할 수 있는 도구라고 가정하자. Microsoft Azure에서는 Azure Cognitive 서비스의 일종인 **Face**를 활용하여 이것을 해결할 수 있다.

Cognitive 서비스 리소스 생성하기

Azure 구독에서 **Cognitive Services** 리소스를 생성하자.

> **알림**: 만일 이미 Cognitive 서비스 리소스를 이미 가지고 있다면 Azure 포털에 있는 **최식 리소스**에 있는 리소스를 열어서 키와 엔드포인트를 복사해서 아래 셀에 붙여 넣으면 된다. 아니면 아래 순서로 만들면 된다

1. 브라우저의 새로운 탭을 열고, Azure portal(https://portal.azure.com)을 (https://portal.azure.com)을) 입력하고 , Microsoft계정으로 로그인한다..
2. **+ 리소스 만들기** 버튼을 클릭하고, *Cognitive Services* 서비스를 찾은 다음, **Cognitive Services** 리소스를 다음과 같은 내용으로 생성한다.

   - **이름**: 유일한 이름을 입력한다*(*가능하면 영문과 숫자사용*)*.
   - **구독**: *Azure* 구독선택.
   - **위치**: 가능한 위치*(*한국 중부 추천*)*:
   - **가격책정계층**: 표준 S0
   - **리소스 그룹**: 원하는 유리한 이름*(*가능하면 영문과 숫자사용*)*.

3. 배포가 완료될 때까지 기다린다. 그런 다음 Cognitive Services 리소스로 이동하여 *개요 페이지에서 링크를 클릭하여 서비스 키를 관리한다. 클라이언트 응용 프로그램에서 Cognitive Services 리소스에 연결하려면 엔드포인트과 키가 필요하다.

Cognitive Services 리소스에 있는 키와 엔드포인트 가져오기

Cgnitive Services 리소스를 사용하기 위해서는, 클라이언트 응용프로그램에서는 엔드포인트와 인증 키가 필요합니다.client applications need its endpoint and authentication key:

1. Azure portal에서, Cgnitive Services 리소스를 선택하고 **키 및 엔트포인트** 페이지를 선택한 다음 **키1** 을 복사하여 아래의 **YOUR_COG_KEY**.를 붙여 넣는다.
2. 리소스에 있는 **엔드포인트** 를 복사해서 아래의 **YOUR_COG_ENDPOINT**.에 붙여 넣는다.
3. 셀을 선택한 다음 셀 왼쪽에있는 **셀 실행**(▷) 버튼을 클릭하여 아래 코드를 실행한다.

In [ ]:

```
cog_key = 'YOUR_COG_KEY'
cog_endpoint = 'YOUR_COG_ENDPOINT'

print('Ready to use cognitive services at {} using key {}'.format(cog_endpoint, cog_key))
```

Cognitive Services 리소스에서 Face 서비스를 이용하기 위해서는 Azure Cognitive Services Face 패키지를 설치해야 한다.

In [ ]:

```
! pip install azure-cognitiveservices-vision-face
```

이제 Cognitive 서비스 리소스와 설치된 SDK 패키지를 가지고 있으므로 가게 있는 사람 얼굴을 검지하기 위해 Face 서비스를 이용할 수 있다. 예를 확인하기 위해 아래 셀의 코드를 실행해보라.

In [ ]:

```
from azure.cognitiveservices.vision.face import FaceClient
from msrest.authentication import CognitiveServicesCredentials
from python_code import faces
import os
%matplotlib inline

# 얼굴 검지 클라이언트를 만든다.
face_client = FaceClient(cog_endpoint, CognitiveServicesCredentials(cog_key))

# 이미지를 연다.
image_path = os.path.join('data', 'face', 'store_cam2.jpg')
image_stream = open(image_path, "rb")

# 얼굴을 검지한다.
detected_faces = face_client.face.detect_with_stream(image=image_stream)

# 얼굴을 나타낸다(python_code/faces.py의 코드를 이용한다)
faces.show_faces(image_path, detected_faces)
```

검지된 얼굴들은 각각 유일한 ID가 할당되고 응용프로그램은 감지된 개별 얼굴을 구분할 수 있다.

아래 셀을 실행해서 쇼핑하고 있는 고객들의 얼굴에 대한 ID값을 확인한다.

In [ ]:

```
# 이미지를 연다.
image_path = os.path.join('data', 'face', 'store_cam3.jpg')
image_stream = open(image_path, "rb")

# 얼굴을 감지한다.
detected_faces = face_client.face.detect_with_stream(image=image_stream)

# 얼굴들을 나타낸다.(ython_code/faces.py에 있는 코드를 이용한다)
faces.show_faces(image_path, detected_faces, show_id=True)
```

얼굴의 속성들을 분석한다.

Face리소스는 단지 얼굴을 검지하는 것 외의 더 많은 일들을 수행할 수 있다. 얼굴 특징을 분석할 수 있고 나이와 감정상태에 등에 대한 설명을 제공한다. 예를들면 아래의 코드를 실행하여 쇼핑객의 얼굴 속성들을 분석해 보자.For example, run the code below to analyze the facial attributes of a shopper.

In [ ]:

```
# 이미지를 연다
image_path = os.path.join('data', 'face', 'store_cam1.jpg')
image_stream = open(image_path, "rb")

# 얼굴을 감지하고 얼굴 특성을 지정한다.
attributes = ['age', 'emotion']
detected_faces = face_client.face.detect_with_stream(image=image_stream, return_face_attributes=
attributes)

# 얼굴들과 속성값들을 나타낸다(python_code/faces.py에 있는 코드를 이용한다).
faces.show_face_attributes(image_path, detected_faces)
```

이미지에있는 고객의 감정 상태 점수를 바탕으로 쇼핑 경험에 대한 행복정도를 판정할 수 있다.

비슷한 얼굴 찾기

검지된 각 얼굴에 대한 얼굴 ID 값들은 각 얼굴들끼리 일치도를 알아보는데 사용된다.이런 ID를 이용하여 이전에 검지된 얼굴과 비교하는데 사용되고 비슷한 특징이 있는 얼굴들을 찾는데 사용된다.

예를들면 아래의 셀에 있는 코드를 실행하여 하나의 사진에 있는 쇼핑객과 다른 사진에 있는 모습을 비교하여 일치하는 얼굴을 찾는다.

In [ ]:

```
# 이미지 1에 있는 첫번째 울글의 ID를 가져온다.
image_1_path = os.path.join('data', 'face', 'store_cam3.jpg')
image_1_stream = open(image_1_path, "rb")
image_1_faces = face_client.face.detect_with_stream(image=image_1_stream)
face_1 = image_1_faces[0]

# 두번째 이미지에 있는 Face ID 값들을 가져온다.
image_2_path = os.path.join('data', 'face', 'store_cam2.jpg')
image_2_stream = open(image_2_path, "rb")
image_2_faces = face_client.face.detect_with_stream(image=image_2_stream)
image_2_face_ids = list(map(lambda face: face.face_id, image_2_faces))

# 이미지 1에 있는 얼굴과 비슷한 얼굴이 이미지 2에 있는는지 찾는다.
similar_faces = face_client.face.find_similar(face_id=face_1.face_id, face_ids=image_2_face_ids)

# 이미지 1에 있는 얼굴과 비슷한 얼굴을 이미지 2에서 찾아서 나타낸다(python_code/face.py에 있는
 코드를 사용한다).
faces.show_similar_faces(image_1_path, face_1, image_2_path, image_2_faces, similar_faces)
```

얼굴 인식(Facial Regcognition)하기

지금까지 여러분은 Face서비스로 얼굴과 얼굴의 특징을 감지할 수 있고, 서로 비슷한 두 얼굴을 식별할 수 있다는 것을 봤다. 특정 사람의 얼굴을 인식하도록 Face를 학습시키는 얼굴 인식*(Face Recognition)* 솔루션을 구현하면 한 단계 더 나아갈 수 있다. 이는 소셜 미디어 응용 프로그램에서 친구의 사진을 자동으로 태그하거나 생체 인식 확인 시스템의 일부로 얼굴 인식을 사용하는 등 다양한 시나리오에서 유용할 수 있다. 이런 작업들이 어떻게 일어나는지 살펴보기 위해 Northwind Traders 회사가 얼굴 인식서비스를 이용하여 IT 부서의 권한이 있는 직원만 보안 시스템에 액세스 하도록 하는 시스템을 만들고 싶다고 가정한다.

권한이 있는 직원을 대표할 수 있는 직원그룹을 만드는 작업부터 시작해보자.

In [ ]:

```
group_id = 'employee_group_id'
try:
    # Delete group if it already exists
    face_client.person_group.delete(group_id)
except Exception as ex:
    print(ex.message)
finally:
    face_client.person_group.create(group_id, 'employees')
    print ('Group created!')
```

직원 그룹이 존재하므로 해당 그룹에 포함시키기 원하는 직원들을 그룹에 포함시킨다.그리고 나서 각 직원들에 대한 여러종류의 사진들을 등록하여 Face서비스가 각 사람들의 독특한 특징들에 대한 것들을 학습하도록 한다. 이상적인 것은 특정사람이 다양한 포즈와 서로다른 감정을 나타내는 사진이 필요하다. 우리는 Wendell이라는 한명의 직원을 등록하고 해당 직원에 대한 여러 종류의 사진을 등록할 것이다.

In [ ]:

```
import matplotlib.pyplot as plt
from PIL import Image
import os
%matplotlib inline

# 직원(Wendell)을 그룹에 추가한다.
wendell = face_client.person_group_person.create(group_id, 'Wendell')

# Wendel/에 대한 사진들을 가져온다
folder = os.path.join('data', 'face', 'wendell')
wendell_pics = os.listdir(folder)

# 사진들을 등록한다.
i = 0
fig = plt.figure(figsize=(8, 8))
for pic in wendell_pics:
    # 직원 그룹이 있는 사람에게 각 사진들을 추가한다.
    img_path = os.path.join(folder, pic)
    img_stream = open(img_path, "rb")
    face_client.person_group_person.add_face_from_stream(group_id, wendell.person_id, img_stream
)

    # 각 이미지들을 표시한다.
    img = Image.open(img_path)
    i +=1
    a=fig.add_subplot(1,len(wendell_pics), i)
    a.axis('off')
    imgplot = plt.imshow(img)
plt.show()
```

추가된 직원과 사진들을 등록하였다면 Face를 학습하여 각 사진을 인식하도록 할 수 있다.

In [ ]:

```
face_client.person_group.train(group_id)
print('Trained!')
```

이제 학습된 모델을 사용하여 이미지에 있는 얼굴들을 구분하는데 사용할 수 있다.

In [ ]:

```
# 두번째 이미지에 있는 얼굴 ID를 가져온다.
image_path = os.path.join('data', 'face', 'employees.jpg')
image_stream = open(image_path, "rb")
image_faces = face_client.face.detect_with_stream(image=image_stream)
image_face_ids = list(map(lambda face: face.face_id, image_faces))

# 인식한 얼굴이름들을 가져온다
face_names = {}
recognized_faces = face_client.face.identify(image_face_ids, group_id)
for face in recognized_faces:
    person_name = face_client.person_group_person.get(group_id, face.candidates[0].person_id).na
me
    face_names[face.face_id] = person_name

# 인식한 알굴들을 나타낸다.
faces.show_recognized_faces(image_path, image_faces, face_names)
```

심화 학습

Face 인지 서비스에 대하여 더 많이 알고 싶다면 Face documentation (https://docs.microsoft.com/azure/cognitive-services/face/) 를 참조하라.

In [ ]:

```

```