Social Media Image Analysis with Google Cloud Vision: Final Report

Paine Leffler, Jiaming Zhao, and Eleanor Leffler

***Abstract*— This is the final project report for Fundamentals of Digital Archeology. The report will provide the motivation of the project, a detailed discussion of the context and data, a description of the quantitative method used to analyze data, a description of the results, related work, limitations, issues encountered, and future work. For this project, we analyzed images extracted from Twitter and Instagram using the Twitter API and an Instagram scrapper. We used the Google Cloud Vision API to generate the labels of the images. To increase efficiency, we stored the Google Cloud Vision API objects with the username, URL, platform name, and date. Lastly, we used the Node JS API and data visualization tools to build the web application that presents the results.**

I. INTRODUCTION

Social media has revolutionized the way information is shared on the Internet. Because sharing is so quick and easy, pictures and personal information have become abundantly available. Additionally, online presences have become much more reflective in personality of the individual. However, judging an individual by his or her social media can be biased or incorrect.

As much as individuals do not want to be judged by the pictures or data about them online, many people still do. Also, an image in memory can seem to be meaningless to humans, but with further analysis, there are hidden patterns that can provide clues to what the images contain. The main goal of this project will be to analyze that information and provide labels for a specific individual.

II. MOTIVATION

Individuals are publishing less textual information and more pictorial information by taking hundreds of pictures every week and posting them to some sort of social media. While humans can easily recognize attributes of pictures, analyzing hundreds or thousands of pictures becomes time consuming.

Utilizing Google Cloud Vision, this project allows users to quickly understand how their social media might be perceived by outside observers. For example, employers often use social media to judge the behavior of a candidate. This project gives candidates a tool to determine how an employer might perceive him or her. This project will help people understand how their public images might affect their reputation.

III. CONTEXT AND DATA

Our primary data sources were Instagram and Twitter. Using the Facebook API was more complicated than we expected, so we did not have the Facebook component working in time. We used the Twitter API and an Instagram scrapper to retrieve image URLs from the social media sources so that we pass the URLs into Google Cloud Vision's API.


```
"created_at": "Fri Jul 29 16:16:33 +0000
    2016",
    "id": 759060050589528065,
    "id_str": "759060050589528065",
    "text": "Wow\n Such song",
    "truncated": false,
    "entities": {
        "hashtags": [],
        "symbols": [],
        "user_mentions": [],
        "urls": [],
        "media": [{
            "id": 759060043283050496,
            "id_str":
                "759060043283050496",
            "indices": [27, 50],
            "media_url":
                "http:......jpg",
            "media_url_https":
                "https:......jpg",
            ...
            }]
    }
```


Fig. 1. Example of Twitter Response Label Object


```
[{
   "id": 759060050589528065,
   "caption": "progress pic #gains
       #muscle #crossfit #vegan"
   "shortcode": D3PQN10A,
   "image": [{ ... }],
   "dimensions": {...}
     "likes": 5
     "comments": {...}
     "owner": {...}
     "timestamp":
         "2017-11-28T21:31:53.413Z"
}, {...} ... ]
```


Fig. 2. Example of Instagram Object

The structures of the objects that the Twitter API and the Instagram scrapper returned are shown above in Figures 1 and 2.

Google Cloud Vision's API returns an object shown in 3. We stored the results of the Google Cloud Vision response in

```
{
   "results": [
     "labelAnnotations": [
       {
         "mid": "/m/01ctsf",
         "locale": "",
         "description": "atmosphere",
         "score": 0.7913299798965454,
         "confidence": 0,
         "topicality": 0,
         "boundingPoly": null,
         "locations": [],
         "properties": []
       },
      ...
      ]
}
```


Fig. 3. Example of Vision Response Label Object from SpaceX


```
{
   "Username": paineleffler
   "Url": https://...jpg
   "Platform": 'Twitter'
   "Results": [{ ... }]
   "Date":  "$date":
       "2017-11-28T21:31:53.413Z"
}
```


Fig. 4. Example of Database Object

MongoDB on MLAB. This was done to save API requests and speed up user analysis. Since we were not concerned about space, we kept the entire Google response so that analysis on other properties such as facial coordinates, image location, and related images could potentially be used in later analysis. Figure 4 shows an example of the database object.

At the moment of writing this report, we have data for 103 unique users, 2593 unique images, and 1749 unique labels. To provide these analytics in our application, we have a specific object structure in MongoDB. In each database object, we identify the image URL, the username, the social media platform, the date retrieved, and the Google Vision results object. To provide the number of unique labels, a route was created to build a set from all of the entries in the result objects. The set size was then returned to provide a proper count.

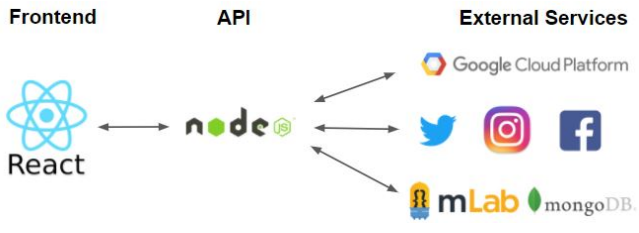

Fig. 5. Project Architecture

Figure 5 summarizes the process of abstracting and analyzing the data and displays the overall architecture of the project.

IV. QUANTITATIVE METHODS

All labels from Google Cloud Vision stored in MongoDB. However, not of the all labels were sent with the response to our application. We decided to strip the uninteresting or vague words so that we could only analyze more unique labels that distinguished people. The labels that were blacklisted were labels that gave a high-level attribute or description of the image. These blacklisted labels included words like: ”product”, ”profession”, ”photograph”, ”fun”, ”material”, ”number”, ”line”, and ”color”, while the interesting labels included words like: ”guitar”, ”kayak”, ”alcoholic beverage”, ”football”, ”hockey”, ”coffee”, ”corgi”, and ”canyon”. The uninteresting words were determined after large amounts of trial and error with image analysis. Typically, these were seen in very high quantities across images from many users. These labels are a result of high level attributes given during the training of the Google Vision API.

To compare two users, we gathered all of their data across all social media platforms and compared only the labels in common. To do this, two MongoDB requests had to be made. Then only after both had returned data, the first set was used to check against the other set. Any labels from the first set that were not contained in the second set were removed. This was done because the only data labels when comparing are the ones in common.

V. RESULTS

This section contains examples of the data and output from the web application built. All of the charts are generated using an open source package called Chart JS. This allows to dynamically show interactive data sets in the browser. Each label can hovered over to discover exact values. This is valuable in the case of large data sets where the bar graphs have very narrow bars.

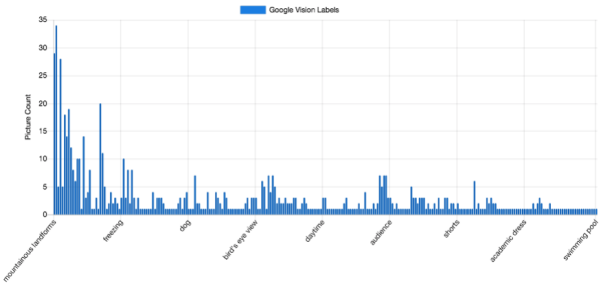

Fig. 6. Paine’s Individual Labels from Instagram

In Figure 6, all of Paine’s Instagram labels are shown. These are the labels of the 50 most recent images that have Vision Scores above 0.8. It also excludes the uninteresting words that are filtered by the API Wrapper. As you can see, each count reflects an image that the label was given too. Since each image can have any number of labels, the best way to show this data was a bar chart.

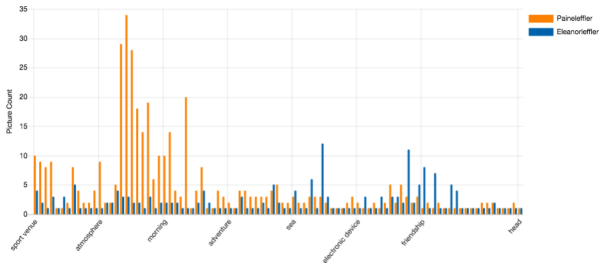

Fig. 7. Paine and Eleanor Comparison

It is easy to see in Figure 7 that even though Eleanor and Paine are siblings, they have a large difference in the quantity of labels that are in common.

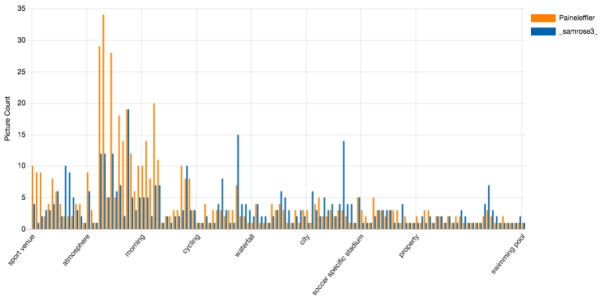

Fig. 8. Paine and Sam Comparison

It is clear to see in Figure 8, Paine and Sam have a large amount of overlap with labels that pertain to mountains, outdoors, nature, and landscapes. This is an accurate assumption because Paine and Sam go hiking and post many pictures from these locations.

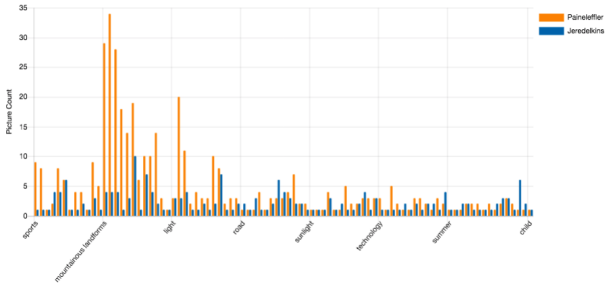

Fig. 9. Paine and Jered Comparison

Figure 9 above is very interesting because not only is there large amounts of overlap, there are few labels that have different label quantities.

However, in Figure 10, there is more overlap showing that these two people share more labels, and thus they probably are more similar in personality and share more interests.

VI. LIMITATIONS

As far as social media content goes, the data was limited to public users only. This kept the application workflow relatively simple and easy to use. Most people would probably not authenticate with an application they do not trust, so in the end, this made the most sense.

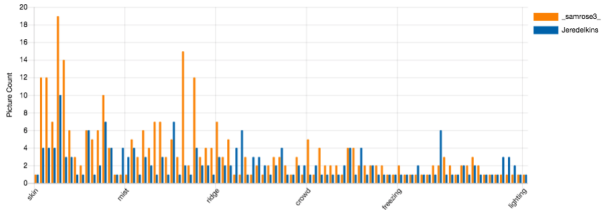

Fig. 10. Jered and Sam Comparison

With the Instagram scraper, we were unable to get ”carousel” images where the images are formatted as a slider. This is because of how Instagram’s React JS was adjusted through an update. This means we are only able to retrieve flat images from a user’s profile.

With Twitter’s API, we were limited in the way we requested tweets. In the structure of their API, it is possible to make requests for a number of a user’s most recent posts. This was limiting in a way because specifying a hard number of tweets is different time periods for different users. Also, this method did not allow us to access a specific number of images. We could only check the number of most recent tweets for media URLs.

Facebook’s Graph API was also a limitation on our data. This was discussed above in issues encountered. To recap, with the way Facebook requires authentication of applications, we were unable to access user’s albums of profile images.

VII. ISSUES ENCOUNTERED

One of the biggest issues that arose was getting images from Instagram. Instagram is very protective of their API keys. To circumnavigate this, an Instagram UI scraper was used. This way, any public Instagram users could have their image URLs retrieved. A number of Instagram scrapers were integrated and tested, however, most were outdated and filled with runtime errors. Another problem with finding and open source solution was that original authors rarely came back to merge in user pull requests or fix these bugs. Eventually, one scraper returned the correct formatted URL that Google had access too.

Also, synchronous data and asynchronous analysis was something we had to always deal with. Retrieving data from MongoDB and analyzing images with the Google Vision API have much different run times. With the way that data was displayed, this needed to be taken into account. Loading SVG’s were used and each results request reported back to the application state that it was done so that it could be tracked.

Another issue that we encountered was authentication with the Facebook Graph API. In order to access the profile picture album of a specific user, the user needed to authenticate our application with their credentials. This severely limited us with the Facebook Graph API. We were only able to search for public users and were unable to pull profile

image URLs. Also, Facebook ended their API username search. Therefore we could not search for users based on a unique username. This made it challenging to search for users because names are not unique identifiers. Another issue with this specific workflow was that we had to search for a name, figure out the correct person, get their user id number, then make specific requests with that id number. This made the Facebook Graph API very difficult to work with.

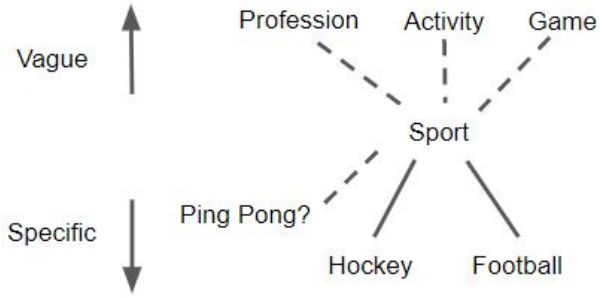

Fig. 11. Categorizing Labels

There are additional issues with classification of labels. We experienced a wide variety of results from different APIs. Many of these were limited to free requests or they were inconsistent. For the purposes of this project, we wanted clean and concise topics that users could be given percentages of images that relate to these topics. This way regardless of image count, users can be compared. The biggest issue we discovered with this was the level of detail as shown in Figure 11. Where do you draw the line for how vague a category should be? Another issue that arose was how opinions could alter the category. There was some human input or judgment needed for some categories.

VIII. FUTURE WORK

There are many uses and applications for this technology. Computer vision technology can be used for job recruiting, accessibility plug-ins, team selections, marketing analysis, and even suggestion algorithms. Suggestion algorithms could take images posted by users and suggest other potential friends, pages related to interests, or even other posts and videos.

A future extension of this application could be analyzing GIFs or videos from YouTube. The application could access the video and split the content into a certain number of frames. With these frames, the application could process them using Google's API and give a general idea of what proportion of the video or GIF contains certain labels.

Another possible feature would be to allow users to authenticate with their social media accounts to give full access of personal images to the application. By doing this, we can provide a full analysis of all users, public or private. This way the application can become a personal tool to analyze what content is being portrayed to the audience on social media.

Other configurations on the application would make it more applicable to bigger friend groups. For example, on the comparison page, we could allow adding more than just two people to compare. This could easily be integrated with Chart JS's data set structure. This way, small friend groups can see what they all have in common.

This leads to potentially restructuring the data as a graph data base. This would especially be interesting in finding associations and groups among similar people and similar labels.

IX. RELATED WORK

Computer vision is a popular machine learning topic. There have been many projects related to this idea in many different fields. Some relate to translations of signs or text to different languages. Some are even to place emoji smiley faces on people's faces in images. There are some vision projects out there that try to determine how old a user is from an image submitted. The applications for this idea are endless. However, from our research, we were unable to find anyone doing the analysis we were doing with social media images. This shows an enormous qualitative value in this type of image analysis.

X. CONCLUSIONS

In conclusion, this computer vision technology is very remarkable. However, it is not perfect yet. Through testing, it was discovered that there are many cases where images reveal only high level results instead of specific information. The speed in which the human brain recognizes a face or image is extremely fast. Humans can associate faces and images without the need of much context.

With computer vision, this is much different. Sometimes it is not very clear what is desired or what the target of the vision should be. For example, when an image of Antelope Canyon was submitted to the Google Vision API, results such as 'desktop background' comes back. This is an interesting training error because this image is typically used as a desktop background and is one of the default images for Mac OS backgrounds. This is where human input is required to favor more specific labels that don't pertain to web contexts.

There were more examples of this type of training error seen in a large amount of slightly different labels for one specific object. In an image of a pool table, many different labels came up: "Billiard table, pool, English Billiards, snooker, cue sports". All of these labels are describing different nouns that are almost the identical.

A nice feature would be to return hierarchal data sets from broad to specific with the respective confidence levels. For example, this could help represent that the computer is 100 percent confident it is a table for Cue Sports, but it is only 70 percent confident that it is an Billiards table, and only 30 percent confident that it is a snooker table. By structuring data this way, the end user can set a specification for how specific or how vague they want the response labels to be.

Having widely different label detail levels makes it difficult to accurately compare individuals. For example if you had two users: user 1 has a label of "outdoors" and user 2 has a label of "kayaking". Kayaking is outdoors, but how

can you compare the two? This makes classification very difficult.

Other observations we noticed was that Instagram tended to have more activities and contain more personal and accurate information. However, Twitter was a little different. Because of Re-Tweeting, many images in tweets were not the searched users images. This makes Twitter's platform more difficult to find accurate information. Because of this same reason, more labels such as: advertising, products, and text came up. These types of labels were much less prevalent on Instagram.

Twitter was also discovered to be much less active with our current generation. The 50 most recent tweets could have contained many from several years ago.

After analyzing Instagram's results we discovered many users were posting pictures much more frequently. Typically, Instagram pictures are taken by the users. This vastly improves the relevancy of the pictorial information associated with the user.

REFERENCES

[1] https://cloud.google.com/vision/docs/apis
[2] https://developer.twitter.com/en/docs
[3] https://developers.facebook.com/docs/javascript
[4] https://github.com/ivolimasilva/public-instagram
[5] https://github.com/jerairrest/react-chartjs-2
[6] https://github.com/paineleffler/vision-api
[7] https://github.com/paineleffler/vision-ui