

Ronjon Kundu
23215183

Setting up Python Environment

By using the code to make the environment for language detection

pip3 install iso-639

After that modify a python script to detect the language with the confidence level for four different languages

```
ronjon@ronjon-VirtualBox: ~/lab9
import boto3
import math
from iso639 import languages

client = boto3.client('comprehend')

# Detect Entities - English
response = client.detect_dominant_language(
    Text="The French Revolution was a period of social and political upheaval in France and its colonies beginning in 1789 and ending in 1799.",
)
print("English")
print(response['Languages'])
print((languages.get(alpha2 = response['Languages'][0]['LanguageCode']).name) + ' detected with ' + str(math.floor(response['Languages'][0]['Score']*100)) + '% confidence')

# Detect Entities - Spanish
response_spanish = client.detect_dominant_language(
    Text="El Quijote es la obra más conocida de Miguel de Cervantes Saavedra. Publicada su primera parte con el título de El ingenioso hidalgo don Quijote de la Mancha a comienzos de 1605, es una de las obras más destacadas de la literatura española y la literatura universal, y una de las más traducidas. En 1615 aparecería la segunda parte del Quijote de Cervantes con el título de El ingenioso caballero don Quijote de la Mancha.",
)
print("Spanish")
print(response_spanish['Languages'])
print((languages.get(alpha2 = response_spanish['Languages'][0]['LanguageCode']).name) + ' detected with ' + str(math.floor(response_spanish['Languages'][0]['Score']*100)) + '% confidence')

# Detect Entities - French
response_french = client.detect_dominant_language(
    Text="Moi je n'étais rien Et voilà qu'aujourd'hui Je suis le gardien Du sommeil de ses nuits Je l'aime à mourir Vous pouvez détruire Tout ce qu'il vous plaira Elle n'a qu'à ouvrir L'espace de ses bras Pour tout reconstruire Pour tout reconstruire Je l'aime à mourir",
)
print("French")
print(response_french['Languages'])
print((languages.get(alpha2 = response_french['Languages'][0]['LanguageCode']).name) + ' detected with ' + str(math.floor(response_french['Languages'][0]['Score']*100)) + '% confidence')

# Detect Entities - Italian
response_italian = client.detect_dominant_language(
    Text="L'amor che move il sole e l'altre stelle.",
)
print("Italian")
print(response_italian['Languages'])
print((languages.get(alpha2 = response_italian['Languages'][0]['LanguageCode']).name) + ' detected with ' + str(math.floor(response_italian['Languages'][0]['Score']*100)) + '% confidence')
```

Fig 1: Detect languages code

```
ronjon@ronjon-VirtualBox:~/lab9$ python3 language_detect.py
English
[{'LanguageCode': 'en', 'Score': 0.9961233139038886}]
English detected with 99% confidence
Spanish
[{'LanguageCode': 'es', 'Score': 0.9980844259262885}]
Spanish detected with 99% confidence
French
[{'LanguageCode': 'fr', 'Score': 0.997617244720459}]
French detected with 99% confidence
Italian
[{'LanguageCode': 'it', 'Score': 0.993222532272339}]
Italian detected with 99% confidence
ronjon@ronjon-VirtualBox:~/lab9$
```

Fig 2: Output of language detection

After modifying the python script to detect the Sentiment of the language for four different languages

```
### STEP 2 - NLU Sentiment Analysis

# NLU - English
english_text = "The French Revolution was a period of social and political upheaval in France and its colonies beginning in 1789 and ending in 1799."
sentiment_english = client.detect_sentiment(Text = english_text,
                                             LanguageCode = response['Languages'][0]['LanguageCode'])

print("English - Sentiment Analysis")
print(sentiment_english['Sentiment'])
print(sentiment_english['SentimentScore'])

# NLU - Spanish
spanish_text = "El Quijote es la obra más conocida de Miguel de Cervantes Saavedra. Publicada su primera parte con el título de El ingenioso hidalgo don Quijote de la Mancha a comienzos de 1605, es una de las obras más destacadas de la literatura española y la literatura universal, y una de las más traducidas. En 1615 aparecería la segunda parte del Quijote de Cervantes con el título de El ingenioso caballero don Quijote de la Mancha."
sentiment_spanish = client.detect_sentiment(Text = spanish_text,
                                             LanguageCode = response_spanish['Languages'][0]['LanguageCode'])

print("Spanish - Sentiment Analysis")
print(sentiment_spanish['Sentiment'])
print(sentiment_spanish['SentimentScore'])

# NLU - French
french_text = "Moi je n'étais rien Et voilà qu'aujourd'hui Je suis le gardien Du sommeil de ses nuits Je l'aime à mourir Vous pouvez détruire Tout ce qu'il vous plaira Elle n'a qu'à ouvrir L'espace de ses bras Pour tout reconstruire Pour tout reconstruire Je l'aime à mourir"
sentiment_french = client.detect_sentiment(Text = french_text,
                                             LanguageCode = response_french['Languages'][0]['LanguageCode'])

print("French - Sentiment Analysis")
print(sentiment_french['Sentiment'])
print(sentiment_french['SentimentScore'])

# NLU - Italian
italian_text = "L'amor che move il sole e l'altre stelle."
sentiment_italian = client.detect_sentiment(Text = italian_text,
                                             LanguageCode = response_italian['Languages'][0]['LanguageCode'])

print("Italian - Sentiment Analysis")
print(sentiment_italian['Sentiment'])
print(sentiment_italian['SentimentScore'])
```

Fig 3: Detect languages sentiments code

```

ronjon@ronjon-VirtualBox:~/Lab9$ python3 language_detect.py
English - Sentiment Analysis
NEUTRAL
{'Positive': 0.00022422113397624344, 'Negative': 0.0005216890713199973, 'Neutral': 0.9992488026618958, 'Mixed': 5.199335191719001e-06}
Spanish - Sentiment Analysis
NEUTRAL
{'Positive': 0.1598433405160904, 'Negative': 0.0008151692454703152, 'Neutral': 0.8380755186080933, 'Mixed': 0.001266040955670178}
French - Sentiment Analysis
NEGATIVE
{'Positive': 0.3935803174972534, 'Negative': 0.5921157598495483, 'Neutral': 0.013823595829308033, 'Mixed': 0.0004802222247235477}
Italian - Sentiment Analysis
POSITIVE
{'Positive': 0.985332190990448, 'Negative': 0.004254049155861139, 'Neutral': 0.009053979068994522, 'Mixed': 0.0013597647193819284}
ronjon@ronjon-VirtualBox:~/Lab9$

```

Fig 4: Detect languages sentiments output

After modifying the python script to detect the entities of the language for four different languages

```

#####
### STEP 3 -Detecting entities
# Detect entities - English
entity_english = client.detect_entities(
    Text=english_text,
    LanguageCode=response['Languages'][0]['LanguageCode']
)
print("English")
print(entity_english['Entities'])

# Detect entities - Spanish
entity_spanish = client.detect_entities(
    Text=spanish_text,
    LanguageCode=response_spanish['Languages'][0]['LanguageCode']
)
print("Spanish")
print(entity_spanish['Entities'])

# Detect entities - French
entity_french = client.detect_entities(
    Text=french_text,
    LanguageCode=response_french['Languages'][0]['LanguageCode']
)
print("French")
print(entity_french['Entities'])

# Detect entities - Italian
entity_italian = client.detect_entities(
    Text=italian_text,
    LanguageCode=response_italian['Languages'][0]['LanguageCode']
)
print("Italian")
print(entity_italian['Entities'])

```

Fig 5: Detect languages entities code

```

ronjon@ronjon-VirtualBox:~/Lab9$ python3 language_detect.py
ronjon@ronjon-VirtualBox:~/Lab9$ python3 language_detect.py
English
[{'Score': 0.9943565130233765, 'Type': 'EVENT', 'Text': 'French Revolution', 'BeginOffset': 4, 'EndOffset': 21}, {'Score': 0.99414416740417, 'Type': 'LOCATION', 'Text': 'France', 'BeginOffset': 71, 'EndOffset': 77}, {'Score': 0.9984388947486877, 'Type': 'DATE', 'Text': '1789', 'BeginOffset': 188, 'EndOffset': 194}, {'Score': 0.9990737438201904, 'Type': 'DATE', 'Text': '1799', 'BeginOffset': 127, 'EndOffset': 131}]
Spanish
[{'Score': 0.9886308908462524, 'Type': 'TITLE', 'Text': 'El Quijote', 'BeginOffset': 0, 'EndOffset': 10}, {'Score': 0.9992279410362244, 'Type': 'PERSON', 'Text': 'Miguel de Cervantes Saavedra', 'BeginOffset': 38, 'EndOffset': 66}, {'Score': 0.9098508609397888, 'Type': 'QUANTITY', 'Text': 'primera parte', 'BeginOffset': 81, 'EndOffset': 94}, {'Score': 0.9835476279259728, 'Type': 'TITLE', 'Text': 'El ingenioso hidalgo don Quijote de la Mancha', 'BeginOffset': 112, 'EndOffset': 157}, {'Score': 0.8697961568832397, 'Type': 'DATE', 'Text': '1605', 'BeginOffset': 173, 'EndOffset': 177}, {'Score': 0.6641538143157959, 'Type': 'QUANTITY', 'Text': 'una', 'BeginOffset': 182, 'EndOffset': 185}, {'Score': 0.9596462845802307, 'Type': 'OTHER', 'Text': 'española', 'BeginOffset': 231, 'EndOffset': 239}, {'Score': 0.719598114490509, 'Type': 'QUANTITY', 'Text': 'una de las más traducidas', 'BeginOffset': 269, 'EndOffset': 294}, {'Score': 0.9852679967880249, 'Type': 'DATE', 'Text': '1615', 'BeginOffset': 299, 'EndOffset': 303}, {'Score': 0.94227135181427, 'Type': 'QUANTITY', 'Text': 'segunda parte', 'BeginOffset': 318, 'EndOffset': 331}, {'Score': 0.9497652649879456, 'Type': 'TITLE', 'Text': 'Quijote de Cervantes', 'BeginOffset': 336, 'EndOffset': 356}, {'Score': 0.9899224638938904, 'Type': 'TITLE', 'Text': 'El ingenioso caballero don Quijote de la Mancha', 'BeginOffset': 374, 'EndOffset': 421}]
French
[{'Score': 0.9872375726699829, 'Type': 'DATE', 'Text': 'aujourd'hui', 'BeginOffset': 32, 'EndOffset': 43}, {'Score': 0.6959105134010315, 'Type': 'QUANTITY', 'Text': 'Tout ce qu'il', 'BeginOffset': 127, 'EndOffset': 140}, {'Score': 0.6048811078071594, 'Type': 'QUANTITY', 'Text': 'tout', 'BeginOffset': 200, 'EndOffset': 204}, {'Score': 0.5311335325241089, 'Type': 'QUANTITY', 'Text': 'tout', 'BeginOffset': 223, 'EndOffset': 227}]
Italian
[]
ronjon@ronjon-VirtualBox:~/Lab9$

```

Fig 6: Detect languages entities output

After modifying the python script to detect the keyphrases of the language for four different languages

```

#####
### STEP 4 -Detecting keyphrases
# Detect keyphrases - English
keyphrase_english = client.detect_key_phrases(
    Text=english_text,
    LanguageCode = response['Languages'][0]['LanguageCode']
)
print("English")
print(keyphrase_english['KeyPhrases'])

# Detect keyphrases - Spanish
keyphrase_spanish = client.detect_key_phrases(
    Text=spanish_text,
    LanguageCode = response_spanish['Languages'][0]['LanguageCode']
)
print("Spanish")
print(keyphrase_spanish['KeyPhrases'])

# Detect keyphrases - French
keyphrase_french = client.detect_key_phrases(
    Text=french_text,
    LanguageCode = response_french['Languages'][0]['LanguageCode']
)
print("French")
print(keyphrase_french['KeyPhrases'])

# Detect keyphrases - Italian
keyphrase_italian = client.detect_key_phrases(
    Text=italian_text,
    LanguageCode = response_italian['Languages'][0]['LanguageCode']
)
print("Italian")
print(keyphrase_italian['KeyPhrases'])

```

Fig 7: Detect languages keyphrases code


```

English
[{'Score': 0.9999843239784241, 'Text': 'The French Revolution', 'BeginOffset': 0, 'EndOffset': 21}, {'Score': 0.999777674674988, 'Text': 'a period', 'BeginOffset': 26, 'EndOffset': 34}, {'Score': 0.999951241912842, 'Text': 'social and political upheaval', 'BeginOffset': 38, 'EndOffset': 67}, {'Score': 0.9999372959136963, 'Text': 'France', 'BeginOffset': 71, 'EndOffset': 77}, {'Score': 0.9999833703841077, 'Text': 'its colonies', 'BeginOffset': 82, 'EndOffset': 94}, {'Score': 0.9999758005142212, 'Text': '1789', 'BeginOffset': 108, 'EndOffset': 112}, {'Score': 0.9999212826590609, 'Text': '1799', 'BeginOffset': 127, 'EndOffset': 131}]
Spanish
[{'Score': 0.9994539022445679, 'Text': 'El Quijote', 'BeginOffset': 0, 'EndOffset': 10}, {'Score': 0.9999617338180542, 'Text': 'la obra', 'BeginOffset': 14, 'EndOffset': 21}, {'Score': 0.9988439679145813, 'Text': 'nás conocida', 'BeginOffset': 22, 'EndOffset': 34}, {'Score': 0.999977296303406, 'Text': 'Miguel de Cervantes Saavedra', 'BeginOffset': 38, 'EndOffset': 66}, {'Score': 0.999961226844788, 'Text': 'su primera parte', 'BeginOffset': 78, 'EndOffset': 94}, {'Score': 0.9999792575836182, 'Text': 'el título', 'BeginOffset': 99, 'EndOffset': 108}, {'Score': 0.956083834712952, 'Text': 'El ingenioso hidalgo don Quijote de la Mancha', 'BeginOffset': 112, 'EndOffset': 157}, {'Score': 0.9998094439506531, 'Text': 'comenzos', 'BeginOffset': 160, 'EndOffset': 169}, {'Score': 0.9962199330329895, 'Text': '1605', 'BeginOffset': 173, 'EndOffset': 177}, {'Score': 0.9999752044677734, 'Text': 'las obras', 'BeginOffset': 189, 'EndOffset': 198}, {'Score': 0.9998710751533508, 'Text': 'nás destacadas', 'BeginOffset': 199, 'EndOffset': 213}, {'Score': 0.9999703168869019, 'Text': 'la literatura es pañola', 'BeginOffset': 217, 'EndOffset': 239}, {'Score': 0.9999440312385559, 'Text': 'la literatura universal', 'BeginOffset': 242, 'EndOffset': 265}, {'Score': 0.9995015263557434, 'Text': 'las más traducidas', 'BeginOffset': 276, 'EndOffset': 294}, {'Score': 0.9999786019325256, 'Text': 'la segunda parte', 'BeginOffset': 315, 'EndOffset': 331}, {'Score': 0.9999293088912964, 'Text': 'Quijote de Cervantes', 'BeginOffset': 336, 'EndOffset': 356}, {'Score': 0.9999479855404663, 'Text': 'el título', 'BeginOffset': 361, 'EndOffset': 370}, {'Score': 0.9364849328994751, 'Text': 'Ingenioso caballero don Quijote de la Mancha', 'BeginOffset': 377, 'EndOffset': 421}]
French
[{'Score': 0.999990433120728, 'Text': 'Moi', 'BeginOffset': 0, 'EndOffset': 3}, {'Score': 0.9550336599349976, 'Text': 'Je', 'BeginOffset': 4, 'EndOffset': 6}, {'Score': 0.9563182592319168, 'Text': 'n'étais rien', 'BeginOffset': 7, 'EndOffset': 19}, {'Score': 0.9490335583686829, 'Text': 'aujourd'hui', 'BeginOffset': 32, 'EndOffset': 43}, {'Score': 0.94124943017959, 'Text': 'Je suis le gardien du sommeil de ses nuits', 'BeginOffset': 44, 'EndOffset': 86}, {'Score': 0.9999150037765503, 'Text': 'Je', 'BeginOffset': 87, 'EndOffset': 89}, {'Score': 0.9998370409011841, 'Text': 'L', 'BeginOffset': 90, 'EndOffset': 92}, {'Score': 0.9990161657333374, 'Text': 'Vous', 'BeginOffset': 106, 'EndOffset': 110}, {'Score': 0.9828138351440, 'Text': 'Tout ce', 'BeginOffset': 127, 'EndOffset': 134}, {'Score': 0.9646840691566467, 'Text': 'qu', 'BeginOffset': 135, 'EndOffset': 138}, {'Score': 0.9985837936401367, 'Text': 'Il', 'BeginOffset': 138, 'EndOffset': 140}, {'Score': 0.9997034668922424, 'Text': 'vous', 'BeginOffset': 141, 'EndOffset': 145}, {'Score': 0.9997631907463074, 'Text': 'Elle', 'BeginOffset': 153, 'EndOffset': 157}, {'Score': 0.9946832658680352, 'Text': 'L'espace de ses bras', 'BeginOffset': 174, 'EndOffset': 194}, {'Score': 0.8932915329933167, 'Text': 'tout', 'BeginOffset': 200, 'EndOffset': 204}, {'Score': 0.9624954462851392, 'Text': 'tout', 'BeginOffset': 223, 'EndOffset': 227}, {'Score': 0.9998551607131958, 'Text': 'Je', 'BeginOffset': 241, 'EndOffset': 243}, {'Score': 0.9999161958694458, 'Text': 'L', 'BeginOffset': 244, 'EndOffset': 246}]
Italian
[{'Score': 0.9999063618039579, 'Text': 'L'amor', 'BeginOffset': 0, 'EndOffset': 6}, {'Score': 0.9997649788856506, 'Text': 'che', 'BeginOffset': 7, 'EndOffset': 10}, {'Score': 0.99997717421051, 'Text': 'il sole', 'BeginOffset': 16, 'EndOffset': 23}, {'Score': 0.9999290180680725, 'Text': 'l'altra stelle', 'BeginOffset': 26, 'EndOffset': 40}]
ronjon@ronjon-VirtualBox:~/Lab9$

```

Fig 8: Detect languages keyphrases output

After modifying the python script to detect the syntax of the language for four different languages

```

### STEP 5 - Detecting syntax
# Detect syntax - English
syntax_english = client.detect_syntax(
    Text=english_text,
    LanguageCode=response['Languages'][0]['LanguageCode']
)
print("English")
print(syntax_english['SyntaxTokens'])

# Detect syntax - Spanish
syntax_spanish = client.detect_syntax(
    Text=spanish_text,
    LanguageCode=response_spanish['Languages'][0]['LanguageCode']
)
print("Spanish")
print(syntax_spanish['SyntaxTokens'])

# Detect syntax - French
syntax_french = client.detect_syntax(
    Text=french_text,
    LanguageCode=response_french['Languages'][0]['LanguageCode']
)
print("French")
print(syntax_french['SyntaxTokens'])

# Detect syntax - Italian
syntax_italian = client.detect_syntax(
    Text=italian_text,
    LanguageCode=response_italian['Languages'][0]['LanguageCode']
)
print("Italian")
print(syntax_italian['SyntaxTokens'])

```

Fig 9: Detect languages syntax code

```

ronjon@ronjon-VirtualBox:~/Lab9$ python language_detect.py
English
[{"token": "The", "start": 0, "end": 1, "score": 0.9999843239784241}, {"token": "French", "start": 1, "end": 2, "score": 0.999977674674988}, {"token": "Revolution", "start": 2, "end": 3, "score": 0.999951241912842}, {"token": "social", "start": 3, "end": 4, "score": 0.9999372959136963}, {"token": "and", "start": 4, "end": 5, "score": 0.9999833703841077}, {"token": "political", "start": 5, "end": 6, "score": 0.9999758005142212}, {"token": "upheaval", "start": 6, "end": 7, "score": 0.9999212826590609}, {"token": "in", "start": 7, "end": 8, "score": 0.9999843239784241}, {"token": "1789", "start": 8, "end": 9, "score": 0.999977674674988}, {"token": "a", "start": 9, "end": 10, "score": 0.999951241912842}, {"token": "period", "start": 10, "end": 11, "score": 0.9999372959136963}, {"token": "in", "start": 11, "end": 12, "score": 0.9999833703841077}, {"token": "France", "start": 12, "end": 13, "score": 0.9999758005142212}, {"token": "its", "start": 13, "end": 14, "score": 0.9999212826590609}, {"token": "colonies", "start": 14, "end": 15, "score": 0.9999843239784241}, {"token": "in", "start": 15, "end": 16, "score": 0.999977674674988}, {"token": "1799", "start": 16, "end": 17, "score": 0.999951241912842}, {"token": "at", "start": 17, "end": 18, "score": 0.9999372959136963}, {"token": "the", "start": 18, "end": 19, "score": 0.9999833703841077}, {"token": "end", "start": 19, "end": 20, "score": 0.9999758005142212}], [{"token": "El", "start": 0, "end": 1, "score": 0.9994539022445679}, {"token": "Quijote", "start": 1, "end": 2, "score": 0.9999617338180542}, {"token": "de", "start": 2, "end": 3, "score": 0.9988439679145813}, {"token": "la", "start": 3, "end": 4, "score": 0.999977296303406}, {"token": "obra", "start": 4, "end": 5, "score": 0.999961226844788}, {"token": "de", "start": 5, "end": 6, "score": 0.9999792575836182}, {"token": "Miguel", "start": 6, "end": 7, "score": 0.956083834712952}, {"token": "de", "start": 7, "end": 8, "score": 0.9998094439506531}, {"token": "Cervantes", "start": 8, "end": 9, "score": 0.9962199330329895}, {"token": "Saavedra", "start": 9, "end": 10, "score": 0.9999752044677734}, {"token": "en", "start": 10, "end": 11, "score": 0.9998710751533508}, {"token": "1605", "start": 11, "end": 12, "score": 0.9999703168869019}, {"token": "en", "start": 12, "end": 13, "score": 0.9999440312385559}, {"token": "la", "start": 13, "end": 14, "score": 0.9995015263557434}, {"token": "literatura", "start": 14, "end": 15, "score": 0.9999786019325256}, {"token": "es", "start": 15, "end": 16, "score": 0.9999293088912964}, {"token": "pa\u00f1ola", "start": 16, "end": 17, "score": 0.9999479855404663}, {"token": "de", "start": 17, "end": 18, "score": 0.9364849328994751}, {"token": "la", "start": 18, "end": 19, "score": 0.9999290180680725}, {"token": "literatura", "start": 19, "end": 20, "score": 0.9999843239784241}, {"token": "universal", "start": 20, "end": 21, "score": 0.999977674674988}, {"token": "de", "start": 21, "end": 22, "score": 0.999951241912842}, {"token": "las", "start": 22, "end": 23, "score": 0.9999372959136963}, {"token": "m\u00e1s", "start": 23, "end": 24, "score": 0.9999833703841077}, {"token": "traducidas", "start": 24, "end": 25, "score": 0.9999758005142212}, {"token": "de", "start": 25, "end": 26, "score": 0.9999212826590609}, {"token": "la", "start": 26, "end": 27, "score": 0.9999843239784241}, {"token": "segunda", "start": 27, "end": 28, "score": 0.999977674674988}, {"token": "parte", "start": 28, "end": 29, "score": 0.999951241912842}, {"token": "de", "start": 29, "end": 30, "score": 0.9999372959136963}, {"token": "Cervantes", "start": 30, "end": 31, "score": 0.9999833703841077}, {"token": "en", "start": 31, "end": 32, "score": 0.9999758005142212}, {"token": "el", "start": 32, "end": 33, "score": 0.9999212826590609}, {"token": "t\u00edtulo", "start": 33, "end": 34, "score": 0.9999843239784241}, {"token": "de", "start": 34, "end": 35, "score": 0.999977674674988}, {"token": "1789", "start": 35, "end": 36, "score": 0.999951241912842}, {"token": "y", "start": 36, "end": 37, "score": 0.9999372959136963}, {"token": "1799", "start": 37, "end": 38, "score": 0.9999833703841077}, {"token": "at", "start": 38, "end": 39, "score": 0.9999758005142212}, {"token": "the", "start": 39, "end": 40, "score": 0.9999212826590609}, {"token": "end", "start": 40, "end": 41, "score": 0.9999843239784241}], [{"token": "L'amor", "start": 0, "end": 1, "score": 0.9999063618039579}, {"token": "che", "start": 1, "end": 2, "score": 0.9997649788856506}, {"token": "il", "start": 2, "end": 3, "score": 0.99997717421051}, {"token": "sole", "start": 3, "end": 4, "score": 0.9999290180680725}, {"token": "l'altra", "start": 4, "end": 5, "score": 0.9999843239784241}, {"token": "stelle", "start": 5, "end": 6, "score": 0.999977674674988}, {"token": "in", "start": 6, "end": 7, "score": 0.999951241912842}, {"token": "il", "start": 7, "end": 8, "score": 0.9999372959136963}, {"token": "sole", "start": 8, "end": 9, "score": 0.9999833703841077}, {"token": "e", "start": 9, "end": 10, "score": 0.9999758005142212}, {"token": "il", "start": 10, "end": 11, "score": 0.9999212826590609}, {"token": "sole", "start": 11, "end": 12, "score": 0.9999843239784241}, {"token": "e", "start": 12, "end": 13, "score": 0.999977674674988}, {"token": "l'altra", "start": 13, "end": 14, "score": 0.999951241912842}, {"token": "stelle", "start": 14, "end": 15, "score": 0.9999372959136963}, {"token": "in", "start": 15, "end": 16, "score": 0.9999833703841077}, {"token": "il", "start": 16, "end": 17, "score": 0.9999758005142212}, {"token": "sole", "start": 17, "end": 18, "score": 0.9999212826590609}, {"token": "e", "start": 18, "end": 19, "score": 0.9999843239784241}, {"token": "il", "start": 19, "end": 20, "score": 0.999977674674988}, {"token": "sole", "start": 20, "end": 21, "score": 0.999951241912842}, {"token": "e", "start": 21, "end": 22, "score": 0.9999372959136963}, {"token": "il", "start": 22, "end": 23, "score": 0.9999833703841077}, {"token": "sole", "start": 23, "end": 24, "score": 0.9999758005142212}, {"token": "e", "start": 24, "end": 25, "score": 0.9999212826590609}, {"token": "il", "start": 25, "end": 26, "score": 0.9999843239784241}, {"token": "sole", "start": 26, "end": 27, "score": 0.999977674674988}, {"token": "e", "start": 27, "end": 28, "score": 0.999951241912842}, {"token": "il", "start": 28, "end": 29, "score": 0.9999372959136963}, {"token": "sole", "start": 29, "end": 30, "score": 0.9999833703841077}, {"token": "e", "start": 30, "end": 31, "score": 0.9999758005142212}, {"token": "il", "start": 31, "end": 32, "score": 0.9999212826590609}, {"token": "sole", "start": 32, "end": 33, "score": 0.9999843239784241}, {"token": "e", "start": 33, "end": 34, "score": 0.999977674674988}, {"token": "l'altra", "start": 34, "end": 35, "score": 0.999951241912842}, {"token": "stelle", "start": 35, "end": 36, "score": 0.9999372959136963}, {"token": "in", "start": 36, "end": 37, "score": 0.9999833703841077}, {"token": "il", "start": 37, "end": 38, "score": 0.9999758005142212}, {"token": "sole", "start": 38, "end": 39, "score": 0.9999212826590609}, {"token": "e", "start": 39, "end": 40, "score": 0.9999843239784241}, {"token": "il", "start": 40, "end": 41, "score": 0.999977674674988}, {"token": "sole", "start": 41, "end": 42, "score": 0.999951241912842}, {"token": "e", "start": 42, "end": 43, "score": 0.9999372959136963}, {"token": "il", "start": 43, "end": 44, "score": 0.9999833703841077}, {"token": "sole", "start": 44, "end": 45, "score": 0.9999758005142212}, {"token": "e", "start": 45, "end": 46, "score": 0.9999212826590609}, {"token": "il", "start": 46, "end": 47, "score": 0.9999843239784241}, {"token": "sole", "start": 47, "end": 48, "score": 0.999977674674988}, {"token": "e", "start": 48, "end": 49, "score": 0.999951241912842}, {"token": "il", "start": 49, "end": 50, "score": 0.9999372959136963}, {"token": "sole", "start": 50, "end": 51, "score": 0.9999833703841077}, {"token": "e", "start": 51, "end": 52, "score": 0.9999758005142212}, {"token": "il", "start": 52, "end": 53, "score": 0.9999212826590609}, {"token": "sole", "start": 53, "end": 54, "score": 0.9999843239784241}, {"token": "e", "start": 54, "end": 55, "score": 0.999977674674988}, {"token": "l'altra", "start": 55, "end": 56, "score": 0.999951241912842}, {"token": "stelle", "start": 56, "end": 57, "score": 0.9999372959136963}, {"token": "in", "start": 57, "end": 58, "score": 0.9999833703841077}, {"token": "il", "start": 58, "end": 59, "score": 0.9999758005142212}, {"token": "sole", "start": 59, "end": 60, "score": 0.9999212826590609}, {"token": "e", "start": 60, "end": 61, "score": 0.9999843239784241}, {"token": "il", "start": 61, "end": 62, "score": 0.999977674674988}, {"token": "sole", "start": 62, "end": 63, "score": 0.999951241912842}, {"token": "e", "start": 63, "end": 64, "score": 0.9999372959136963}, {"token": "il", "start": 64, "end": 65, "score": 0.9999833703841077}, {"token": "sole", "start": 65, "end": 66, "score": 0.9999758005142212}, {"token": "e", "start": 66, "end": 67, "score": 0.9999212826590609}, {"token": "il", "start": 67, "end": 68, "score": 0.9999843239784241}, {"token": "sole", "start": 68, "end": 69, "score": 0.999977674674988}, {"token": "e", "start": 69, "end": 70, "score": 0.999951241912842}, {"token": "il", "start": 70, "end": 71, "score": 0.9999372959136963}, {"token": "sole", "start": 71, "end": 72, "score": 0.9999833703841077}, {"token": "e", "start": 72, "end": 73, "score": 0.9999758005142212}, {"token": "il", "start": 73, "end": 74, "score": 0.9999212826590609}, {"token": "sole", "start": 74, "end": 75, "score": 0.9999843239784241}, {"token": "e", "start": 75, "end": 76, "score": 0.999977674674988}, {"token": "l'altra", "start": 76, "end": 77, "score": 0.999951241912842}, {"token": "stelle", "start": 77, "end": 78, "score": 0.9999372959136963}, {"token": "in", "start": 78, "end": 79, "score": 0.9999833703841077}, {"token": "il", "start": 79, "end": 80, "score": 0.9999758005142212}, {"token": "sole", "start": 80, "end": 81, "score": 0.9999212826590609}, {"token": "e", "start": 81, "end": 82, "score": 0.9999843239784241}, {"token": "il", "start": 82, "end": 83, "score": 0.999977674674988}, {"token": "sole", "start": 83, "end": 84, "score": 0.999951241912842}, {"token": "e", "start": 84, "end": 85, "score": 0.9999372959136963}, {"token": "il", "start": 85, "end": 86, "score": 0.9999833703841077}, {"token": "sole", "start": 86, "end": 87, "score": 0.9999758005142212}, {"token": "e", "start": 87, "end": 88, "score": 0.9999212826590609}, {"token": "il", "start": 88, "end": 89, "score": 0.9999843239784241}, {"token": "sole", "start": 89, "end": 90, "score": 0.999977674674988}, {"token": "e", "start": 90, "end": 91, "score": 0.999951241912842}, {"token": "il", "start": 91, "end": 92, "score": 0.9999372959136963}, {"token": "sole", "start": 92, "end": 93, "score": 0.9999833703841077}, {"token": "e", "start": 93, "end": 94, "score": 0.9999758005142212}, {"token": "il", "start": 94, "end": 95, "score": 0.9999212826590609}, {"token": "sole", "start": 95, "end": 96, "score": 0.9999843239784241}, {"token": "e", "start": 96, "end": 97, "score": 0.999977674674988}, {"token": "l'altra", "start": 97, "end": 98, "score": 0.999951241912842}, {"token": "stelle", "start": 98, "end": 99, "score": 0.9999372959136963}, {"token": "in", "start": 99, "end": 100, "score": 0.9999833703841077}, {"token": "il", "start": 100, "end": 101, "score": 0.9999758005142212}, {"token": "sole", "start": 101, "end": 102, "score": 0.9999212826590609}, {"token": "e", "start": 102, "end": 103, "score": 0.9999843239784241}, {"token": "il", "start": 103, "end": 104, "score": 0.999977674674988}, {"token": "sole", "start": 104, "end": 105, "score": 0.999951241912842}, {"token": "e", "start": 105, "end": 106, "score": 0.9999372959136963}, {"token": "il", "start": 106, "end": 107, "score": 0.9999833703841077}, {"token": "sole", "start": 107, "end": 108, "score": 0.9999758005142212}, {"token": "e", "start": 108, "end": 109, "score": 0.9999212826590609}, {"token": "il", "start": 109, "end": 110, "score": 0.9999843239784241}, {"token": "sole", "start": 110, "end": 111, "score": 0.999977674674988}, {"token": "e", "start": 111, "end": 112, "score": 0.999951241912842}, {"token": "il", "start": 112, "end": 113, "score": 0.9999372959136963}, {"token": "sole", "start": 113, "end": 114, "score": 0.9999833703841077}, {"token": "e", "start": 114, "end": 115, "score": 0.9999758005142212}, {"token": "il", "start": 115, "end": 116, "score": 0.9999212826590609}, {"token": "sole", "start": 116, "end": 117, "score": 0.9999843239784241}, {"token": "e", "start": 117, "end": 118, "score": 0.999977674674988}, {"token": "l'altra", "start": 118, "end": 119, "score": 0.999951241912842}, {"token": "stelle", "start": 119, "end": 120, "score": 0.9999372959136963}, {"token": "in", "start": 120, "end": 121, "score": 0.9999833703841077}, {"token": "il", "start": 121, "end": 122, "score": 0.9999758005142212}, {"token": "sole", "start": 122, "end": 123, "score": 0.9999212826590609}, {"token": "e", "start": 123, "end": 124, "score": 0.9999843239784241}, {"token": "il", "start": 124, "end": 125, "score": 0.999977674674988}, {"token": "sole", "start": 125, "end": 126, "score": 0.999951241912842}, {"token": "e", "start": 126, "end": 127, "score": 0.9999372959136963}, {"token": "il", "start": 127, "end": 128, "score": 0.9999833703841077}, {"token": "sole", "start": 128, "end": 129, "score": 0.9999758005142212}, {"token": "e", "start": 129, "end": 130, "score": 0.9999212826590609}, {"token": "il", "start": 130, "end": 131, "score": 0.9999843239784241}, {"token": "sole", "start": 131, "end": 132, "score": 0.999977674674988}, {"token": "e", "start": 132, "end": 133, "score": 0.999951241912842}, {"token": "il", "start": 133, "end": 134, "score": 0.9999372959136963}, {"token": "sole", "start": 134, "end": 135, "score": 0.9999833703841077}, {"token": "e", "start": 135, "end": 136, "score": 0.9999758005142212}, {"token": "il", "start": 136, "end": 137, "score": 0.9999212826590609}, {"token": "sole", "start": 137, "end": 138, "score": 0.9999843239784241}, {"token": "e", "start": 138, "end": 139, "score": 0.999977674674988}, {"token": "l'altra", "start": 139, "end": 140, "score": 0.999951241912842}, {"token": "stelle", "start": 140, "end": 141, "score": 0.9999372959136963}, {"token": "in", "start": 141, "end": 142, "score": 0.9999833703841077}, {"token": "il", "start": 142, "end": 143, "score": 0.9999758005142212}, {"token": "sole", "start": 143, "end": 144, "score": 0.9999212826590609}, {"token": "e", "start": 144, "end": 145, "score": 0.9999843239784241}, {"token": "il", "start": 145, "end": 146, "score": 0.999977674674988}, {"token": "sole", "start": 146, "end": 147, "score": 0.999951241912842}, {"token": "e", "start": 147, "end": 148, "score": 0.9999372959136963}, {"token": "il", "start": 148, "end": 149, "score": 0.9999833703841077}, {"token": "sole", "start": 149, "end": 150, "score": 0.9999758005142212}, {"token": "e", "start": 150, "end": 151, "score": 0.9999212826590609}, {"token": "il", "start": 151, "end": 152, "score": 0.9999843239784241}, {"token": "sole", "start": 152, "end": 153, "score": 0.999977674674988}, {"token": "e", "start": 153, "end": 154, "score": 0.999951241912842}, {"token": "il", "start": 154, "end": 155, "score": 0.9999372959136963}, {"token": "sole", "start": 155, "end": 156, "score": 0.9999833703841077}, {"token": "e", "start": 156, "end": 157, "score": 0.9999758005142212}, {"token": "il", "start": 157, "end": 158, "score": 0.9999212826590609}, {"token": "sole", "start": 158, "end": 159, "score": 0.9999843239784241}, {"token": "e", "start": 159, "end": 160, "score": 0.999977674674988}, {"token": "l'altra", "start": 160, "end": 161, "score": 0.
```

Entities - An *entity* of a text is the real-world object such as people, places, and commercial items in it.

Key Phrase - A *key phrase* is a noun the modifiers that distinguish the entities in a text.

Syntax – It is the words from the document. The nouns, verbs, adjectives and so on in a document. Syntax analysis helps to understand the relationship of the words in the document.

A bucket is created with AWS console and uploaded four different pictures for Label Recognition, Image Moderation, Facial Analysis and Text detection from an image.

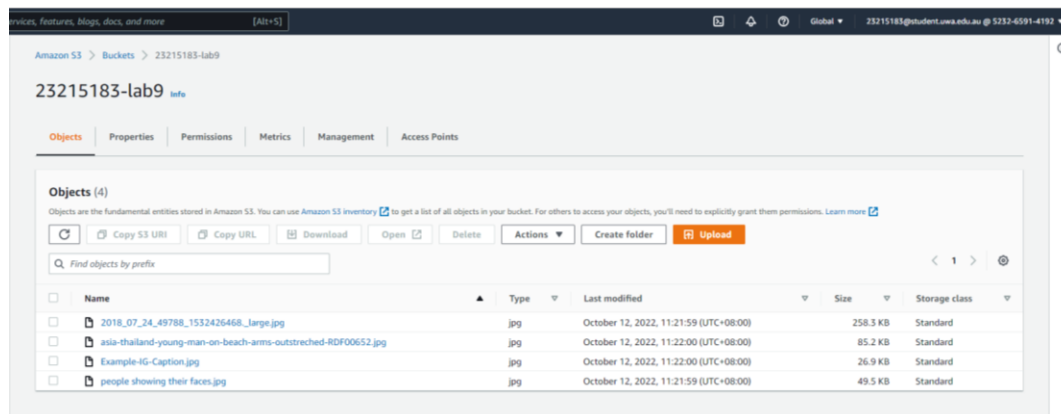


Fig 11: S3 bucket with pictures

Then modify a python script for Label Recognition

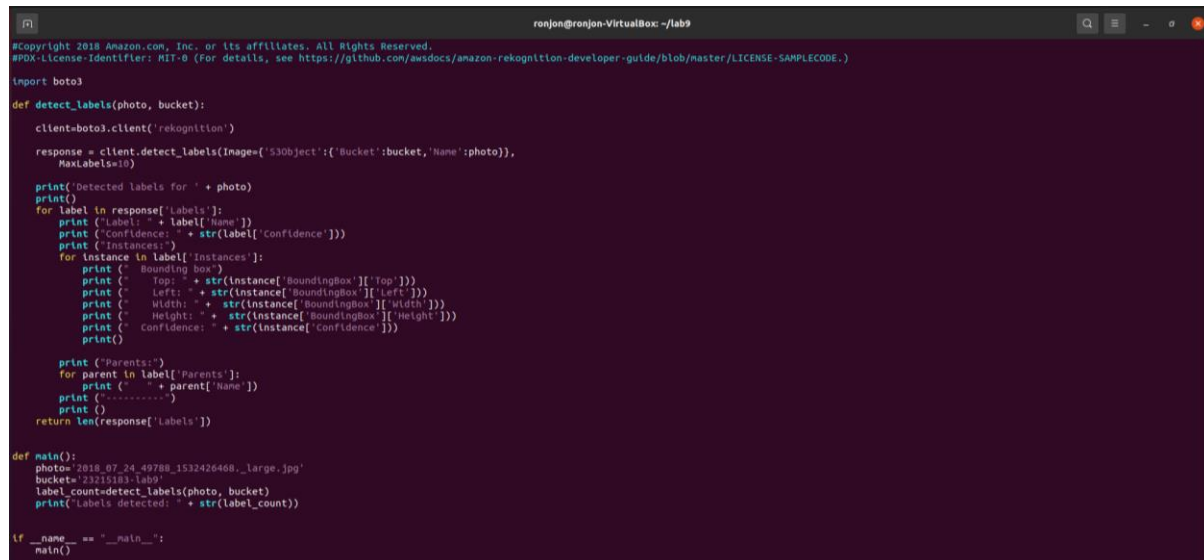


Fig 12: Code for label recognition

Use the picture



```
Detected Labels for 2024_01_24_0100_1000000000_01.jpg
Label: Grass
Confidence: 0.95077486110812
Instances:
Parents:
Plant
-----
Label: Plant
Confidence: 0.95077486110812
Instances:
Parents:
-----
Label: City
Confidence: 0.97321838170866
Instances:
Parents:
Urban
Building
-----
Label: Urban
Confidence: 0.97321838170866
Instances:
Parents:
-----
Label: Building
Confidence: 0.97321838170866
Instances:
Parents:
-----
Label: High Rise
Confidence: 0.9730077479007794
Instances:
Parents:
City
Urban
Building
-----
Label: Lawn
Confidence: 0.88801218361759
Instances:
Parents:
Grass
Plant
-----
Label: Downtown
Confidence: 0.8880075
Instances:
Parents:
City
Urban
Building
-----
Label: Park
Confidence: 0.253486280364
Instances:
Parents:
Lawn
Grass
Plant
-----
Label: Park(1)
Confidence: 0.1724115480138
```

Fig 13: output for label recognition

After modifying a python script for image moderation

```
ronjon@ronjon-VirtualBox: ~/lab9
import boto3

def moderate_image(photo, bucket):

    client=boto3.client('rekognition')

    response = client.detect_moderation_labels(Image={'S3Object':{'Bucket':bucket,'Name':photo}}
)

    print('Detected labels for ' + photo)
    for label in response['ModerationLabels']:
        print (label['Name'] + ' : ' + str(label['Confidence']))
        print (label['ParentName'])
    return len(response['ModerationLabels'])

def main():
    photo='asia-thailand-young-man-on-beach-arms-outstretched-RDF00652.jpg'
    bucket='23215183-lab9'
    label_count=moderate_image(photo, bucket)
    print("Labels detected: " + str(label_count))

if __name__ == "__main__":
    main()
```

Fig 14: Code for Image moderation

Using the picture



```
Detected labels for asia-thailand-young-man-on-beach-arms-outstretched-RDF00652.jpg
Suggestive : 86.2708969116211

Barechested Male : 86.2708969116211
Suggestive
Labels detected: 2
ronjon@ronjon-VirtualBox:~/lab9$
```

Fig 15: Output of image moderation

For facial analysis a python script is modified

```

renjon@renjon-VirtualBox: ~/lab9
import boto3
import json

def detect_faces(photo, bucket):

    client=boto3.client('rekognition')

    response = client.detect_faces(Image={'S3Object':{'Bucket':bucket,'Name':photo}},Attributes=['ALL'])

    print('Detected faces for ' + photo)
    for faceDetail in response['FaceDetails']:
        print('The detected face is between ' + str(faceDetail['AgeRange']['Low'])
              + ' and ' + str(faceDetail['AgeRange']['High']) + ' years old')

        print('Here are the other attributes:')
        print(json.dumps(faceDetail, indent=4, sort_keys=True))

        # Access predictions for individual face details and print them
        print('Gender: ' + str(faceDetail['Gender']))
        print('Smile: ' + str(faceDetail['Smile']))
        print('Eyeglasses: ' + str(faceDetail['Eyeglasses']))
        print('Emotions: ' + str(faceDetail['Emotions'][0]))

    return len(response['FaceDetails'])

def main():
    photo='people showing their faces.jpg'
    bucket='23215183-lab9'
    face_count=detect_faces(photo, bucket)
    print('Faces detected: ' + str(face_count))

if __name__ == "__main__":
    main()

```



Fig 16: Code and picture used for facial analysis

```

detection-VirtualBox>python detect_faces.py
detection-VirtualBox>python detect_faces.py
ected faces for people showing their faces. You
e detected face is between 36 and 44 years old
re are the other attributes:

'AgeRange': {
  'High': 44,
  'Low': 36,
},
'Beard': {
  'Confidence': 81.7656523203125,
  'Value': false,
},
'BoundingBox': {
  'Height': 0.2766589943962897,
  'Left': 0.4911177853393555,
  'Top': 0.3439351842239604,
  'Width': 0.1466162681111267,
},
'Confidence': 99.9781795314046,
'Emotions': [
  {
    'Confidence': 96.12842236328125,
    'Type': 'HAPPY',
  },
  {
    'Confidence': 7.114441189575195,
    'Type': 'SURPRISED',
  },
  {
    'Confidence': 5.992613435394287,
    'Type': 'FUD',
  },
  {
    'Confidence': 2.1876699924688994,
    'Type': 'SAD',
  },
  {
    'Confidence': 0.5438438369259654,
    'Type': 'CONFUSED',
  },
  {
    'Confidence': 0.485968979393115,
    'Type': 'ANGRY',
  },
  {
    'Confidence': 0.4045119349751282,
    'Type': 'DISGUSTED',
  },
  {
    'Confidence': 0.38797341016922,
    'Type': 'CALM',
  },
},
'Eyeglasses': {
  'Confidence': 95.8766786679688,
  'Value': false,
},
'Gender': {
  'Confidence': 95.8766786679688,
  'Value': 'Male',
},
'Landmarks': [
  {
    'Type': 'eyelLeft',
    'X': 0.5263119995292864,
    'Y': 0.3849546687679226,
  },
  {
    'Type': 'eyelRight',
    'X': 0.580288661256104,
    'Y': 0.27894433691833496,
  },
  {
    'Type': 'noseLeft',
    'X': 0.559465229511281,
    'Y': 0.385712623873893525,
  },
  {
    'Type': 'noseRight',
    'X': 0.6047527194823132,
    'Y': 0.35768766663131734,
  },
  {
    'Type': 'nose',
    'X': 0.5833824467638997,
    'Y': 0.3491680948582886,
  },
  {
    'Type': 'leftEyebrowLeft',
    'X': 0.501812467883955,
    'Y': 0.292387349582499,
  },
  {
    'Type': 'leftEyebrowRight',
    'X': 0.526447534615157,
    'Y': 0.2786543369293213,
  },
  {
    'Type': 'rightEyebrowLeft',
    'X': 0.5582521821864758,
    'Y': 0.2587853678182339,
  },
  {
    'Type': 'rightEyebrowRight',
    'X': 0.580288661256104,
    'Y': 0.3849546687679226,
  },
  {
    'Type': 'mouthLeft',
    'X': 0.4911177853393555,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'mouthRight',
    'X': 0.6047527194823132,
    'Y': 0.35768766663131734,
  },
  {
    'Type': 'mouth',
    'X': 0.5479359595959596,
    'Y': 0.35073171875,
  },
  {
    'Type': 'chinBottom',
    'X': 0.6023680080305118,
    'Y': 0.4759517233377346,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.638892550084248,
    'Y': 0.323558642182332,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.6203346252441486,
    'Y': 0.2263873742637417,
  },
  {
    'Type': 'chinTop',
    'X': 0.4911177853393555,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinBottom',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinTop',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinBottom',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinTop',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinBottom',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinTop',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinBottom',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinTop',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinBottom',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinTop',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinBottom',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawRight',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'chinTop',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'midJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.3439351842239604,
  },
  {
    'Type': 'upperJawLeft',
    'X': 0.3439351842239604,
    'Y': 0.34393518
```

Fig 17: output for facial analysis

To extract text from an image modify a python script

```
ronjon@ronjon-VirtualBox: ~/lab9
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_text(photo, bucket):
    client=boto3.client('rekognition')

    response=client.detect_text(Image={'S3Object':{'Bucket':bucket,'Name':photo}})

    textDetections=response['TextDetections']
    print ('Detected text\n-----')
    for text in textDetections:
        print ('Detected text:' + text['DetectedText'])
        print ('Confidence: ' + "{:.2f}".format(text['Confidence']) + "%")
        print ('Id: {}'.format(text['Id']))
        if 'ParentId' in text:
            print ('Parent Id: {}'.format(text['ParentId']))
        print ('Type: ' + text['Type'])
        print()
    return len(textDetections)

def main():
    bucket='23215183-lab9'
    photo='Example-IG-Caption.jpg'
    text_count=detect_text(photo,bucket)
    print("Text detected: " + str(text_count))

if __name__ == "__main__":
    main()
```



Fig 18: Code and picture used to extract text from an image

```
Detected text
-----
Detected text:COME SAIL AWAY
Confidence: 99.83%
Id: 0
Type:LINE

Detected text:COME
Confidence: 99.78%
Id: 1
Parent Id: 0
Type:WORD

Detected text:SAIL
Confidence: 100.00%
Id: 2
Parent Id: 0
Type:WORD

Detected text:AWAY
Confidence: 99.69%
Id: 3
Parent Id: 0
Type:WORD

Text detected: 4
ronjon@ronjon-VirtualBox:~/lab9$
```

Fig 19: output of extract text from an image

After completing all tasks in the lab delete the bucket

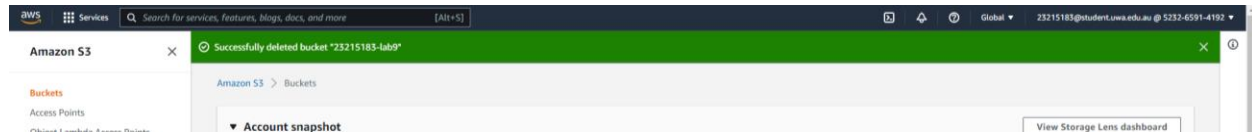


Fig 20: Delete bucket