Matroid full-stack challenge (backend)

Part of Matroid's functionality is searching video streams for specific faces. For example, if you use a famous actors detector, you could find out the timestamps when Angelina Jolie appears in the video by calling Matroid's API. The `label_dict` field maps between numbers and a particular actor from the detector. The `detection` field has as its key the timestamp where something was detected, then an array of detections containing the bounding boxes of where something was detected (represented as a ratio of the image between 0.0000 and 1.0000) and the confidence scores of the detection (between 0.00 and 1.00).

For the example JSON below, Angelina Jolie was detected at the timestamp of 89 seconds at a bounding box with a top left corner 23.77% from the left side and 20.21% from the top of the image and a height of 38.96% and width of 16.28% of the image. The confidence for this prediction was 0.95 out of 1.00, indicating that the detector strongly believes the face in this bounding box is Angelina.

```
{
  "download_progress": 100,
  "classification_progress": 8,
  "status": "Video Download Complete. Classifying Video",
  "label_dict": {"0":"Angelina Jolie","1":"Brad Pitt"},
  "state": "running",
  "detections": {
    "89": [
      {
        "labels": {
          "0": .95
        },
        "bbox": {
         "left": 0.2377,
         "top": 0.2021,
         "width": 0.1628,
         "height": 0.3896
       }
      }
    ],
    "92": [
      {
        "labels": {
          "0": .16,
          "2": .8
        },
        "bbox": {
          "left": 0.7576,
          "top": 0.2375,
          "width": 0.0597,
          "height": 0.1313
        }
      },
      {
        "labels": {
          "0": .87,
        },
        "bbox": {
          "left": 0.5047,
          "top": 0.1708,
          "width": 0.055,
          "height": 0.1292
        }
      },
    ]
  }
}
```

However, users may be interested in not just the specific points when Angelina appears but the ranges of times where she's in the video. Since 89 and 92 are "close together" in some sense, you could say that Angelina is in the video from the time period 89 seconds to 92 seconds.

Your challenge is to create an HTTP API which will take in a video id corresponding to a video that Matroid has already started classifying, and instead of returning the timestamps and scores for each person, return the ranges of time in the video where the person of interest appears in the following JSON format:

```
'[{ "person": "angelina jolie", "scenes": ["0-90.5", "534.5-659",
"777-1024"] }, { "person": "brad pitt", "scenes": [] }, { "person":
"kerry washington", "scenes": ["30-90.5", "500-600"] }]'
```

Come up with a simple and reasonable metric of what constitutes a continuous scene and document it along with your assumptions in a file README.md. A scene should represent a mostly continuous period of time where a person is visible. Scenes should be at least 2 seconds long, but it's up to you to decide which scores are high enough and how far apart each entry can be before it's not considered a continuous scene anymore.

The sample actors detector for testing has the following labels:

- Adam Brody
- Angelina Jolie
- Brad Pitt
- Kerry Washington
- Michelle Monaghan
- Vince Vaughn

Your solution should work for all facial recognition detectors.

Feature requirements

Core requirements and specifications

- We will set you up with a Matroid account that has as many credits as you need for testing. For completing the assignment, we will give you an additional $100 worth of Matroid credits to play around with.
- Document the HTTP verb and resource inside the README for your endpoint
- It's up to you whether or not you want to include people that have no scenes in the response.
- Your endpoint only needs to accept Matroid video ids, not URLs or videos. Please see the Matroid documentation on how to classify videos in the Overview section of the Call API tab. You can assume the user already began classifying the video and a video ID exists. To manually test the app you will probably have to classify videos, but we will provide some video IDs that return results.
    - Sample video to use for testing.
        - Sample Video ID: 5a3ec65b37939b0013123535
        - Detector ID: 5a3ebf485162a700138fa669
- Because the video classification process is asynchronous, your endpoint will have to compute the scenes based on the available data. Users should be able to query your API and find out the current scenes even if the video is not done classifying.
    - Make sure your scene definition still makes sense in this scenario or if there are further simplifications you'd have to do if the result is not complete
- Submissions using Node.js with at least version 8.0.0 are strongly preferred, but submissions in other languages will be considered. If you use Node.js, you can use the Matroid API's Node.js client within your app: https://github.com/matroid/matroid-node
    - It's OK to use a framework such as Express.js or Koa.js for your server code
    - Feel free to use transpiling tools such as babel or TypeScript
- Make sure to return reasonable errors upon bad input, classification failures, etc. The video results endpoint can return a variety of errors; you don't have to do something different for each one, but your API should at least return a response instead of crashing. To see an example error message format from our API, try passing in a bogus video ID.
- **Implement at least two unit tests.**

- You don't need to put any authentication on your API endpoint
- You will have **3 days** to complete the assignment. It should take 3-6 hours to complete this, with roughly 1/3rd of the time on implementation, 1/3rd of the time on planning and documentation, and 1/3rd of the time on QA and testing.

Evaluation

- Code style: your code should use standard Javascript conventions and indentation. Variables and functions should have appropriate and descriptive names. When in doubt, consult the Airbnb Javascript style guide.
- JavaScript fluency: concise, modular, readable, documented, modern code is preferred
- Appropriate handling of errors, edge cases
- Existence of two unit tests and a command to run the test suite
- Reasonable definition of scene and a correct implementation of your specification
- Discussion in README.md of ways to improve your app
- Extra features are fine, but for fairness reasons, you will only be evaluated on your implementation of the minimum specifications and what you write in README.md.
- Do not post your code in a publicly accessible repository like Github (unless it's private)

Submission

- Email us a Google Drive or Dropbox link with a .zip file containing all the required code**.**
- Please include a bash script that we can run to install your application on a fresh Ubuntu 16.04 machine. Assume that Node.js and other dependencies are not installed. You can test your installation process on a fresh Ubuntu virtual machine.

Have fun working on the app! Afterward, feel free to make your own detector on Matroid with your favorite actors and find their scenes in videos.