PII redaction: Passport

Redaction architecture

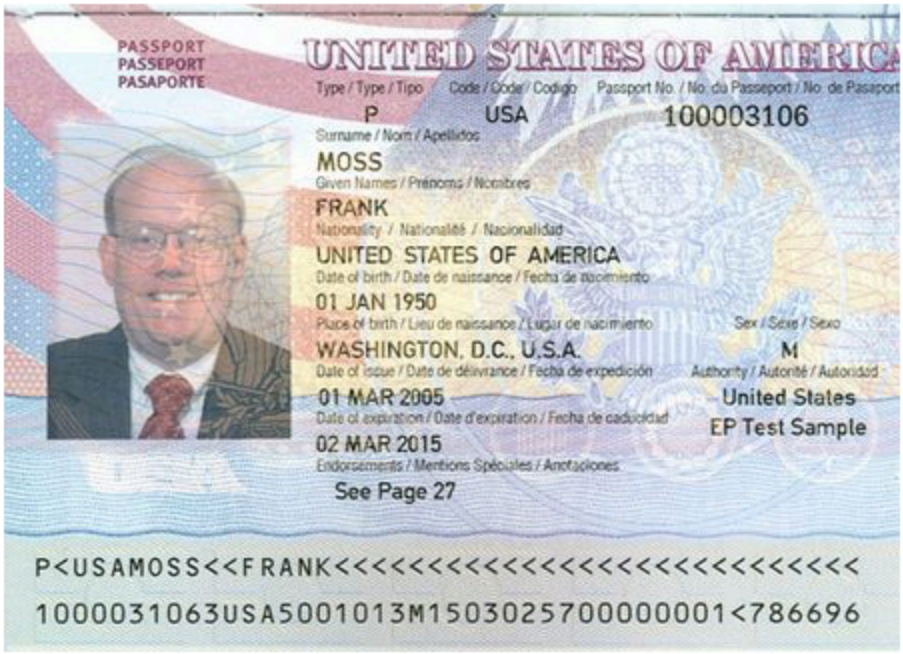
In this example, we will use the Notebooks feature of Amazon SageMaker to create an interactive notebook with Python code. These notebooks are just one part of Amazon SageMaker, a fully-managed service that covers the entire machine learning workflow to label and prepare your data, choose an algorithm, train the algorithm, tune and optimize it for deployment, make predictions, and take action. In this example though, for the actual machine learning and prediction, we will be using Amazon Rekognition to extract text from the images and Amazon Comprehend to help us to identify and detect the PII. All of our image files will be read from and written to a bucket in Amazon Simple Storage Service (Amazon S3), an object storage service that offers industry-leading scalability, data availability, security, and performance. Even though this demo uses a jupyter notebook, you can write this python code in lambda and trigger this lambda when an object is uploaded in S3. This can help with automation. This example is plainly for demo and understanding.

In [133…

```
#Define the S3 bucket and object for the   image we want to analyze.  Also define t
bucket='<your bucket name>'
#object='ImagePIIRedaction/washington-drivers-license.jpg'
#object='ImagePIIRedaction/wa-license.png'
#object='ImagePIIRedaction/sampledriverslicense.jpg'
object='ImagePIIRedaction/samplepassport-1.png'
redacted_box_color='red'
dpi = 72
pii_detection_threshold = 0.00


#If the image is in DICOM format, convert it to PNG
if (object.split(".")[-1:][0] == "dcm"):
    ! aws s3 cp s3://{bucket}/{object} .
    ! convert -format png {object.split("/")[-1:][0]} {object.split("/")[-1:][0]}.p
    ! aws s3 cp {object.split("/")[-1:][0]}.png s3://{bucket}/{object}.png
    object=object+'.png'
    print(object)

#Import all of the required libraries
%matplotlib inline
import boto3
import json
import io
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from PIL import Image
import numpy as np
```

```
import matplotlib as mpl
from imageio import imread

import base64
#import cStringIO



#Implement AWS Services
rekognition=boto3.client('rekognition')
comprehend = boto3.client(service_name='comprehend')
s3=boto3.resource('s3')

#Download the image from S3 and hold it in memory
img_bucket = s3.Bucket(bucket)
img_object = img_bucket.Object(object)
xray = io.BytesIO()
img_object.download_fileobj(xray)
img = np.array(Image.open(xray), dtype=np.uint8)
print(img.shape)
#Set the image color map to grayscale, turn off axis grapiing, and display the imag
height, width,channel = img.shape
# What size does the figure need to be in inches to fit the image?
figsize = width / float(dpi), height / float(dpi)
# Create a figure of the right size with one axes that takes up the full figure
fig = plt.figure(figsize=figsize)
ax = fig.add_axes([0, 0, 1, 1])
# Hide spines, ticks, etc.
ax.axis('off')
# Display the image.
ax.imshow(img, cmap='gray')
plt.show()
```

```
(640, 451, 4)
```



In [134…

```
#Use Amazon Rekognition to detect all of the text in the   image
#response=rekognition.detect_text(Image={'S3Object':{'Bucket':bucket,'Name':object}
response=rekognition.detect_text(Image={'Bytes':xray.getvalue()})
textDetections=response['TextDetections']
print ('Aggregating detected text...')
textblock=""
offsetarray=[]
totallength=0

#The various text detections are returned in a JSON object.  Aggregate the text int
#keep track of the offsets.  This will allow us to make a single call to Amazon Com
#pii detection and minimize our Comprehend   service charges.
for text in textDetections:
    if text['Type'] == "LINE":
            offsetarray.append(totallength)
            totallength+=len(text['DetectedText'])+1
```

```
            textblock=textblock+text['DetectedText']+" "
            print ("adding '"+text['DetectedText']+"', length: "+str(len(text['Dete
offsetarray.append(totallength)
totaloffsets=len(offsetarray)
```

```
Aggregating detected text...
adding 'PASSPORT UNITE', length: 14, offsetarray: [0]
adding 'PASSEDORT STATES OFF AMERIC', length: 27, offsetarray: [0, 15]
adding 'DASAPORTE Type Code /ode Codigo Passport No No Paseort No', length: 57, offs
etarray: [0, 15, 43]
adding 'du', length: 2, offsetarray: [0, 15, 43, 101]
adding 'die', length: 3, offsetarray: [0, 15, 43, 101, 104]
adding 'P USA 100003106', length: 15, offsetarray: [0, 15, 43, 101, 104, 108]
adding 'Surname Non Aellidiot', length: 21, offsetarray: [0, 15, 43, 101, 104, 108,
124]
adding 'MOSS', length: 4, offsetarray: [0, 15, 43, 101, 104, 108, 124, 146]
adding 'Grvent Names/ Prenoen Ncotre', length: 28, offsetarray: [0, 15, 43, 101, 10
4, 108, 124, 146, 151]
adding 'FRANK', length: 5, offsetarray: [0, 15, 43, 101, 104, 108, 124, 146, 151, 18
0]
adding 'Nationniy Nationalos Naciorualidid', length: 34, offsetarray: [0, 15, 43, 10
1, 104, 108, 124, 146, 151, 180, 186]
adding 'UNITED STATES AMERICA', length: 21, offsetarray: [0, 15, 43, 101, 104, 108,
124, 146, 151, 180, 186, 221]
adding 'OF', length: 2, offsetarray: [0, 15, 43, 101, 104, 108, 124, 146, 151, 180,
186, 221, 243]
adding 'Date ol birth/ Date de Fech e ngomnierto', length: 40, offsetarray: [0, 15,
43, 101, 104, 108, 124, 146, 151, 180, 186, 221, 243, 246]
adding '01 JAN 1950', length: 11, offsetarray: [0, 15, 43, 101, 104, 108, 124, 146,
151, 180, 186, 221, 243, 246, 287]
adding 'PLce brth/ de', length: 13, offsetarray: [0, 15, 43, 101, 104, 108, 124, 14
6, 151, 180, 186, 221, 243, 246, 287, 299]
```

In [135…

```
#Call Amazon Comprehend  and pass it the aggregated text from our  image.
pii_boxes_list=[]
piilist=comprehend.detect_pii_entities(Text = textblock, LanguageCode='en')

#Amazon Comprehend  will return a JSON object that contains all of the PII detected
#offset values that describe where the PII begins and ends.  We can use this to det
#detected by Amazon Rekognition should be redacted.  The 'pii_boxes_list' list is c
#bounding boxes that potentially contain PII.
print ('Finding PII text...')
not_redacted=0
for pii in piilist['Entities']:
    print(pii['Type'])
    if pii['Score'] > pii_detection_threshold:
        for i in range(0,totaloffsets-1):
            if offsetarray[i] <= pii['BeginOffset'] < offsetarray[i+1]:
                if textDetections[i]['Geometry']['BoundingBox'] not in pii_boxes_li
                    print ("detected as type '"+pii['Type']+"' and will be redacted
                    pii_boxes_list.append(textDetections[i]['Geometry']['BoundingBo
    else:
        print (" was detected as type '"+pii['Type']+"', but did not meet the confi
        not_redacted+=1
pii_boxes_list.append(textDetections[5]['Geometry']['BoundingBox'])
print(textDetections[9])
```

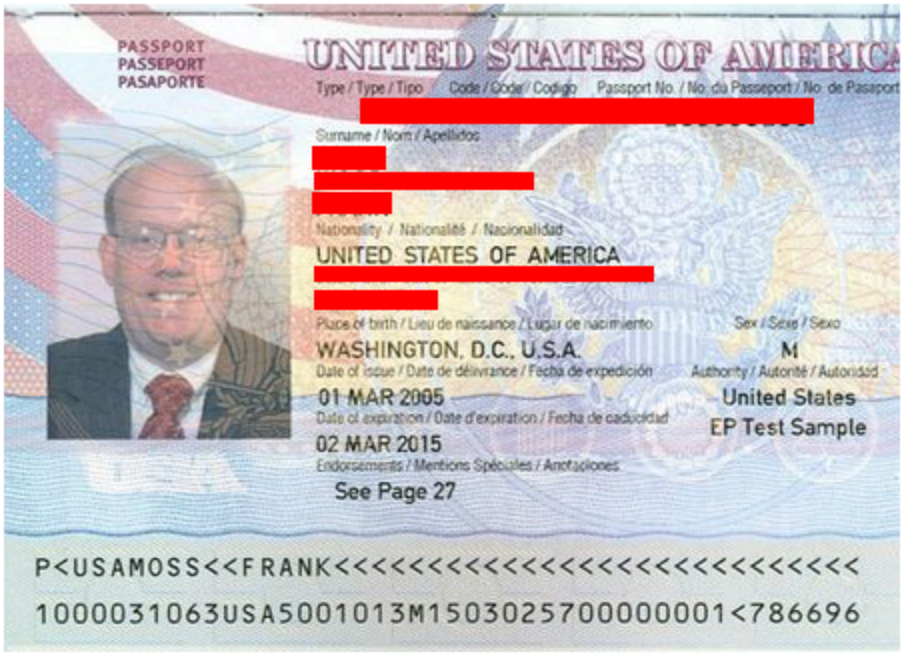
```
print ("Found", len(pii_boxes_list), "text boxes to redact.")
pii_boxes_list.append(textDetections[9]['Geometry']['BoundingBox'])
print (not_redacted, "additional text boxes were detected, but did not meet the con
```

```
Finding PII text...
NAME
detected as type 'NAME' and will be redacted.
NAME
detected as type 'NAME' and will be redacted.
NAME
detected as type 'NAME' and will be redacted.
NAME
detected as type 'NAME' and will be redacted.
DATE_TIME
detected as type 'DATE_TIME' and will be redacted.
{'DetectedText': 'FRANK', 'Type': 'LINE', 'Id': 9, 'Confidence': 99.55354309082031,
'Geometry': {'BoundingBox': {'Width': 0.08869179338216782, 'Height': 0.0171875003725
2903, 'Left': 0.3414634168148041, 'Top': 0.3515625}, 'Polygon': [{'X': 0.34146341681
48041, 'Y': 0.3515625}, {'X': 0.4301552176475525, 'Y': 0.3515625}, {'X': 0.430155217
6475525, 'Y': 0.3687500059604645}, {'X': 0.3414634168148041, 'Y': 0.368750005960464
5}]}}
Found 6 text boxes to redact.
0 additional text boxes were detected, but did not meet the confidence score thresho
ld.
```

In [136…

```
#Now this list of bounding boxes will be used to draw red boxes over the PII text.
height, width, channel = img.shape
# What size does the figure need to be in inches to fit the image?
figsize = width / float(dpi), height / float(dpi)
# Create a figure of the right size with one axes that takes up the full figure
fig = plt.figure(figsize=figsize)
ax = fig.add_axes([0, 0, 1, 1])
ax.imshow(img)
plt.imshow(img, cmap='gray')
for box in pii_boxes_list:
    #The bounding boxes are described as a ratio of the overall image dimensions, s
    #by the total image dimensions to get the exact pixel values for each dimension
    x = img.shape[1] * box['Left']
    y = img.shape[0] * box['Top']
    width = img.shape[1] * box['Width']
    height = img.shape[0] * box['Height']
    rect = patches.Rectangle((x,y),width,height,linewidth=0,edgecolor=redacted_box_
    ax.add_patch(rect)
#Ensure that no axis or whitespaces is printed in the image file we want to save.
plt.axis('off')
plt.gca().xaxis.set_major_locator(plt.NullLocator())
plt.gca().yaxis.set_major_locator(plt.NullLocator())

#Save redacted   image to the same Amazon S3 bucket, in PNG format, with 'de-id-' i
#filename.
img_data = io.BytesIO()
plt.savefig(img_data, bbox_inches='tight', pad_inches=0, format='png')
img_data.seek(0)
#Uncomment the line below to write the redacted image to S3
#img_bucket.put_object(Body=img_data, ContentType='image/png', Key='de-id-'+object)
```

Out[136… 0



In [ ]:

In [ ]: