Proyecto Pipeline de Procesamiento de Video: Análisis y Etiquetado de Videos Mediante AWS

Integrantes:

- Patrick Xavier Marquez Choque
- Jean Carlo Cornejo Cornejo
- Oscar Andree Mendoza Alejandro

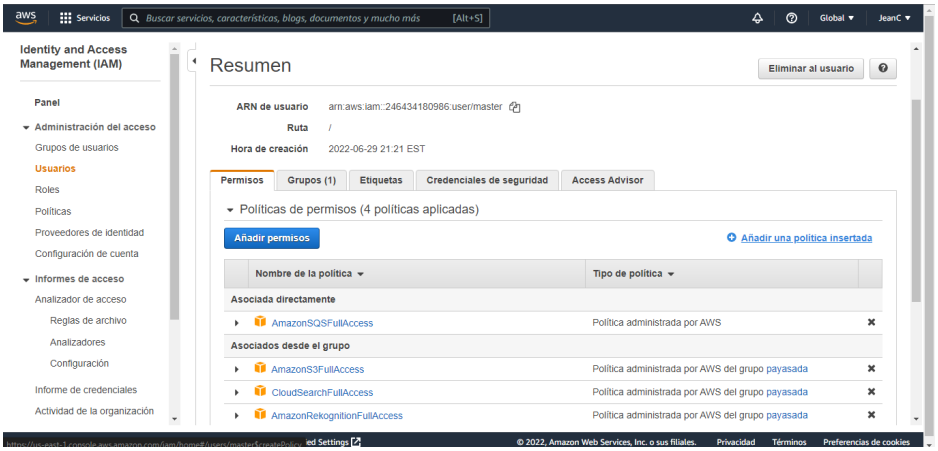
Objetivos

Usar el Servicio AWS-Lambda (o afines) y FFmpeg para procesar de video Implementar un mecanismo de reconocimiento para etiquetado de imágenes (like AWS Rekognition) Implementar una forma de indexación basado en videos sobre etiquetas (like AWS CloudSearch)

Introducción

El link del repositorio con el código fuente es el siguiente: https://github.com/patrick03524/Proyecto-Pipeline-de-Procesamiento-de-Video-con-AWS/blob/main/AWSLambda.ipynb

En el siguiente documento se explican los pasos qué se tomaron en cuenta para realizar el trabajo de etiquetar videos mediante un análisis usando **AWS Rekognition** y una búsqueda mediante **AWS CloudSearch** para realizar consultas sobre las etiquetas presentes en los videos.



Se realizaron pruebas procesando 3 vídeos cortos en los frames utilizando además la librería FFmpeg para obtener así los resultados, guardarlos en una carpeta de Drive y posteriormente procesarlos con Google Colab.

Metodología y Experimentos

Paso 1: Instalación de la SDK de Amazon Web services (AWS) -> (Boto3 AWS SDK)

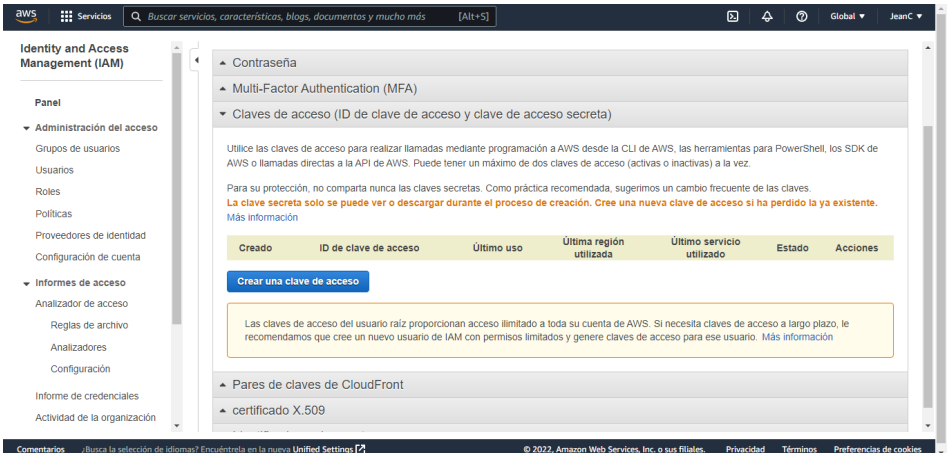
El trabajo se realizó usando google colab y las funciones se toman de las librerías de AWS, específicamente de su SDK, Boto3, facilitando el procesado y las queries a realizar al servicio, a su vez qué el manejo de archivos y datos.

```
[1] !pip install boto3

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting boto3
  Downloading boto3-1.24.21-py3-none-any.whl (132 kB)
     |████████████████████████████████| 132 kB 5.2 MB/s
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 6.8 MB/s
Collecting botocore<1.28.0,>=1.27.21
  Downloading botocore-1.27.21-py3-none-any.whl (8.9 MB)
     |████████████████████████████████| 8.9 MB 53.3 MB/s
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting urllib3<1.27,>=1.25.4
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 49.9 MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/dist-packages (from botocore<1.28.0,>=1.27.21->boto3) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.28.0,>=1.27.21->boto3) (1.15.0)
Installing collected packages: urllib3, jmespath, botocore, s3transfer, boto3
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
    Uninstalling urllib3-1.24.3:
      Successfully uninstalled urllib3-1.24.3
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
requests 2.23.0 requires urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1, but you have urllib3 1.26.9 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
Successfully installed boto3-1.24.21 botocore-1.27.21 jmespath-1.0.1 s3transfer-0.6.0 urllib3-1.26.9
```

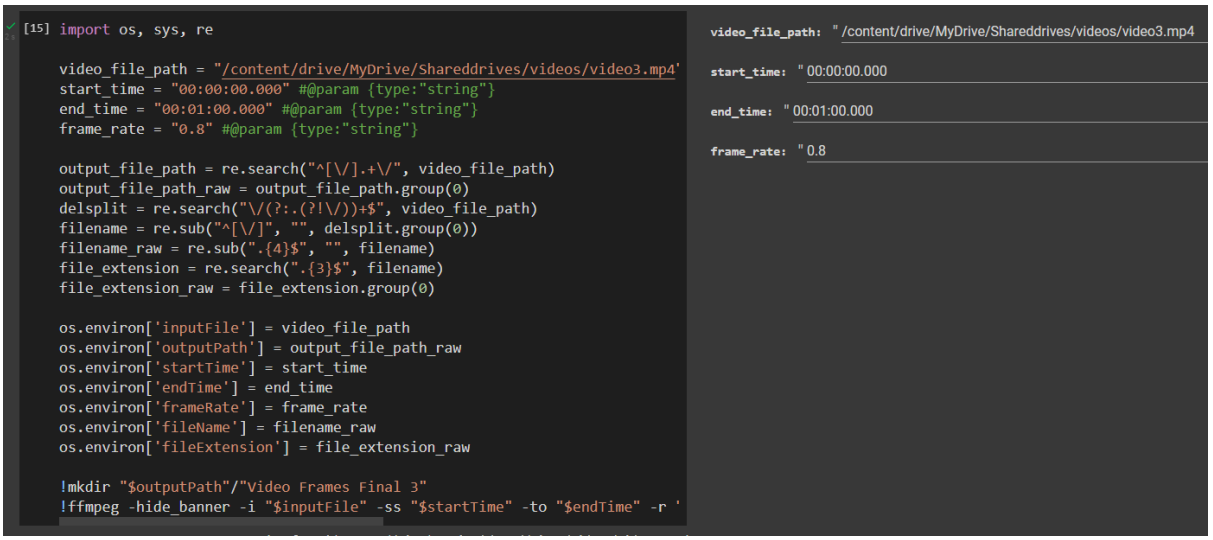
Paso 2: Configuración de AWS en Google Colaboratory -> (AWS CLI)

También se toma en cuenta qué es necesario configurar los contenedores de S3 y los servicios a utilizar, mediante una cuenta, atribuyendo los permisos necesarios para acceder de manera remota.



Paso 3: Convertir los videos a Frames -> (FFmpeg)

El workflow comienza con videos cargados a google drive qué son procesados con FFmpeg, para obtener una cantidad de frames del video y almacenarlos.



```
[15] import os, sys, re

     video_file_path = "/content/drive/MyDrive/Shareddrives/videos/video3.mp4"
     start_time = "00:00:00.000" #@param {type:"string"}
     end_time = "00:01:00.000" #@param {type:"string"}
     frame_rate = "0.8" #@param {type:"string"}

     output_file_path = re.search("^[\/].+\/", video_file_path)
     output_file_path_raw = output_file_path.group(0)
     delsplit = re.search("\/(?:.(?!\/))+$", video_file_path)
     filename = re.sub("^[\/]", "", delsplit.group(0))
     filename_raw = re.sub(".{4}$", "", filename)
     file_extension = re.search(".{3}$", filename)
     file_extension_raw = file_extension.group(0)

     os.environ['inputFile'] = video_file_path
     os.environ['outputPath'] = output_file_path_raw
     os.environ['startTime'] = start_time
     os.environ['endTime'] = end_time
     os.environ['frameRate'] = frame_rate
     os.environ['fileName'] = filename_raw
     os.environ['fileExtension'] = file_extension_raw

     !mkdir "$outputPath"/"Video Frames Final 3"
     !ffmpeg -hide_banner -i "$inputFile" -ss "$startTime" -to "$endTime" -r '
```

```
video_file_path: " /content/drive/MyDrive/Shareddrives/videos/video3.mp4
start_time: " 00:00:00.000
end_time: " 00:01:00.000
frame_rate: " 0.8
```

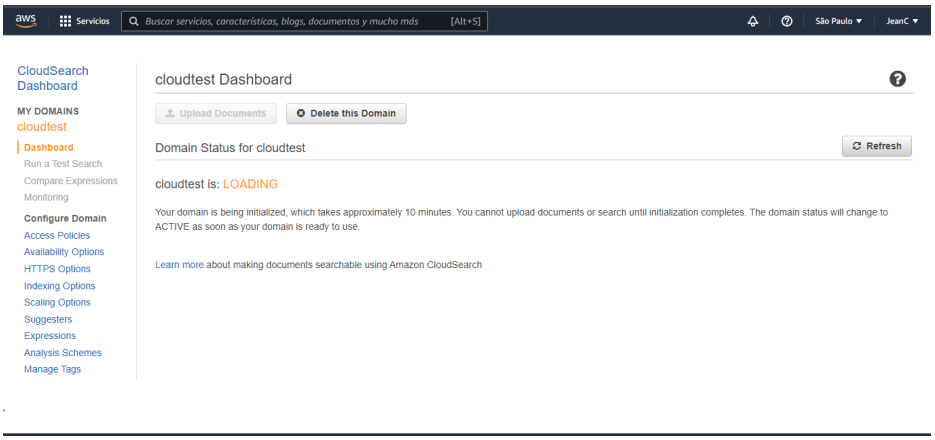
Paso 4: Convertir los Frames a una lista de diccionarios para poder subirlo a AWS -> (AWS Rekognition)

Luego de la obtención de frames, se hace una llamada a AWS Rekognition, pasando como dato cada frame tomado y de estos se obtienen las etiquetas correspondientes a cada uno de estos archivos, a su vez las etiquetas son almacenadas.

```
[11] def aws_rekognition(image):
       client = boto3.client('rekognition')
       with open(image, 'rb') as image_source:
         image_bytes = image_source.read()
       response = client.detect_labels(Image={'Bytes':image_bytes}, MaxLabels=10)
       return response
```

Paso 5: Cargar la lista de diccionario como un archivo .csv y posteriormente subirlo dentro de un dominio -> (AWS CloudSearch)

En el mismo google colab se realiza un procesado de las etiquetas, ya qué son listas de diccionarios y esto puede complicar un poco el workflow, por ello se manejan para extraerlas como archivo csv y son subidos al sistema de amazon CloudSearch

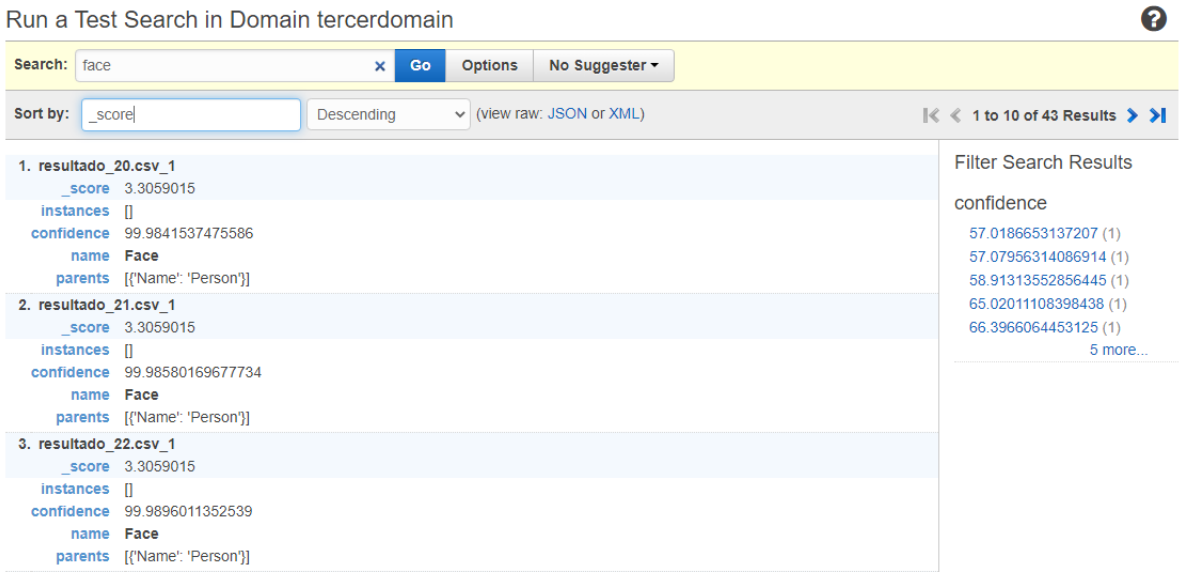


Paso 6: Posteriormente se realizan las consultas de las búsquedas en el dominio(por nombre, confidencialidad y referencia) -> (Query AWS CloudSearch)

Luego de subir los archivos a CloudSearch se pueden realizar búsquedas basadas en las diferentes etiquetas presentes en cada archivo mediante la web de Amazon.

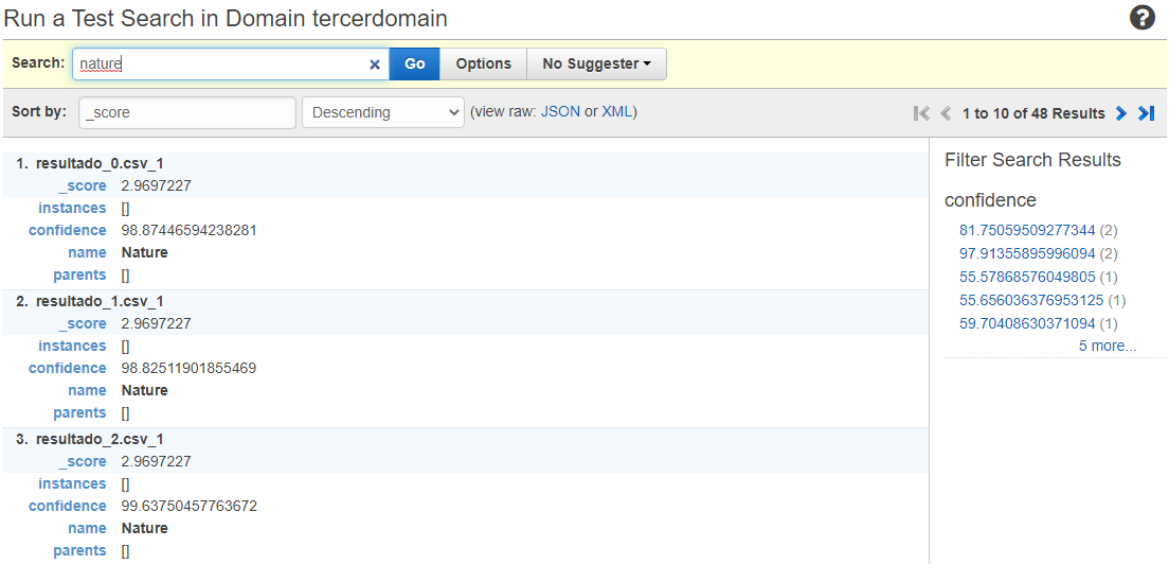
Como experimentos hemos realizado el etiquetado de 3 vídeos con poda duración(aproximadamente 10 segundos) para obtener etiquetas de los frames

en cada segundo que aparecen y así deducir las similitudes y diferencias de estos videos.



En esta captura podemos apreciar la pantalla de consulta de etiquetas con el nombre “face”, estas etiquetas están organizadas en 30 archivos con el siguiente rango:

- Etiquetas con el nombre resultado 0 - 9 pertenecen a los primeros 10 frames del primer video
- Etiquetas con el nombre resultado 10 - 19 pertenecen a los primeros 10 frames del segundo video
- Etiquetas con el nombre resultado 20 - 29 pertenecen a los primeros 10 frames del tercer video



Conclusión

1. *Como conclusión de este trabajo podemos mencionar que utilizar estas herramientas son bastante útiles cuando se quiere realizar algún tipo de tarea de procesamiento, sobre todo procesamiento de video. Utilizando el SDK de Amazon nos da mucha facilidad para utilizar los comandos de los servicios como **Rekognition** y **CloudSearch**; obteniendo así etiquetas de los objetos que aparecen en los videos utilizados en los experimentos realizados.*

2. *De los resultados obtenidos en nuestros 3 videos, queremos destacar la utilidad de la herramienta para realizar el reconocimiento de datos en las imágenes, es así como se puede observar qué nuestros primeros dos videos comparten similitudes en las etiquetas, en contraste, nuestro tercer video es muy diferente a los previos y las etiquetas hacen referencia a ese resultado.*