Week 1 → Computer Vision

Outline

- Amzn recognition → CV
- Amzn textract → data extraction
- Amzn Comprehend → lang processing

Amazon Rekognition

↳ Fully managed, deep-learning-based recognition service
↳ Can detect things, or labels, in an image
↳ uses Pre-trained models → APIs that are called

Image API operations
→ CompareFaces
→ DetectFaces
→ DetectLabels → detects objects, events, concepts
→ DetectModerationLabels → detects content to exclude
→ DetectProtectiveEquipment → face, head, hand covers
→ DetectText
→ RecognizeCelebrities

Video API operations
→ GetCelebrityRecognition
→ GetContentModeration
→ GetFaceDetection
→ GetFaceSearch
→ GetLabelDetection
→ GetPersonTracking
→ GetSegmentDetection
→ GetTextDetection

APIs that can work with collections of faces:

IndexFaces → stores faces/images into a collection
SearchFaces → checks collection for like face (faceID)
SearchFacesByImage → Checks collection using image

Can train rekognition using own labeled datasets

CreateDataset → used to train

DetectCustomLabels → used to predict/test

Exercise 1 - Working with Amazon Rekognition

↳ using AMZN Rekognition to improve search results for an imgs app

↳ relies on build/data.json to populate an img search

T1: Creating an AWS Cloud9 environment

→ Services → Cloud9 → create env → name → create

Terminal → Pip install boto3

→ download & extract exercise src files

→ cd exercise-Rekognition

→ Python3 "build_json.py"

↳ open build_json.py → vim ____

→ ~~file~~ loops thru all files in public/Photos/*.jpeg

Code → reads all bytes of img file

→ sends data to DetectLabels API

To Run Application:

Python3 build_json.py > build/data.json → Pipes data to where web app expects it

In new terminal → go to exercise folder

→ Python3 -m http.server 8080 -d build → Runs on 8080

→ preview it

→ replace code

entry["Labels"] = response["Labels"]

↳ populates the Labels key with the response from AMZN Rekognition

↳ Can now search based on recognized labels from each pic

Amazon Textract

↳ Fully managed ML service that analyzes documents and automatically extracts text & structure

API operations

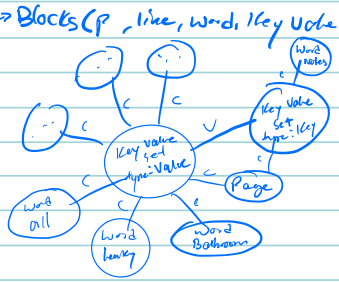
→ Analyze Document / Start Document analysis / get Document Analysis
(Start → begins process, returns job ID; get → retrieves status & results of a job)

↳ Takes feature types we are interested in:

API Interesting

Forms

↳ returns DS in JSON
↳ Blocks (P, line, word, key value set)



Queries

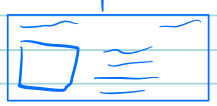
Request:
"Text": "What are the notes" → Nat long Request
"Alias": "Notes"

Response:
"BlockType": "Query_Result",
"Confidence": 95.0,
"Text": "Leaky Bathroom tap. otherwise all good"

→ Analyze Expense / Start Expense Analysis / get Expense Analysis

→ used to extract data from receipts/invoices

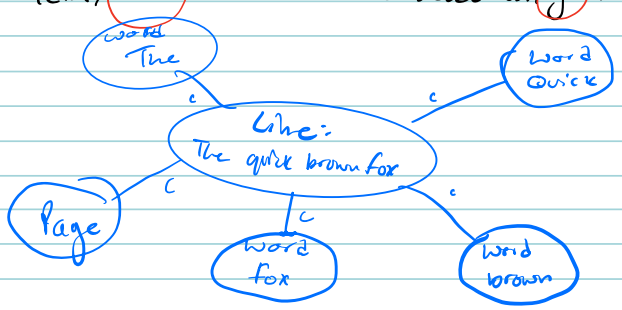
→ Analyze ID



[
{
fname
lname
etc.

→ Detect Document Text / Start Document Text Detection / get Doc Text Detection

ex.
The quick brown fox



Exercise 2: Working with AMZN Textract

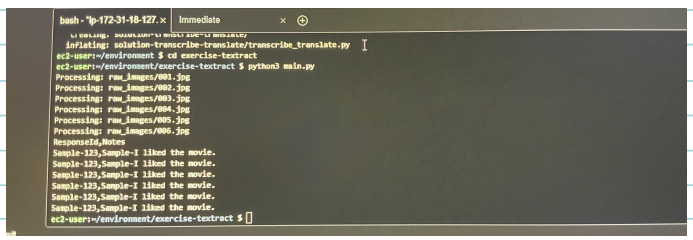
↳ Using Amazon textract to extract text from handwritten movie-review cards

Task 1: Install AWS SDK for python (boto3) & extract exercise files

↳ And Run the Ap

1. getting files
   - Pip install boto3
   - wget ------
   - unzip exercise-source.zip

2. running the App
   - CD exercise-textract
   - python3 main.py



→ processes raw images
→ displays Comma-Separated Values (CSV) output with placeholder data

Task 2: updating the application

↳ Completing the missing code in 'main.py'

↳ getting the extracted text from AMZN Textract instead of the Samples

```
1  import glob
2  import boto3
3  import json
4  import csv
5  import sys
6
7  csv_array = []
8  client = boto3.client('textract')
9  for filename in glob.glob('raw_images/*.jpg'):
10     csv_row = {}
11     print(f"Processing: {filename}")
12     with open(filename, 'rb') as fd:
13         file_bytes = fd.read()
14
15     response = client.analyze_document(
16         Document={'Bytes': file_bytes},
17         FeatureTypes=["QUERIES"],
18         QueriesConfig={
19             'Queries': [
20                 {'Text': 'What is the response id', 'Alias': 'ResponseId'},
21                 {'Text': 'What are the notes?', 'Alias': 'Notes'},
22             ]
23         }
24     )
25
26     # uncomment this to see the format of the reponse
27     print(json.dumps(response, indent=2))
28
29     #####
30     # Replace this code with a solution to populate a dictionary with the results from textract
31     #####
32     for block in response["Blocks"]:
33         if block["BlockType"] == "QUERY":
34             query_alias = block["Query"]["Alias"]
35             answer_id = next(rel["Ids"] for rel in block["Relationships"] if rel['Type'] == "ANSWER")[0]
36             answer_text = next(b for b in response["Blocks"] if b["Id"] == answer_id)["Text"]
37             csv_row[query_alias] = answer_text
38     csv_array.append(csv_row)
39
40 writer = csv.DictWriter(sys.stdout, fieldnames=["ResponseId", "Notes"], dialect='excel')
41 writer.writeheader()
42 for row in csv_array:
43     writer.writerow(row)
44
45
```

1. libraries
2. → initialize array → stores dicts that contain extracted data from each processed image
3. → calls AWS Textract → assigns to client object
4. → loop thru images
5. → stores extracted data for that image (dictionary)
6. → print current image being processed
7. → opens and reads img file into 'file_bytes'
8. → calls analyze-document API
9. → Call Textract to analyze the document
10. → sends img bytes to textract
11. → specifies to use Queries feature type
12. → defines 2 Queries
13. → response id
14. → Notes'
15. → shows JSON data (to see if extracted correctly)
16. → Task
17. Added Code
18. → iterates thru the blocks in the textract response
19. → looks for Queries
20. → extract Alias of the Query (response id, Notes)
21. → finds the Id of the corresponding answer block
22. → Next() → iterates over the relationships looking for Type: "ANSWER" and retrieves the first ID
23. → store extracted answer text in dictionary under corresponding Query alias
24. → uses answer Id to find corresponding block that contains text
25. Quickly Add to the list
26. → Writes & outputs each data row

Amazon Comprehend

↳ Natural language processing service that uses ML to uncover valuable insights & connections in text

Detect-Dominant-Language APIs → real time analysis
↳ detects what language the text is in
Batch-detect-Dominant-language → real time analysis
↳ same but for a batch of documents (up to 25 docs/call)
Start-dominant-language-detection-job → asynch batch processing

Detect-entities APIs → detecting named entities
↳ real time, single docs
Batch-detect-entities
↳ real time, batch of docs
Start-entities-detection-job
↳ asynch, batch of docs

ex. Text: I am John from Company Inc

Result: Type: Person
Text: John
Type: organization
Text: Company Inc

Detect-Key-Phrases APIs
Batch
Start-K-P-d-Job

Detecting Personally Identifying Information Entities

Detect-pii-entities

describe-pii-entities-detection-job

Start-pii-entities-detection-Job

Contains-Pii-entities

ex. Text: Hello John, your address is 123 Test St.

Result: Entities(
Type: Name
Type: Address

Detect-Sentiment → takes doc and detects sentiment
batch-detect-Sentiment
Start-Sentiment-detection-job

ex. Text: "I Loved shopping here"

ex. Result: Sentiment: Positive

*Custom models (not trained by amazon)

Custom Classification

Create-document-Classifier → Classifies documents
↳ basically, you have to set up the models → create labels
↳ supply data for learning

Label	Document
Comedy	
Tragedy	

↓
Model

Custom entity recognition

Create-entity-recognizer API
↳ used to supply label training data to create a model

documents
Martha is an engineer.
John is an engineer for 15 yrs
Saanvi has been the manager for 4 years → Model

Entity list

Label	Type
Martha	Engineer
John	Engineer
Saanvi	Manager

↓
Model

Custom Models

Create-entity-recognizer
Create-document-Classifier
↓
Model → (Create-endpoint) → Endpoint

↑
Asynch API ops
↳ Start-document-Classification-Job
↳ Start-entities-detection-Job

↑
Real time API Ops
↳ Classify-document
↳ detect-entities

Exercise 3: Working with AMZN Comprehend

↳ Use AMZN Comprehend to process the movie-feedback CSVs file to retrieve the Sentiment for each movie review

Task 1: Running the App

1. getting files
   - Pip install boto3
   - wget ———————
   - unzip exercise-source.zip

2. running the App
   - CD exercise-textract
   - python3 main.py

→ You get 2 errors
↳ "Missing required parameter in input"

Task 2: updating the app

↳ Update the call to batch_detect_Sentiment w/ all required Params

```
Welcome | main.py | main.py

1  import boto3
2  import csv
3
4  # read the movies CSV and populate the all_notes rray with all the of notes
5  with open("movies.csv", 'r') as fd:
6      reader = csv.DictReader(fd, fieldnames=["ResponseId", "Notes"], dialect='excel')
7      all_notes = [ row["Notes"] for row in reader ]
8
9  client = boto3.client('comprehend')
10
11 #####
12 # Complete the call to batch_detect_sentiment
13 #####
14 response = client.batch_detect_sentiment(
15     TextList=all_notes,
16     LanguageCode='en'
17 )
18
19
20 for result in response["ResultList"]:
21     index = result["Index"]
22     sentiment = result["Sentiment"]
23
24     print(sentiment, all_notes[index])
```

reading

both are required, regardless of Problem for Sentiment

outputs Sentiments for each

output

```
bash - "ip-172-31-18-127.x | Immediate | exercise-textract/main.py

    return self._make_api_call(operation_name, kwargs)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/botocore/client.py", line 974, in _make_api_call
    request_dict = self._convert_to_request_dict(
  File "/home/ec2-user/.local/lib/python3.9/site-packages/botocore/client.py", line 1048, in _convert_to_request_dict
    request_dict = self._serializer.serialize_to_request(
  File "/home/ec2-user/.local/lib/python3.9/site-packages/botocore/validate.py", line 381, in serialize_to_request
    raise ParamValidationError(report=report.generate_report())
botocore.exceptions.ParamValidationError: Parameter validation failed:
Missing required parameter in input: "TextList"
Missing required parameter in input: "LanguageCode"
ec2-user:~/environment/exercise-comprehend $ python3 main.py
  File "/home/ec2-user/environment/exercise-comprehend/main.py", line 16
    LanguageCode='en
                 ^
SyntaxError: EOL while scanning string literal
ec2-user:~/environment/exercise-comprehend $ python3 main.py
POSITIVE I liked every thing about it!
POSITIVE I want to watch the movie again with my daughter Was very entertained.
POSITIVE I liked everything from the first scene!
POSITIVE I liked the characters, and found the story very engaging
POSITIVE the movie was great - I would recommend to all my Friends
NEGATIVE I Found the movie boring.
ec2-user:~/environment/exercise-comprehend $
```

Quiz: Week 1

1. Which statement is true for all uses of machine learning? 1 point

- Uses algorithms to learn and improve from training data.
- (•) Uses multilayer networks to build models that are inspired by the human brain
- Uses algorithms that are suited to only one purpose
- Requires a human to perform inferences or estimates from a trained model

2. A software engineer wants to process a collection of images to find all the images that contain a copyrighted logo. Which Amazon Rekognition API operation would help with this project? 1 point

- DetectLabels
- DetectModerationLabels
- (•) DetectCustomLabels
- DetectFaces

3. A software engineering team was supplied with scans of store receipts. The team wants populate a database with all the details of the line items that appear on the receipts. Which Amazon Textract API operation would be best suited to this task? 1 point

- AnalyzeDocument
- (•) AnalyzeExpense
- AnalyzeID
- GetDocumentText

4. A software engineer needs to determine the dominant language of a large set of documents. The documents are currently stored in an Amazon Simple Storage Service (Amazon S3) bucket. Which steps should the software engineer follow to determine the dominant language of each document? 1 point

- (•) Write a script to load the contents of each file, and use an Amazon Comprehend API operation to determine the dominant language.
- Write a script to use the Amazon Comprehend API, and pass the location of each file in Amazon S3 as a parameter.
- Write a script to load the contents of each file, and use an Amazon Translate API operation to determine the dominant language.
- Write a script to use the Amazon Translate API, and pass the location of each file in Amazon S3 as a parameter.