Ativar Vision API

Para utilizar a Vision API será necessário ter um projeto do Google Cloud com faturamento ativado. Acesse o Console do Google CLoud (https://console.cloud.google.com/), selecione o projeto que deseja utilizar e garanta que o faturamento está ativado.

Acesse o link a seguir para ativar a sua API: https://console.cloud.google.com/flows/enableapi?apiid=vision.googleapis.com&redirect=https%3A%2F%2Fconsole.cloud.google.com&hl=pt-br&_ga=2.123366283.-162959472.1548538306&_gac=1.218236395.1557958120.CjwKCAjw8e7mBRBsEiwAFdYMToPTz-ZtPhoCHDYQAvD_BwE (https://console.cloud.google.com/flows/enableapi?apiid=vision.googleapis.com&redirect=https%3A%2F%2Fconsole.cloud.google.com&hl=pt-br&_ga=2.123366283.-162959472.1548538306&_gac=1.218236395.1557958120.CjwKCAjw8e7mBRBsEiwAFdYMToPTz-ZtPhoCHDYQAvD_BwE)

Criar as credenciais:

1. Acesse : https://console.cloud.google.com/apis/api/vision.googleapis.com/overview?project=aula-2904&hl=pt-br (https://console.cloud.google.com/apis/api/vision.googleapis.com/overview?project=aula-2904&hl=pt-br)
2. Clique em "Criar credenciais"
3. Selecione "Cloud Vision API"
4. Responda "Não" para caso deseje utilizar as credencias para App Engine e Compute Engine.
5. Dê um nome a sua conta de serviço e selecione o papel "Projeto", depois "Proprietário".
6. Clique em continuar e salve as credenciais em um local seguro.

Utilizando a Vision API

Para instalar a biblioteca do Google Cloud Vision vamos utilizar o gerenciador de pacotes do Python "pip".

In [2]:

```
!pip install --upgrade google-cloud-vision
```

```
Collecting google-cloud-vision
  Downloading https://files.pythonhosted.org/packages/f2/bf/112a0707a42596
1516693ac526725bc3f51db44fc3d02998da3ee2b82ef1/google_cloud_vision-0.36.0-
py2.py3-none-any.whl (383kB)
Requirement already satisfied, skipping upgrade: google-api-core[grpc]<2.
0.0dev,>=1.6.0 in c:\users\jcalv\anaconda3\lib\site-packages (from google-
cloud-vision) (1.8.2)
Requirement already satisfied, skipping upgrade: pytz in c:\users\jcalv\an
aconda3\lib\site-packages (from google-api-core[grpc]<2.0.0dev,>=1.6.0->go
ogle-cloud-vision) (2018.7)
Requirement already satisfied, skipping upgrade: google-auth<2.0dev,>=0.4.
0 in c:\users\jcalv\anaconda3\lib\site-packages (from google-api-core[grp
c]<2.0.0dev,>=1.6.0->google-cloud-vision) (1.6.3)
Requirement already satisfied, skipping upgrade: requests<3.0.0dev,>=2.18.
0 in c:\users\jcalv\anaconda3\lib\site-packages (from google-api-core[grp
c]<2.0.0dev,>=1.6.0->google-cloud-vision) (2.21.0)
Requirement already satisfied, skipping upgrade: googleapis-common-protos!
=1.5.4,<2.0dev,>=1.5.3 in c:\users\jcalv\anaconda3\lib\site-packages (from
google-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (1.5.9)
Requirement already satisfied, skipping upgrade: protobuf>=3.4.0 in c:\use
rs\jcalv\anaconda3\lib\site-packages (from google-api-core[grpc]<2.0.0dev,
>=1.6.0->google-cloud-vision) (3.7.1)
Requirement already satisfied, skipping upgrade: setuptools>=34.0.0 in
c:\users\jcalv\anaconda3\lib\site-packages (from google-api-core[grpc]<2.
0.0dev,>=1.6.0->google-cloud-vision) (40.6.3)
Requirement already satisfied, skipping upgrade: six>=1.10.0 in c:\users\j
calv\anaconda3\lib\site-packages (from google-api-core[grpc]<2.0.0dev,>=1.
6.0->google-cloud-vision) (1.12.0)
Collecting grpcio>=1.8.2; extra == "grpc" (from google-api-core[grpc]<2.0.
0dev,>=1.6.0->google-cloud-vision)
  Downloading https://files.pythonhosted.org/packages/2a/22/bd327063dd0bdf
9d8d640b3185b760707842160e69df909db3fcaab5b758/grpcio-1.20.1-cp37-cp37m-wi
n_amd64.whl (1.6MB)
Requirement already satisfied, skipping upgrade: pyasn1-modules>=0.2.1 in
c:\users\jcalv\anaconda3\lib\site-packages (from google-auth<2.0dev,>=0.4.
0->google-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (0.2.4)
Requirement already satisfied, skipping upgrade: cachetools>=2.0.0 in c:\u
sers\jcalv\anaconda3\lib\site-packages (from google-auth<2.0dev,>=0.4.0->g
oogle-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (3.1.0)
Requirement already satisfied, skipping upgrade: rsa>=3.1.4 in c:\users\jc
alv\anaconda3\lib\site-packages (from google-auth<2.0dev,>=0.4.0->google-a
pi-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (4.0)
Requirement already satisfied, skipping upgrade: urllib3<1.25,>=1.21.1 in
c:\users\jcalv\anaconda3\lib\site-packages (from requests<3.0.0dev,>=2.18.
0->google-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (1.24.1)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in
c:\users\jcalv\anaconda3\lib\site-packages (from requests<3.0.0dev,>=2.18.
0->google-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (2018.11.2
9)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in
c:\users\jcalv\anaconda3\lib\site-packages (from requests<3.0.0dev,>=2.18.
0->google-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (3.0.4)
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in c:\user
s\jcalv\anaconda3\lib\site-packages (from requests<3.0.0dev,>=2.18.0->goog
le-api-core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision) (2.8)
Requirement already satisfied, skipping upgrade: pyasn1<0.5.0,>=0.4.1 in
c:\users\jcalv\anaconda3\lib\site-packages (from pyasn1-modules>=0.2.1->go
ogle-auth<2.0dev,>=0.4.0->google-api-core[grpc]<2.0.0dev,>=1.6.0->google-c
loud-vision) (0.4.5)
Installing collected packages: google-cloud-vision, grpcio
Successfully installed google-cloud-vision-0.36.0 grpcio-1.20.1
```

Agora vamos import as bibliotecas para utilização do vision e autenticação com o Google Cloud.

In [1]:

```
from google.cloud import vision
from google.cloud.vision import types
from google.oauth2 import service_account
```

Para autenticar vamos utilziar o arquivo json que geramos com as nossas credenciais.

In [2]:

```
credentials = service_account.Credentials.from_service_account_file(r'C:\Users\jcalv\Desktop\Aula 2904-31a3b5e3c5d2.json')
```

In [3]:

```
client = vision.ImageAnnotatorClient(credentials=credentials)
```

Carregando imagens

Sempre que precisarmos analisar uma imagem, basta passar a url no "source.image_uri" do objeto "vision.types.Image()"

In [56]:

```
image = vision.types.Image()
image.source.image_uri = 'http://opais.sapo.mz/upload1/files/2018/Setembro/Semana%201/OBAMA%20TRUMP.jpg'
```

Label Detection

A api de Label Detection retorna informações características sobre a imagem.

In [38]:

```
client.label_detection(image=image)
```

Out[38]:

```
label_annotations {
  mid: "/m/035y33"
  description: "Official"
  score: 0.7639639973640442
  topicality: 0.7639639973640442
}
label_annotations {
  mid: "/m/0244x1"
  description: "Gesture"
  score: 0.7629941701889038
  topicality: 0.7629941701889038
}
label_annotations {
  mid: "/m/081pkj"
  description: "Event"
  score: 0.7534741759300232
  topicality: 0.7534741759300232
}
label_annotations {
  mid: "/m/01xyhv"
  description: "Suit"
  score: 0.7319788932800293
  topicality: 0.7319788932800293
}
label_annotations {
  mid: "/m/012t_z"
  description: "Businessperson"
  score: 0.7243381142616272
  topicality: 0.7243381142616272
}
label_annotations {
  mid: "/m/01kq3x"
  description: "White-collar worker"
  score: 0.693401575088501
  topicality: 0.693401575088501
}
label_annotations {
  mid: "/m/02w3_2"
  description: "Formal wear"
  score: 0.5713130831718445
  topicality: 0.5713130831718445
}
```

Face detection

A api de Face Detection retorna várias informações sobre rostos encontrados, como expressões e suas características.

In [111]:

```
client.face_detection(image=image)
```

Out[111]:

```
face_annotations {
  bounding_poly {
    vertices {
      x: 198
      y: 15
    }
    vertices {
      x: 354
      y: 15
    }
    vertices {
      x: 354
      y: 195
    }
    vertices {
      x: 198
      y: 195
    }
  }
  fd_bounding_poly {
    vertices {
      x: 222
      y: 61
    }
    vertices {
      x: 338
      y: 61
    }
    vertices {
      x: 338
      y: 177
    }
    vertices {
      x: 222
      y: 177
    }
  }
  landmarks {
    type: LEFT_EYE
    position {
      x: 265.35333251953125
      y: 96.77464294433594
      z: 0.0008693091222085059
    }
  }
  landmarks {
    type: RIGHT_EYE
    position {
      x: 309.30291748046875
      y: 97.77118682861328
      z: 12.826346397399902
    }
  }
  landmarks {
    type: LEFT_OF_LEFT_EYEBROW
    position {
      x: 249.83645629882812
      y: 84.84104919433594
      z: -0.2888234555721283
```

```
    }
  }
  landmarks {
    type: RIGHT_OF_LEFT_EYEBROW
    position {
      x: 280.97265625
      y: 85.85307312011719
      z: -5.599928855895996
    }
  }
  landmarks {
    type: LEFT_OF_RIGHT_EYEBROW
    position {
      x: 302.9058532714844
      y: 86.92357635498047
      z: 0.8475637435913086
    }
  }
  landmarks {
    type: RIGHT_OF_RIGHT_EYEBROW
    position {
      x: 326.10919189453125
      y: 87.86721801757812
      z: 21.702760696411133
    }
  }
  landmarks {
    type: MIDPOINT_BETWEEN_EYES
    position {
      x: 292.180908203125
      y: 96.210693359375
      z: -2.8531265258789062
    }
  }
  landmarks {
    type: NOSE_TIP
    position {
      x: 292.68548583984375
      y: 123.55957794189453
      z: -15.60588550567627
    }
  }
  landmarks {
    type: UPPER_LIP
    position {
      x: 289.7351989746094
      y: 142.49606323242188
      z: -4.661035537719727
    }
  }
  landmarks {
    type: LOWER_LIP
    position {
      x: 287.9480895996094
      y: 156.17950439453125
      z: -1.1755465269088745
    }
  }
  landmarks {
    type: MOUTH_LEFT
    position {
```

```
      x: 270.0450439453125
      y: 147.56411743164062
      z: 2.1811609268188477
    }
  }
  landmarks {
    type: MOUTH_RIGHT
    position {
      x: 303.4399108886719
      y: 149.30865478515625
      z: 12.54068660736084
    }
  }
  landmarks {
    type: MOUTH_CENTER
    position {
      x: 288.6594543457031
      y: 148.74949645996094
      z: -1.142162561416626
    }
  }
  landmarks {
    type: NOSE_BOTTOM_RIGHT
    position {
      x: 300.4217224121094
      y: 128.36927795410156
      z: 5.167067050933838
    }
  }
  landmarks {
    type: NOSE_BOTTOM_LEFT
    position {
      x: 278.6382141113281
      y: 126.92578125
      z: -1.8943065404891968
    }
  }
  landmarks {
    type: NOSE_BOTTOM_CENTER
    position {
      x: 290.4975280761719
      y: 131.61602783203125
      z: -5.063995838165283
    }
  }
  landmarks {
    type: LEFT_EYE_TOP_BOUNDARY
    position {
      x: 266.8935241699219
      y: 93.33061981201172
      z: -2.9682624340057373
    }
  }
  landmarks {
    type: LEFT_EYE_RIGHT_CORNER
    position {
      x: 276.8304138183594
      y: 97.8527603149414
      z: 3.106285572052002
    }
  }
```

```
    landmarks {
      type: LEFT_EYE_BOTTOM_BOUNDARY
      position {
        x: 266.1114501953125
        y: 99.30899047851562
        z: -0.16316844522953033
      }
    }
    landmarks {
      type: LEFT_EYE_LEFT_CORNER
      position {
        x: 255.6291046142578
        y: 96.14215850830078
        z: 1.7103313207626343
      }
    }
    landmarks {
      type: LEFT_EYE_PUPIL
      position {
        x: 265.4905090332031
        y: 96.51251220703125
        z: -1.1256279945373535
      }
    }
    landmarks {
      type: RIGHT_EYE_TOP_BOUNDARY
      position {
        x: 312.48907470703125
        y: 95.35151672363281
        z: 10.262094497680664
      }
    }
    landmarks {
      type: RIGHT_EYE_RIGHT_CORNER
      position {
        x: 319.2637023925781
        y: 98.98201751708984
        z: 20.119823455810547
      }
    }
    landmarks {
      type: RIGHT_EYE_BOTTOM_BOUNDARY
      position {
        x: 310.1251525878906
        y: 100.61991882324219
        z: 12.890228271484375
      }
    }
    landmarks {
      type: RIGHT_EYE_LEFT_CORNER
      position {
        x: 301.4379577636719
        y: 98.56497192382812
        z: 10.766583442687988
      }
    }
    landmarks {
      type: RIGHT_EYE_PUPIL
      position {
        x: 312.30804443359375
        y: 98.59532928466797
```

```
        z: 12.432888984680176
      }
    }
    landmarks {
      type: LEFT_EYEBROW_UPPER_MIDPOINT
      position {
        x: 266.3660888671875
        y: 79.52096557617188
        z: -6.074974536895752
      }
    }
    landmarks {
      type: RIGHT_EYEBROW_UPPER_MIDPOINT
      position {
        x: 315.8758850097656
        y: 81.72900390625
        z: 8.283424377441406
      }
    }
    landmarks {
      type: LEFT_EAR_TRAGION
      position {
        x: 222.1542205810547
        y: 116.85243225097656
        z: 47.16167068481445
      }
    }
    landmarks {
      type: RIGHT_EAR_TRAGION
      position {
        x: 323.01763916015625
        y: 121.55029296875
        z: 78.09284210205078
      }
    }
    landmarks {
      type: FOREHEAD_GLABELLA
      position {
        x: 292.4067687988281
        y: 85.89997100830078
        z: -4.079625129699707
      }
    }
    landmarks {
      type: CHIN_GNATHION
      position {
        x: 285.364013671875
        y: 178.6868896484375
        z: 5.906296730041504
      }
    }
    landmarks {
      type: CHIN_LEFT_GONION
      position {
        x: 228.5838623046875
        y: 148.2084197998047
        z: 31.5447940826416
      }
    }
    landmarks {
      type: CHIN_RIGHT_GONION
```

```
      position {
        x: 321.9916687011719
        y: 152.577880859375
        z: 58.57574462890625
      }
    }
    roll_angle: 2.8420138359069824
    pan_angle: 16.159584045410156
    tilt_angle: -0.9995085000991821
    detection_confidence: 0.9947745203971863
    landmarking_confidence: 0.6833332777023315
    joy_likelihood: VERY_UNLIKELY
    sorrow_likelihood: UNLIKELY
    anger_likelihood: VERY_UNLIKELY
    surprise_likelihood: VERY_UNLIKELY
    under_exposed_likelihood: VERY_UNLIKELY
    blurred_likelihood: VERY_UNLIKELY
    headwear_likelihood: VERY_UNLIKELY
  }
  face_annotations {
    bounding_poly {
      vertices {
        x: 364
      }
      vertices {
        x: 520
      }
      vertices {
        x: 520
        y: 179
      }
      vertices {
        x: 364
        y: 179
      }
    }
    fd_bounding_poly {
      vertices {
        x: 371
        y: 40
      }
      vertices {
        x: 488
        y: 40
      }
      vertices {
        x: 488
        y: 158
      }
      vertices {
        x: 371
        y: 158
      }
    }
    landmarks {
      type: LEFT_EYE
      position {
        x: 391.2872314453125
        y: 85.3271484375
        z: -0.00018834824732039124
      }
```

```
    }
    landmarks {
      type: RIGHT_EYE
      position {
        x: 415.1740417480469
        y: 78.41156005859375
        z: -40.04420471191406
      }
    }
    landmarks {
      type: LEFT_OF_LEFT_EYEBROW
      position {
        x: 385.1196594238281
        y: 75.88157653808594
        z: 16.759456634521484
      }
    }
    landmarks {
      type: RIGHT_OF_LEFT_EYEBROW
      position {
        x: 388.2620544433594
        y: 73.01530456542969
        z: -14.753695487976074
      }
    }
    landmarks {
      type: LEFT_OF_RIGHT_EYEBROW
      position {
        x: 400.775390625
        y: 72.38076782226562
        z: -34.49003982543945
      }
    }
    landmarks {
      type: RIGHT_OF_RIGHT_EYEBROW
      position {
        x: 427.28179931640625
        y: 65.6488037109375
        z: -51.51861572265625
      }
    }
    landmarks {
      type: MIDPOINT_BETWEEN_EYES
      position {
        x: 397.5003967285156
        y: 81.7005844116211
        z: -25.649641036987305
      }
    }
    landmarks {
      type: NOSE_TIP
      position {
        x: 388.85894775390625
        y: 108.03510284423828
        z: -34.8874626159668
      }
    }
    landmarks {
      type: UPPER_LIP
      position {
        x: 398.81005859375
```

```
      y: 126.05203247070312
      z: -30.556804656982422
    }
  }
  landmarks {
    type: LOWER_LIP
    position {
      x: 404.1756896972656
      y: 144.08407592773438
      z: -29.90916633605957
    }
  }
  landmarks {
    type: MOUTH_LEFT
    position {
      x: 400.9117431640625
      y: 135.35458374023438
      z: -8.204380989074707
    }
  }
  landmarks {
    type: MOUTH_RIGHT
    position {
      x: 419.9039001464844
      y: 130.02938842773438
      z: -40.09150695800781
    }
  }
  landmarks {
    type: MOUTH_CENTER
    position {
      x: 403.0116271972656
      y: 134.15481567382812
      z: -29.20517349243164
    }
  }
  landmarks {
    type: NOSE_BOTTOM_RIGHT
    position {
      x: 408.72698974609375
      y: 110.1910400390625
      z: -36.544517517089844
    }
  }
  landmarks {
    type: NOSE_BOTTOM_LEFT
    position {
      x: 396.0948181152344
      y: 114.0822982788086
      z: -14.671459197998047
    }
  }
  landmarks {
    type: NOSE_BOTTOM_CENTER
    position {
      x: 397.4366760253906
      y: 116.11595153808594
      z: -29.927568435668945
    }
  }
  landmarks {
```

```
      type: LEFT_EYE_TOP_BOUNDARY
      position {
        x: 388.0926208496094
        y: 82.35718536376953
        z: -1.4912315607070923
      }
    }
    landmarks {
      type: LEFT_EYE_RIGHT_CORNER
      position {
        x: 396.4706115722656
        y: 84.45330810546875
        z: -8.184733390808105
      }
    }
    landmarks {
      type: LEFT_EYE_BOTTOM_BOUNDARY
      position {
        x: 391.3440856933594
        y: 88.61724090576172
        z: -0.5069612860679626
      }
    }
    landmarks {
      type: LEFT_EYE_LEFT_CORNER
      position {
        x: 390.05401611328125
        y: 86.45484924316406
        z: 10.374319076538086
      }
    }
    landmarks {
      type: LEFT_EYE_PUPIL
      position {
        x: 389.92547607421875
        y: 85.59597778320312
        z: 0.03318382427096367
      }
    }
    landmarks {
      type: RIGHT_EYE_TOP_BOUNDARY
      position {
        x: 413.22052001953125
        y: 75.95417785644531
        z: -42.44948196411133
      }
    }
    landmarks {
      type: RIGHT_EYE_RIGHT_CORNER
      position {
        x: 424.71002197265625
        y: 77.66838073730469
        z: -46.61054229736328
      }
    }
    landmarks {
      type: RIGHT_EYE_BOTTOM_BOUNDARY
      position {
        x: 415.9945068359375
        y: 81.79241180419922
        z: -41.318817138671875
```

```
    }
  }
  landmarks {
    type: RIGHT_EYE_LEFT_CORNER
    position {
      x: 410.9632263183594
      y: 80.78704833984375
      z: -32.52395248413086
    }
  }
  landmarks {
    type: RIGHT_EYE_PUPIL
    position {
      x: 415.6860046386719
      y: 79.02849578857422
      z: -42.03990936279297
    }
  }
  landmarks {
    type: LEFT_EYEBROW_UPPER_MIDPOINT
    position {
      x: 383.1494445800781
      y: 68.89527893066406
      z: -0.03910737484693527
    }
  }
  landmarks {
    type: RIGHT_EYEBROW_UPPER_MIDPOINT
    position {
      x: 410.4218444824219
      y: 62.030250549316406
      z: -44.454158782958984
    }
  }
  landmarks {
    type: LEFT_EAR_TRAGION
    position {
      x: 428.88763427734375
      y: 108.75192260742188
      z: 55.55459976196289
    }
  }
  landmarks {
    type: RIGHT_EAR_TRAGION
    position {
      x: 482.27325439453125
      y: 94.60992431640625
      z: -36.51457214355469
    }
  }
  landmarks {
    type: FOREHEAD_GLABELLA
    position {
      x: 392.91998291015625
      y: 71.05912017822266
      z: -25.536909103393555
    }
  }
  landmarks {
    type: CHIN_GNATHION
    position {
```

```
        x: 415.7523498535156
        y: 167.46031188964844
        z: -27.292964935302734
      }
    }
    landmarks {
      type: CHIN_LEFT_GONION
      position {
        x: 419.2526550292969
        y: 140.65476989746094
        z: 38.61252975463867
      }
    }
    landmarks {
      type: CHIN_RIGHT_GONION
      position {
        x: 470.9612121582031
        y: 127.53667449951172
        z: -45.34117889404297
      }
    }
    roll_angle: -7.734320640563965
    pan_angle: -57.87935256958008
    tilt_angle: 4.1450042724609375
    detection_confidence: 0.8947761058807373
    landmarking_confidence: 0.6671503186225891
    joy_likelihood: VERY_UNLIKELY
    sorrow_likelihood: VERY_UNLIKELY
    anger_likelihood: VERY_UNLIKELY
    surprise_likelihood: VERY_UNLIKELY
    under_exposed_likelihood: VERY_UNLIKELY
    blurred_likelihood: VERY_UNLIKELY
    headwear_likelihood: VERY_UNLIKELY
}
```

Web detection

A api do Web Detection é semelhante a Image Search do Google, e retorna termos relativos a imagem encontrados na internet assim como sites onde a imagem foi encontrada.

In [57]:

```
client.web_detection(image=image)
```

Out[57]:

```
web_detection {
  web_entities {
    entity_id: "/m/02mjmr"
    score: 12.954000473022461
    description: "Barack Obama"
  }
  web_entities {
    entity_id: "/m/0cqt90"
    score: 10.950000762939453
    description: "Donald Trump"
  }
  web_entities {
    entity_id: "/m/060d2"
    score: 0.7042999863624573
    description: "President of the United States"
  }
  web_entities {
    entity_id: "/m/09c7w0"
    score: 0.684749960899353
    description: "United States"
  }
  web_entities {
    entity_id: "/t/26tx7xy9_tgvr"
    score: 0.5184999704360962
  }
  web_entities {
    entity_id: "/m/0zdjjz8"
    score: 0.4514999985694885
    description: "2009 Nobel Peace Prize"
  }
  web_entities {
    entity_id: "/m/0gmcgmy"
    score: 0.4301999807357788
    description: "The Obama Story"
  }
  web_entities {
    entity_id: "/m/07wbk"
    score: 0.3977999985218048
    description: "Republican Party"
  }
  web_entities {
    entity_id: "/m/0dtj5"
    score: 0.3741999864578247
    description: "United States Department of Justice"
  }
  web_entities {
    entity_id: "/m/060c4"
    score: 0.3578999936580658
    description: "President"
  }
  full_matching_images {
    url: "http://opais.sapo.mz/upload1/files/2018/Setembro/Semana%201/OBAM
A%20TRUMP.jpg"
  }
  partial_matching_images {
    url: "https://www.mikechurch.com/wp-content/uploads/2018/03/632213450
1.jpg"
  }
  partial_matching_images {
```

```
      url: "https://metro.co.uk/wp-content/uploads/2018/05/sei_14485183.jpg?
quality=90&strip=all"
    }
    partial_matching_images {
      url: "http://www.usmessageboard.com/attachments/182101/"
    }
    partial_matching_images {
      url: "https://newsonsnooze.com/wp-content/uploads/2019/04/donald-trump
-and-his-allies-in-the-senate-want-to-investigate-the-obama-administration
_5cb0bb925738e.jpeg"
    }
    partial_matching_images {
      url: "https://img.huffingtonpost.com/asset/58bb18431900003700bd6c4f.jp
g?ops=1910_1000"
    }
    partial_matching_images {
      url: "https://lh3.googleusercontent.com/CznwH5GIX-3hANu-b7MQRz9Lix2Atv
1GmWw1oSRSrGJzzMLRG5zz6ntVZOFQ5QAJVBTH2mAo=s1280-p-no-v1"
    }
    partial_matching_images {
      url: "https://yournews.com/wp-content/uploads/2018/09/632213450.jpg"
    }
    partial_matching_images {
      url: "https://i.ylilauta.org/62/97f4a180.jpg"
    }
    partial_matching_images {
      url: "https://www.tah-heetch.com/wp-content/uploads/2018/6/donald-trum
p-obama-wiretapped-trump-tower-before-election-fortune.jpg"
    }
    partial_matching_images {
      url: "https://static2.annahar.com/storage/attachments/971/22_674909.jp
g"
    }
    pages_with_matching_images {
      url: "https://sputniknews.com/us/201810271069273782-obama-trump-lies/"
      page_title: "Twitter Reminds <b>Obama of</b> Own &#39;Lies&#39; as He
Accuses Trump <b>of</b> ..."
      partial_matching_images {
        url: "https://cdn2.img.sputniknews.com/images/105130/77/1051307788.j
pg"
      }
    }
    pages_with_matching_images {
      url: "https://metro.co.uk/2018/07/17/barack-obama-appears-condemn-sham
eless-donald-trump-7727296/"
      page_title: "<b>Barack Obama</b> appears to condemn &#39;shameless&#3
9; Donald Trump ..."
      partial_matching_images {
        url: "https://metro.co.uk/wp-content/uploads/2018/05/sei_14485183.jp
g?quality=90&strip=all"
      }
      partial_matching_images {
        url: "https://metro.co.uk/wp-content/uploads/2018/05/sei_14485183.jp
g?quality=90&strip=all&crop=0px%2C73px%2C2500px%2C1314px&resize=1200%2C63
0"
      }
    }
    pages_with_matching_images {
      url: "https://wrko.iheart.com/content/2018-07-23-did-clapper-admit-oba
ma-was-behind-the-russia-witch-hunt/"
      page_title: "Did Clapper Admit <b>Obama</b> Was Behind <b>The</b> Russ
```

```
ia Witch Hunt ..."
    partial_matching_images {
      url: "https://i.iheart.com/v3/re/new_assets/5b55ef42a8ce1c70dea56a5
a?ops=max(650,0),quality(80)"
    }
  }
  pages_with_matching_images {
    url: "https://www.latimes.com/opinion/opinion-la/la-ol-trump-afghanist
an-20170822-story.html"
    page_title: "No, Mr. <b>President</b>, <b>Obama</b> wasn&#39;t a &#39;
nation-builder&#39; in Afghanistan ..."
    partial_matching_images {
      url: "https://www.latimes.com/resizer/yitf1IJZ3mwtDN9bAj9o1jDCpYc=/1
200x0/arc-anglerfish-arc2-prod-tronc.s3.amazonaws.com/public/TFQNLX6ZJFB4R
HBK544KJIZN2M.jpg"
    }
  }
  pages_with_matching_images {
    url: "https://sputniknews.com/europe/201811191069938886-obama-nobel-pe
ace-prize-revoked/"
    page_title: "<b>Barack Obama</b> Might Lose His Nobel Prize &amp; See
Donald Trump Get It"
    partial_matching_images {
      url: "https://cdn2.img.sputniknews.com/images/105130/77/1051307788.j
pg"
    }
  }
  pages_with_matching_images {
    url: "https://twitter.com/thepresobama/status/1019073344824627200?lang
=en"
    page_title: "<b>Barack Obama</b>     Commentary on Twitter: &quot;Team
Treason.\342\200\246 &quot;"
    partial_matching_images {
      url: "https://pbs.twimg.com/media/DiR5ypnV4AAI6jM.jpg"
    }
  }
  pages_with_matching_images {
    url: "https://sputniknews.com/us/201805081064265589-barack-obama-respo
nse-iran-nuclear/"
    page_title: "&#39;Today&#39;s Announcement Is So Misguided&#39;: <b>Ob
ama</b> Slams Trump&#39;s ..."
    partial_matching_images {
      url: "https://cdn2.img.sputniknews.com/images/105130/77/1051307788.j
pg"
    }
  }
  pages_with_matching_images {
    url: "https://www.amny.com/news/politics/trump-alleges-obama-tapped-hi
s-phones-during-campaign-1.13208426"
    page_title: "Trump alleges <b>Obama</b> tapped his phones during campa
ign | am ..."
    partial_matching_images {
      url: "https://cdn.newsday.com/polopoly_fs/1.13208437.1488633974!/htt
pImage/image.jpg_gen/derivatives/landscape_1280/image.jpg"
    }
    partial_matching_images {
      url: "https://cdn.newsday.com/polopoly_fs/1.13208437.1488633974!/htt
pImage/image.jpg_gen/derivatives/landscape_768/image.jpg"
    }
  }
  pages_with_matching_images {
```

```
    url: "https://sputniknews.com/viral/201804181063680020-obama-artificia
l-intelligence-trump-dipshit/"
    page_title: "AI Software FakeApp Forces <b>Obama</b> to Call Trump &#3
9;Total Dipsh*t ..."
    partial_matching_images {
      url: "https://cdn2.img.sputniknews.com/images/105130/77/1051307788.j
pg"
    }
  }
  pages_with_matching_images {
    url: "https://www.wattpad.com/stories/obama/hot"
    page_title: "<b>obama</b> Stories - Wattpad"
    partial_matching_images {
      url: "https://a.wattpad.com/cover/186080059-256-k479259.jpg"
    }
  }
  visually_similar_images {
    url: "https://images.haarets.co.il/image/upload/w_1135,h_853,x_0,y_0,c
_crop,g_north_west/w_281,h_211,q_auto,c_fill,f_auto/fl_any_format.preserve
_transparency.progressive:none/v1516565949/1.5749835.928366706.jpg"
  }
  visually_similar_images {
    url: "https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/get
tyimages-632200060-1524489352.jpg?crop=0.759xw:0.745xh;0.122xw,0.0357xh&re
size=480:*"
  }
  visually_similar_images {
    url: "https://cdn.cnn.com/cnnnext/dam/assets/170306095528-obama-and-tr
ump-at-inauguration-01-2017-large-169.jpg"
  }
  visually_similar_images {
    url: "https://cdn.cnn.com/cnnnext/dam/assets/170902104305-bush-obama-i
nauguration-2009-large-169.jpg"
  }
  visually_similar_images {
    url: "https://cdn.mos.cms.futurecdn.net/RhmCatbrHXuKHNCvMFkDNF-320-80.
jpg"
  }
  visually_similar_images {
    url: "https://s.newsweek.com/sites/www.newsweek.com/files/styles/embed
_tablet/public/2018/02/06/020618melaniatwitter.jpg"
  }
  visually_similar_images {
    url: "https://s.newsweek.com/sites/www.newsweek.com/files/styles/full/
public/2017/11/21/1121bidenobama.jpg"
  }
  visually_similar_images {
    url: "https://static.politifact.com/politifact/photos/Putin_and_Obama.
jpg"
  }
  best_guess_labels {
    label: "barack obama"
    language_code: "en"
  }
}
```


Text Annotations

A api de text annotation é capaz de ler texto em imagens assim como a sua posição.

In [142]:

```
image.source.image_uri = 'https://i.pinimg.com/originals/0c/98/84/0c98845764d76d0874f31
e89f5460464.png'
```

In [143]:

```
response = client.text_detection(image=image)
```

In [144]:

```
response.text_annotations
```

Out[144]:

```
[locale: "en"
description: "I WOULD LIKE\nTo DIE\nON MIA RS\nJUST NOT ON IMPACT\n5\nELON
MUSK\n"
bounding_poly {
  vertices {
    x: 373
    y: 580
  }
  vertices {
    x: 2300
    y: 580
  }
  vertices {
    x: 2300
    y: 2450
  }
  vertices {
    x: 373
    y: 2450
  }
}
, description: "I"
bounding_poly {
  vertices {
    x: 378
    y: 585
  }
  vertices {
    x: 451
    y: 585
  }
  vertices {
    x: 451
    y: 761
  }
  vertices {
    x: 378
    y: 761
  }
}
, description: "WOULD"
bounding_poly {
  vertices {
    x: 522
    y: 580
  }
  vertices {
    x: 1522
    y: 580
  }
  vertices {
    x: 1522
    y: 766
  }
  vertices {
    x: 522
    y: 766
  }
}
```

```
  , description: "LIKE"
  bounding_poly {
    vertices {
      x: 1596
      y: 585
    }
    vertices {
      x: 2137
      y: 585
    }
    vertices {
      x: 2137
      y: 761
    }
    vertices {
      x: 1596
      y: 761
    }
  }
  , description: "To"
  bounding_poly {
    vertices {
      x: 601
      y: 976
    }
    vertices {
      x: 956
      y: 975
    }
    vertices {
      x: 956
      y: 1138
    }
    vertices {
      x: 601
      y: 1139
    }
  }
  , description: "DIE"
  bounding_poly {
    vertices {
      x: 1031
      y: 902
    }
    vertices {
      x: 1819
      y: 901
    }
    vertices {
      x: 1819
      y: 1222
    }
    vertices {
      x: 1032
      y: 1223
    }
  }
  , description: "ON"
  bounding_poly {
    vertices {
      x: 373
```

```
      y: 1399
    }
    vertices {
      x: 676
      y: 1386
    }
    vertices {
      x: 683
      y: 1542
    }
    vertices {
      x: 380
      y: 1555
    }
  }
  , description: "MIA"
  bounding_poly {
    vertices {
      x: 753
      y: 1322
    }
    vertices {
      x: 1480
      y: 1290
    }
    vertices {
      x: 1495
      y: 1624
    }
    vertices {
      x: 768
      y: 1656
    }
  }
  , description: "RS"
  bounding_poly {
    vertices {
      x: 1533
      y: 1318
    }
    vertices {
      x: 2045
      y: 1296
    }
    vertices {
      x: 2059
      y: 1628
    }
    vertices {
      x: 1548
      y: 1651
    }
  }
  , description: "JUST"
  bounding_poly {
    vertices {
      x: 448
      y: 1798
    }
    vertices {
      x: 783
```

```
      y: 1798
    }
    vertices {
      x: 783
      y: 1924
    }
    vertices {
      x: 448
      y: 1924
    }
  }
  , description: "NOT"
  bounding_poly {
    vertices {
      x: 818
      y: 1798
    }
    vertices {
      x: 1153
      y: 1798
    }
    vertices {
      x: 1153
      y: 1904
    }
    vertices {
      x: 818
      y: 1904
    }
  }
  , description: "ON"
  bounding_poly {
    vertices {
      x: 1188
      y: 1801
    }
    vertices {
      x: 1417
      y: 1801
    }
    vertices {
      x: 1417
      y: 1903
    }
    vertices {
      x: 1188
      y: 1903
    }
  }
  , description: "IMPACT"
  bounding_poly {
    vertices {
      x: 1462
      y: 1798
    }
    vertices {
      x: 2027
      y: 1798
    }
    vertices {
      x: 2027
```

```
      y: 1904
    }
    vertices {
      x: 1462
      y: 1904
    }
  }
  , description: "5"
  bounding_poly {
    vertices {
      x: 2129
      y: 1913
    }
    vertices {
      x: 2284
      y: 1899
    }
    vertices {
      x: 2299
      y: 2072
    }
    vertices {
      x: 2144
      y: 2085
    }
  }
  , description: "ELON"
  bounding_poly {
    vertices {
      x: 959
      y: 2365
    }
    vertices {
      x: 1254
      y: 2366
    }
    vertices {
      x: 1254
      y: 2448
    }
    vertices {
      x: 959
      y: 2447
    }
  }
  , description: "MUSK"
  bounding_poly {
    vertices {
      x: 1288
      y: 2363
    }
    vertices {
      x: 1610
      y: 2364
    }
    vertices {
      x: 1610
      y: 2445
    }
    vertices {
      x: 1288
```

```
    y: 2444
  }
}
]
```

In [153]:

```
print(response.text_annotations[0].description)
```

```
I WOULD LIKE
To DIE
ON MIA RS
JUST NOT ON IMPACT
5
ELON MUSK
```

Integrando com Twitter

Agora que temos a integração com a Google Cloud Vision api feita podemos combinar o poder dessa aplicação com a streaming do twitter, fazendo uma análise inteligente sobre cada imagem encontrada para um determinado tema.

Mais informaçõs sobre como fazer a integração com o twitter no repo:
https://github.com/jcalvesoliveira/twitter-api (https://github.com/jcalvesoliveira/twitter-api)

In [4]:

```
import tweepy as tw
from textblob import TextBlob
```

In [5]:

```
consumer_key= 'key'
consumer_secret= 'secret'
access_token= 'token'
access_token_secret= 'token_secret'
```

In [6]:

```
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

Vamos criar algumas listas para armazenar os valores encontrados relativos aos tweets.

In [32]:

```
label = []
joy = []
anger = []
web = []
text = []
url = []
sentiment = []
```

Com algumas alterações podemos fazer várias chamadas de api para cada vez que achamos um tweet com uma foto associada.

In [37]:

```
class MyStreamListener(tw.StreamListener):
    def on_status(self, status):
        print(status.text)

        print('------------------------------')
        print('Idioma:')
        try:
            print(TextBlob(status.text).detect_language())
        except:
            print('Não foi possível detectar o idioma!')

        print('------------------------------')
        print('Tradução:')
        try:
            print(TextBlob(status.text).translate(to='pt'))
        except:
            print('Não foi possível traduzir o tweet!')

        print('Sentimento:')
        print('------------------------------')
        print(TextBlob(status.text).sentiment)









        print('------------------------------')
        print('Foto:')
        if 'media' in status.entities:
            print(status.entities['media'][0]['media_url'])


            image = vision.types.Image()
            image.source.image_uri = status.entities['media'][0]['media_url']

            label.append(client.label_detection(image=image).label_annotations[0].descr
iption)

            joy_tweet = []
            anger_tweet = []
            face_detection = client.face_detection(image=image)
            for face in face_detection.face_annotations:
                joy_tweet.append(face.joy_likelihood)
                anger_tweet.append(face.anger_likelihood)
            joy.append(joy_tweet)
            anger.append(anger_tweet)

            web.append(client.web_detection(image=image).web_detection.web_entities[0].
description)

            text.append(client.text_detection(image=image).text_annotations[0].descript
ion)

            url.append(status.entities['media'][0]['media_url'])

            sentiment.append(TextBlob(status.text).sentiment)

        else:
```

```
            print('Tweet não possui foto associada')
        print('\n')
        print('\n')
```

In [38]:

```
myStream = tw.Stream(auth = auth, listener=MyStreamListener())
```

Agora podemos rodar o nosso código e armazenar os resultados para um pesquisa.

In [ ]:

```
myStream.filter(track=['trump'])
```

Analisando os resultados

Agora podemos montar um DataFrame com os resultados obtidos e analisar as fotos que capturamos.

In [41]:

```
import pandas as pd
```

In [48]:

```
tweets_photos = pd.DataFrame()

tweets_photos['label'] = label
tweets_photos['joy'] = joy
tweets_photos['anger'] = anger
tweets_photos['web'] = web
tweets_photos['text'] = text
tweets_photos['url'] = url
tweets_photos['sentiment'] = sentiment
```

In [50]:

```
tweets_photos
```

Out[50]:

	label	joy	anger	web	text	
0	T-shirt	[5]	[1]	Ilhan Abdullahi Omar	IIHAN OMAR CALLS PRESIDENT\nDONALD TRUMP A DIC...	http://pbs.twimg.com/ext_tw_
1	Internet meme	[5]	[1]	Ilhan Abdullahi Omar	IIHAN OMAR CALLS PRESIDENT\nDONALD TRUMP A DIC...	http://pbs.twimg.com/ext_tw_
2	Internet meme	[]	[]	Iran	RIB\n	http://pbs.twimg.com/media/D6
3	Barechested	[1]	[1]	Lay	Donald J. Trump\n@realDonaldTrump\nIf Iran wan...	http://pbs.twimg.com/media/D
4	Text	[]	[]	Iran	RIB\n	http://pbs.twimg.com/media/D6