ML Services con AWS

Jorge Alejandro Kamlofsky[1,2]

[1]GIA: Grupo de Investigación en Inteligencia Artificial. Universidad Tecnológica Nacional, Facultad Regional Haedo.

[2]CAETI: Centro de Altos Estudios en Tecnología Informática. Universidad Abierta Interamericana, Facultad de Tecnología Informática.

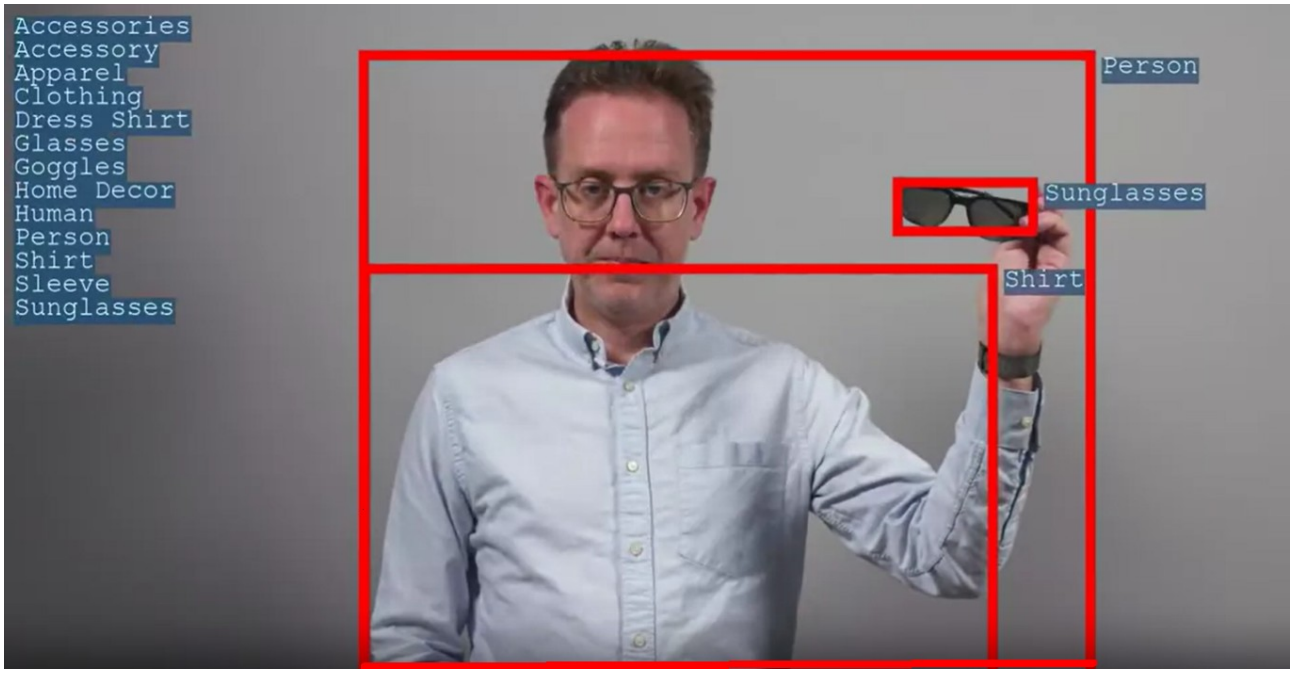
1. Rekognition:

Es un servicio de reconocimiento de objetos en imágenes, Permite reconocer objetos o etiquetas dentro de imágenes. Funciona como un servicio desde la consola de AWS.

1.1 Operaciones:

Principales operaciones en Rekognition:

- **CompareFaces:** Dentro de una colección de caras, identifica aquella que corresponde con la detectada
- **DetectFaces:** genero, rango etario, sonrisa, ojos y/o boca abierta, etc.
- **DetectLabels:** En una imagen le pone etiqueta a objetos reconocidos, presenta clase, con cierta confianza (p/ej.: clase: construcción. Label: puente. Confidence: 0.998).
- DetectModerationLabels
- DetectProtectiveEquipments
- **DetectText:** Identifica textos.
- RecognizeCelebrities



Aunque en general Rekognition usa modelos pre-entrenados, es posible agregar imágenes y re-entrenar modelos.

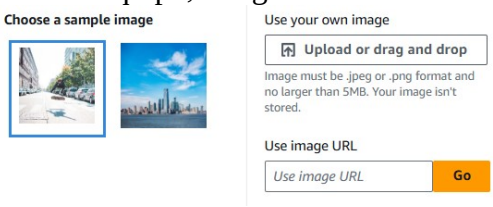
1.2. Ejemplo de Uso en consola

Diríjase a Rekognition:

En la barra de búsqueda escriba “Rekognition”. Luego haga clock sobre el ícono.

Ingrese una imagen:

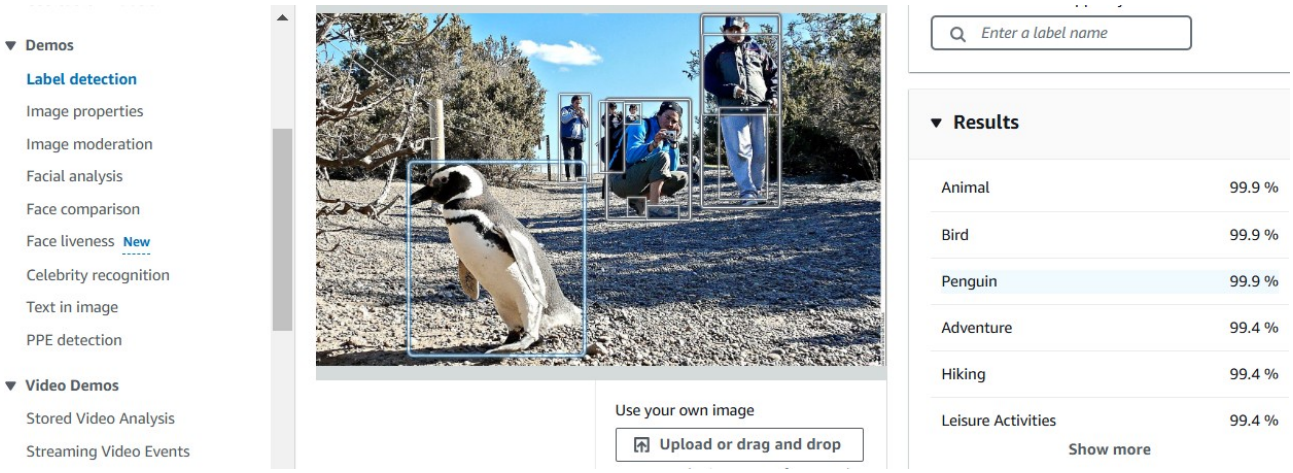
Puede usar una imagen de ejemplo, o bien analizar una imagen propia. Para ello, puede subirla desde su equipo, o ingresar URL



Ingresaremos una imagen de un pingüino de la pinguinera Punta Tombo: foto03.png

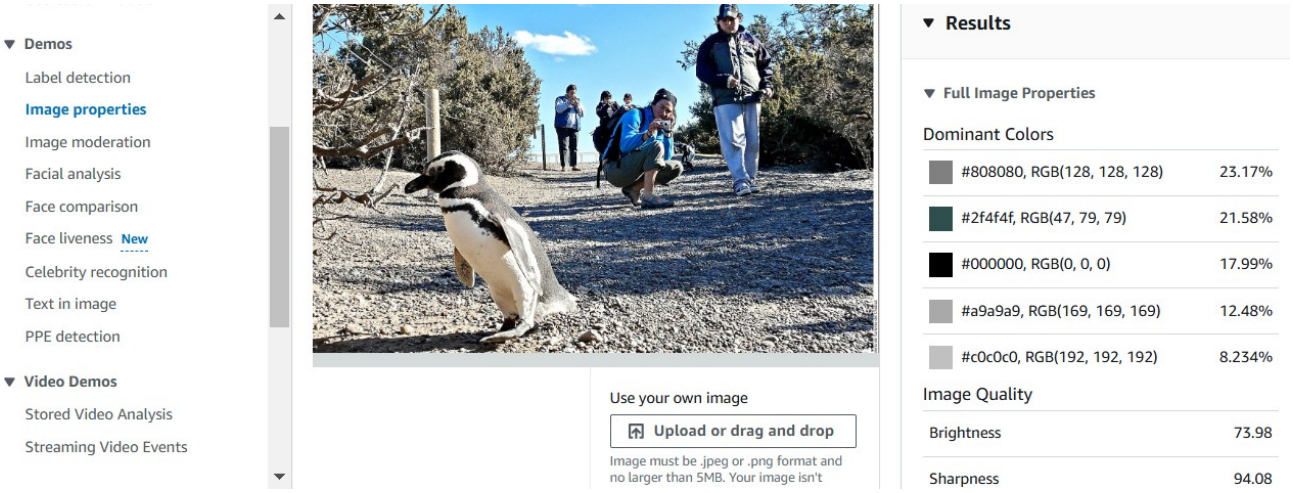
Label Detection

Se hace click en el menú que aparece a la izquierda



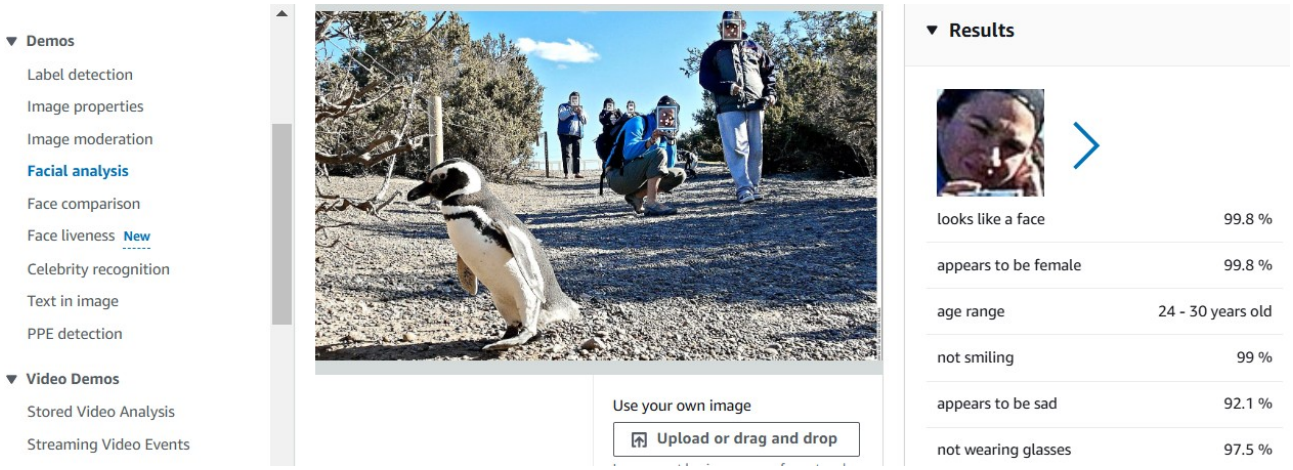
Se presentan resultados relevantes, pero también pueden descargarse todos los resultados haciendo click en: “Download full list”.

Image properties



Se presentan acá propiedades y características de la imagen, no del contenido.

Face Analysis:



Tras detectar un rostro, realiza un análisis del mismo.

1.3. Ejemplo con Python

Imagen: foto.png

Código Python: rekognition_build_json.py

```python
import glob
import boto3
import json

client = boto3.client('rekognition')
combined = []
for filename in glob.glob('public/photos/*.jpeg'):
    with open(filename, 'rb') as fd:
        response = client.detect_labels(Image={'Bytes': fd.read()})
        entry = {  "Filename": filename.replace("public/", "") }
        entry["Labels"] = [
            {
                "Name": label['Name'],
                "Confidence": label['Confidence'],
                "Parents": label['Parents']
            }
            for label in response['Labels']
        ]
        combined.append(entry)

print(json.dumps(combined, indent=2))
```

Salida Json:


```
{
    "Filename": "fotos\\foto03.png",
    "Labels": [
      {
        "Name": "Animal",
        "Confidence": 99.94088745117188,
        "Parents": []
      },
      {
        "Name": "Bird",
        "Confidence": 99.94088745117188,
        "Parents": [
          {
            "Name": "Animal"
          }
        ]
      },
      {
        "Name": "Penguin",
        "Confidence": 99.94088745117188,
        "Parents": [
          {
            "Name": "Animal"
          },
          {
            "Name": "Bird"
          }
        ]
… (hay más)
}
```


2. Textract

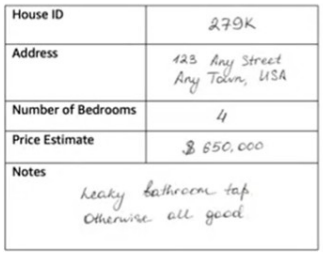
Es un servicio de aprendizaje automático que analiza documentos de texto y automáticamente extrae textos y estructuras.

2.1. Operaciones

Textract realiza las siguientes operaciones:

- AnalyzeDocument: Documentos en General
- AnalyzeExpense: Facturas, recibos, etc.
- AnalyzeID: DNI, licencias de conducir, pasaportes, etc.
- DetectDocumentText

2.1.1. Analisis de Documentos:



House ID	279K
Address	123 Any Street Any Town, USA
Number of Bedrooms	4
Price Estimate	$ 650,000
Notes	Leaky bathroom tap. Otherwise all good.

Request:

```
"Text": "What are the notes?"
"Alias": "Notes"
```

Response:

```
"BlockType": "QUERY_RESULT",
"Confidence": 95.0,
"Text": "Leaky bathroom tap. Otherwise all good.",
```

2.1.2. AnalyzeExpense


```
"SummaryFields": [
  {
    "Type": {
      "Text": "VENDOR_NAME", "Confidence":
97.87217712402344
    },
    "ValueDetection": {
      "Text": "WHOLE FOODS\nMARKET",
      "Geometry": ..,
      "Confidence": 70.45414733886719
    },
  },
  {
    "Type": {
      "Text": "OTHER",
      "Confidence": 84
    },
    "LabelDetection": {
      "Text": "Net Sales:",
```

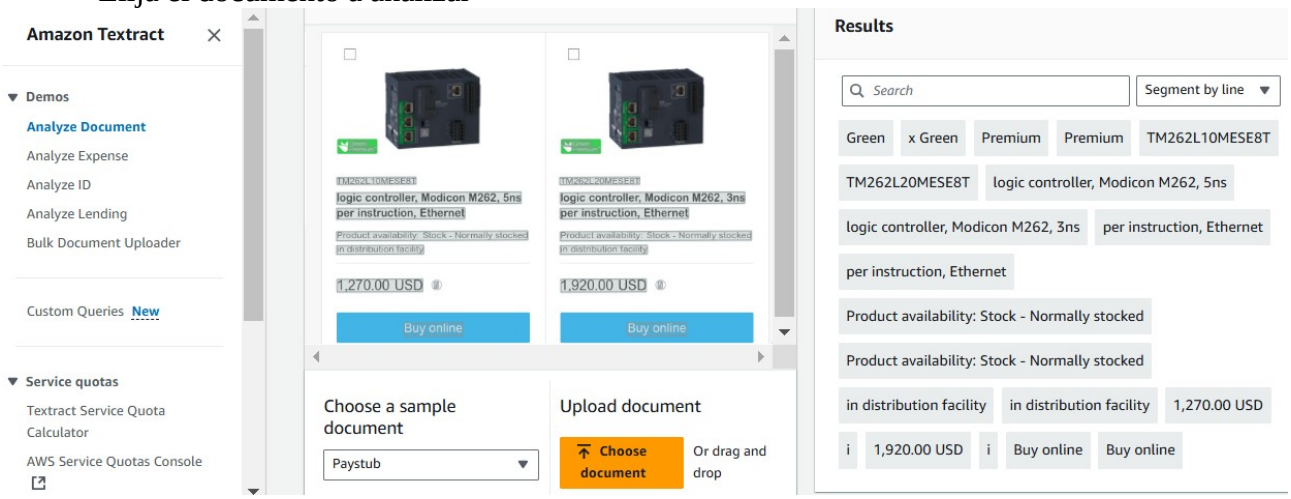
2.1.3. AnalyzeID


```
"IdentityDocumentFields": [
  {
    "Type": { "Text": "FIRST_NAME" },
    "ValueDetection": { "Text": "RUSSELL", "Confidence":
99.3445053100586 }
  },
  {
    "Type": { "Text": "LAST_NAME" },
    "ValueDetection": { "Text": "SAYERS", "Confidence":
98.8835678100586 }
  },
  {
    "Type": { "Text": "MIDDLE_NAME" },
    ...
  },
  {
    "Type": { "Text": "SUFFIX" },
    ...
  },
```

2.2. Ejemplo de uso en consola de AWS

- Ingresar a AWS
- En la barra de búsqueda escribir “Textract”

- Elija el documento a analizar

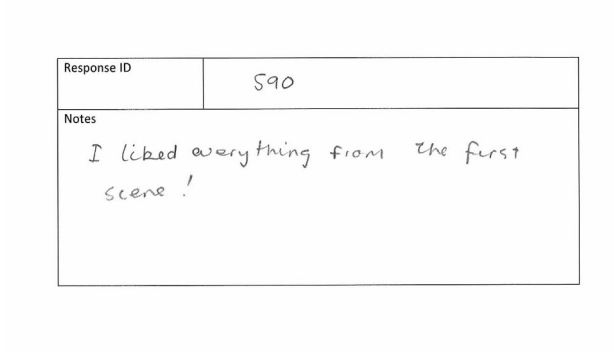


Se observa que se ha detectado correctamente la totalidad del texto presentado dentro de la imagen.

2.3. Ejemplo con Python

2.3.1. analyse_document

Figura: 001.jpg



Código: textract_main.py

```python
# Librerías
import glob
import boto3
import json
import csv
import sys

csv_array = []
client = boto3.client('textract')
for filename in glob.glob("raw_images/001.jpg"):
    csv_row = {}
    print(f"Processing: {filename}")
    with open(filename, 'rb') as fd:
        file_bytes = fd.read()
    # Procesamiento de imágenes
    response = client.analyze_document(
        Document={'Bytes': file_bytes},
        FeatureTypes=["QUERIES"],
        QueriesConfig={
            'Queries': [
                {'Text': 'What is the response id', 'Alias': 'ResponseId'},
                {'Text': 'What are the notes?', 'Alias': 'Notes'},
            ]
        }
    )
    # Response completa a Textract
    print(json.dumps(response, indent=4))

    for block in response["Blocks"]:
        if block["BlockType"] == "QUERY":
            query_alias = block["Query"]["Alias"]
            answer_id = next(rel["Ids"] for rel in block["Relationships"] if rel['Type'] == "ANSWER")[0]
            answer_text = next(b for b in response["Blocks"] if b["Id"] == answer_id)["Text"]
            csv_row[query_alias] = answer_text
    csv_array.append(csv_row)
# Escritura de CSV
writer = csv.DictWriter(sys.stdout, fieldnames=["ResponseId", "Notes"], dialect='excel')
writer.writeheader()
for row in csv_array:
    writer.writerow(row)
```

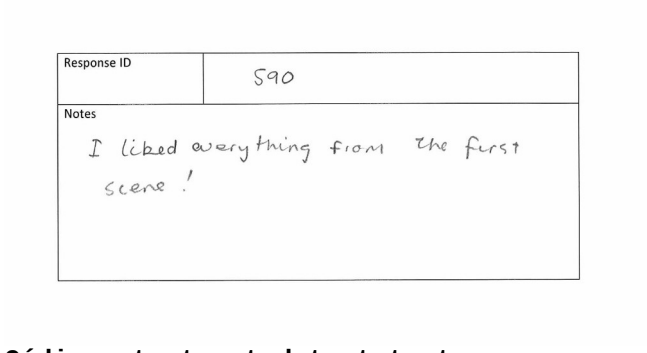
Salida:


```
{
"BlockType": "QUERY",
"Id": "5b95eff1-7b10-4611-9aab-0ffbe1ead1be",
"Relationships": [
  {
  "Type": "ANSWER",
  "Ids": ["2705cd0e-bb73-40df-8954-b91b6ed0d77c"],
  "Query": {
  "Text": "What is the response id",
  "Alias": "ResponseId"}
},
{
"BlockType": "QUERY_RESULT",
"Confidence": 99.0,
"Text": "590",
"Geometry": {
  "BoundingBox": {
    "Width": 0.06050015240907669,
    "Height": 0.01886042021214962,
    "Left": 0.4029218554496765,
    "Top": 0.09675661474466324},
  "Id": "2705cd0e-bb73-40df-8954-b91b6ed0d77c"
},
{
"BlockType": "QUERY",
"Id": "8301f8c4-2eb4-4596-88d1-f06d6df622e9",
"Relationships": [
  {
  "Type": "ANSWER",
  "Ids": ["cbe23a09-2f1e-4972-bdf1-54bcef0f0bb0"],
  "Query": {
  "Text": "What are the notes?",
  "Alias": "Notes"}
},
{
"BlockType": "QUERY_RESULT",
"Confidence": 98.0,
"Text":"I liked every thing from the first scene",
  "Geometry": {
    "BoundingBox": {
      "Width": 0.6340733766555786,
      "Height": 0.06546258926391602,
      "Left": 0.16090142726898193,
      "Top": 0.16538071632385254},
}
```

En este ejemplo, Textract busca la respuesta a una pregunta presentada en el código. Por ello, la estrategia es aquí, buscar la “query” y su correspondiente “query_result”.

2.3.2. detect_text

Figura: 001.jpg



Código: textract_detect_text.py

```python
import glob
import boto3
import json
import csv
import sys

csv_array = []
client = boto3.client('textract')
for filename in glob.glob('raw_images/001.jpg'):
    csv_row = {}
    print(f"Processing: {filename}")
    with open(filename, 'rb') as fd:
        file_bytes = fd.read()

    # Procesamiento de imágenes sin consultas
    response = client.detect_document_text(
        Document={'Bytes': file_bytes}
    )

    # Response completa a Textract
    print(json.dumps(response, indent=4))

    # Procesamiento de la respuesta
    extracted_text = ""
    for block in response["Blocks"]:
        if block["BlockType"] == "LINE":
            extracted_text += block["Text"] + "\n"

    csv_row["ExtractedText"] = extracted_text.strip()
    csv_array.append(csv_row)

# Escritura de CSV
writer = csv.DictWriter(sys.stdout, fieldnames=["ExtractedText"], dialect='excel')
writer.writeheader()
for row in csv_array:
    writer.writerow(row)
```

Salida:


```
{
"BlockType": "LINE",
"Confidence": 99.8954849243164,
"Text": "Response ID",
"Geometry": {
  "BoundingBox": {
    "Width": 0.0989999920129776,
    "Height": 0.013172809965908527,
    "Left": 0.12449004501104355,
    "Top": 0.08601831644773483},},
  "Polygon": [
    {"X": 0.12449190020561218,
     "Y": 0.08608761429786682},
    {"X": 0.22349004447460175,
     "Y": 0.08601831644773483},
    {"X": 0.22348883748054504,
     "Y": 0.09912172704935074},
    {"X": 0.12449004501104355,
     "Y": 0.09919112920761108} ]
},
{
"BlockType": "LINE",
"Confidence": 98.7627182006836,
"Text": "590",
...
},
{
"BlockType": "LINE",
"Confidence": 99.92236328125,
"Text": "Notes",
...
},
{
"BlockType": "LINE",
"Confidence": 95.04239654541016,
"Text": "I liked every thing from the first",
...
},
```


El uso de detect_text puede resultar más sencillo, ya que trae todo el texto detectado. Luego es posible separar los contenidos según necesidad.

3. Comprehend

Amazon – Comprehend es un servicio de procesamiento de lenguaje natural basado en aprendizaje profundo para obtener y descubrir perspectivas y conexiones valiosas a partir de un texto.

3.1. Operaciones Básicas

Principales operaciones de AWS-Comprehend:

- Detección de Entidades
- Frases principales
- Idioma
- Información de Identificadores Personales
- Análisis de sentimientos
- Análisis Sintáctico
- Modelos personalizados de clasificación de textos y entidades

Detecting the dominant language

```
detect-dominant-language
batch-detect-dominant-language
start-dominant-language-detection-job
```

```
{
  "Text": "Mi maestra es Guatamalteca"
}
```

→

```
"Languages": {
 "LanguageCode": "es",
 "Score": 0.8301774859428406
}
```

Detecting named entities: Results

```
{
 "Text": "Hello, I am John from AnyCompany Financial Services.",
 "LanguageCode": "en"
}
```

```
"Entities": [
   {
     "Score": 0.9996919631958008,
     "Type": "PERSON",
     "Text": "John",
     "BeginOffset": 12,
     "EndOffset": 16
   },
   {
     "Score": 0.9934418201446533,
     "Type": "ORGANIZATION",
     "Text": "AnyCompany Financial Services",
     "BeginOffset": 22,
     "EndOffset": 51
```

Detecting PII entities: Results

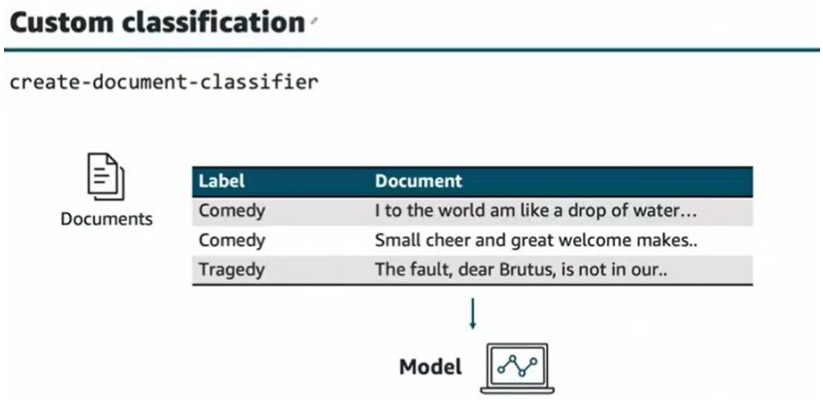
```
{
 "Text": "Hello John, your address is 123 Test St",
 "LanguageCode": "en"
}
```

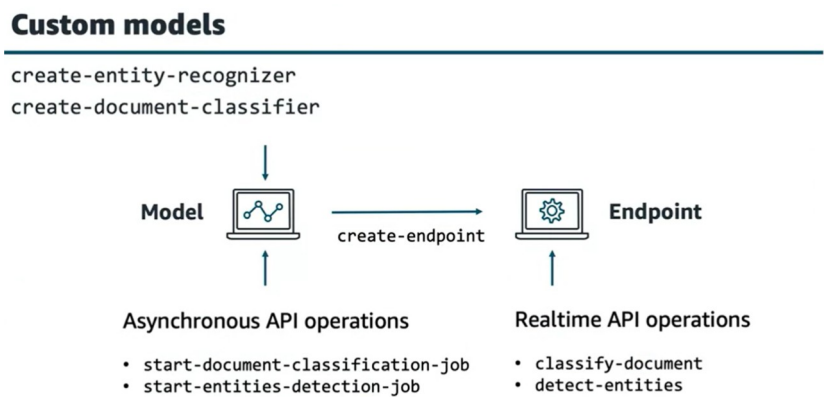
```
"Entities": [
 {
  "Score": 0.9999914169311523,
  "Type": "NAME",
  "BeginOffset": 6,
  "EndOffset": 10
 },
 {
  "Score": 0.9999768733978271,
  "Type": "ADDRESS",
  "BeginOffset": 28,
  "EndOffset": 39
 }
```

Detecting sentiment: Results

```
{
 "Text": "I loved shopping here",
 "LanguageCode": "en"
}
```

```
"Sentiment": {
 "Sentiment": "POSITIVE",
 "SentimentScore": {
  "Positive": 0.99887,
  "Negative": 0.00006,
  "Neutral": 0.00085,
  "Mixed": 0.00020
 }
}
```

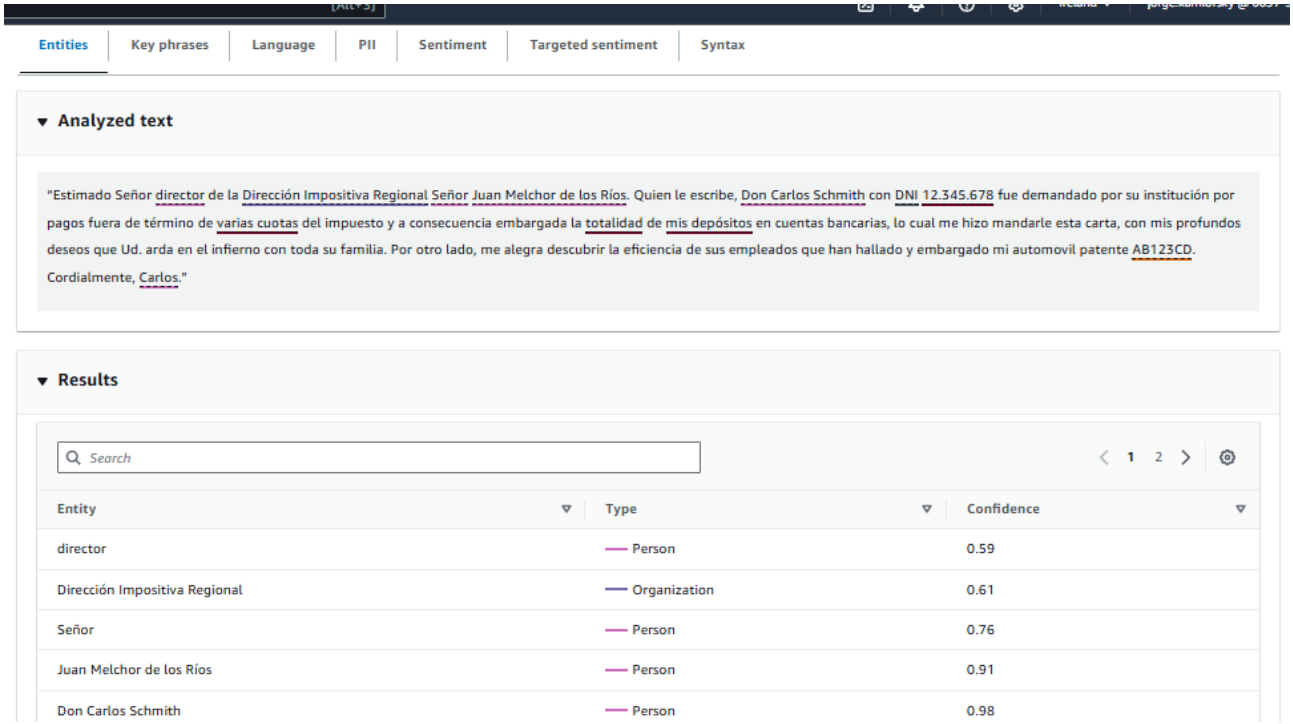




3.2. Un ejemplo desafiante en la consola de AWS-Comprehend

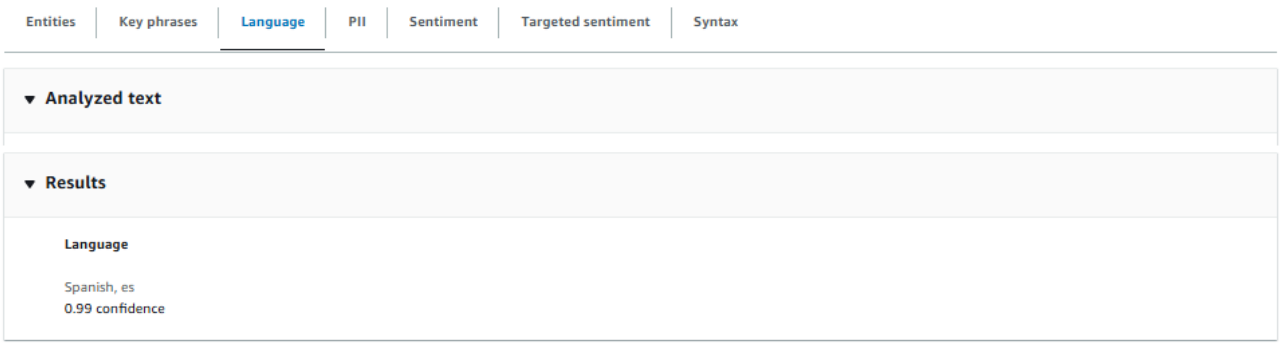
Vamos a introducir un texto en español que incluya información personal, sentimientos variados e incluso ironía, entidades, y demás. Pondremos a prueba a este sistema.
El texto: *"Estimado Señor director de la Dirección Impositiva Regional Señor Juan Melchor de los Ríos. Quien le escribe, Don Carlos Schmith con DNI 12.345.678 fue demandado por su institución por pagos fuera de término de varias cuotas del impuesto y a consecuencia embargada la totalidad de mis depósitos en cuentas bancarias, lo cual me hizo mandarle esta carta, con mis profundos deseos que Ud. arda en el infierno con toda su familia. Por otro lado, me alegra descubrir la eficiencia de sus empleados que han hallado y embargado mi automovil patente AB123CD. Cordialmente, Carlos."*

3.2.1. Entidades



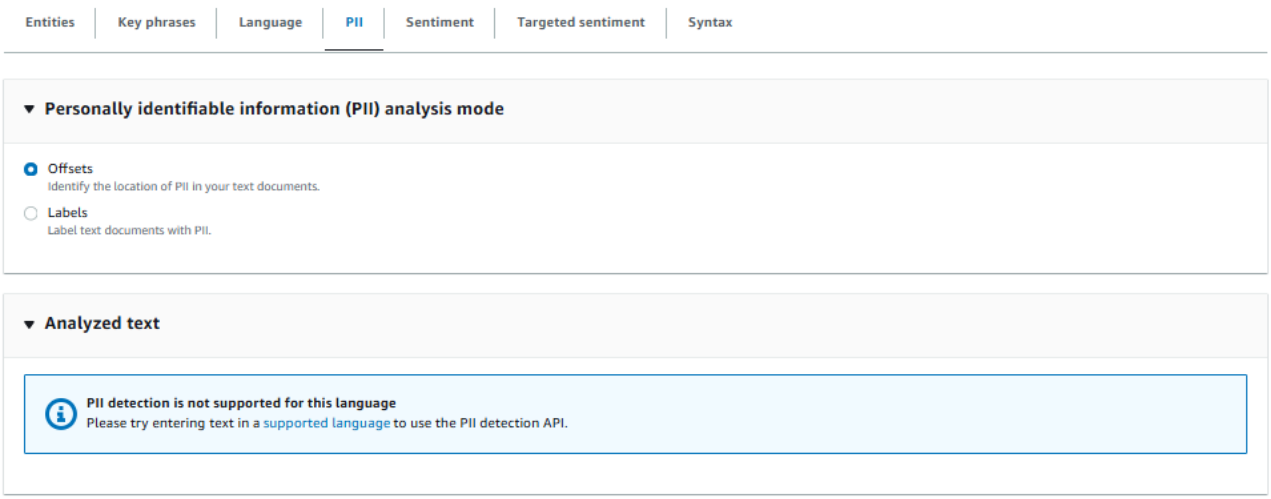
Observar que identifica adecuadamente personas y organizaciones.

3.2.2. Idioma



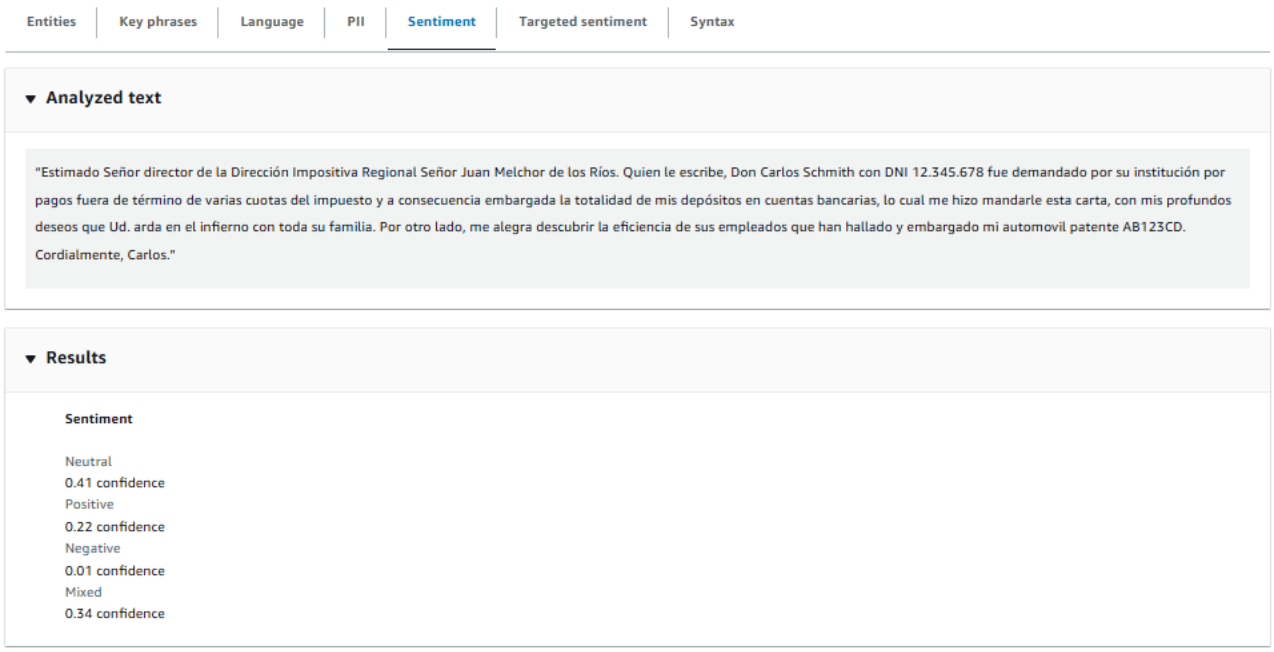
Detección correcta.

3.2.3. Información de Identificadores Personales



No disponible en Español.

3.2.4. Sentimientos



Observar que domina el sentimiento neutral seguido por un sentimiento positivo. Al sentimiento negativo le atribuye un 1% de confianza, cuando por la ironía contenida, es un texto fuertemente negativo.

3.3. Un ejemplo con Python