Amara Auguste, CISC 4900, Fall 2019

A. **HEADER INFORMATION**

   a. **Name**: Amara Auguste
      CUNYFirst ID: 15321452
      Address: 500 Herzl St. Apt 2E, Brooklyn, NY, 11212
      Phone Number: 646-500-2426
      Date: December 16, 2019

   b. **Course Number**: CISC 4900
      Semester: Fall 2019

   c. **Agency/College Office**: BC CIS
      Supervisor Name: Michael Mandel
      Supervisor Phone/Email: mim@sci.brooklyn.cuny.edu
      Address: 2900 Bedford Ave, Ingersoll 2232

B. N/A

C. **Relationship to Supervisor**: I have had a professional education relationship to my supervisor, Professor Michael Mandel, who was my professor for CISC 1600 - Introduction to Multimedia and CISC 3620 - Computer Graphics.

D. **COMPLETE DESCRIPTION OF PROJECT & TASKS ACCOMPLISHED**

   a. **Introduction**: For this project, I have decided to work with Professor Mandel on his project evaluating various image recognition APIs to analyze several images from a trail camera from Alaska to answer the question of whether or not those images contain caribou. In response to this, I have analyzed these images myself to manually answer the question of whether they contain caribou or not, used various image recognition APIs to analyze the same images, and created a java program to evaluate the accuracy of each API in comparison to both my manual data and the other APIs.

   b. **Description of the Problem or Existing System**: Camera Trap Image Classification: Evaluating various image recognition APIs (Google, Microsoft, AWS Rekognition, Clarifai, etc.) on how well they can identify items in images from a trail camera (in this case, determine whether or not the images contain caribou or not) and compare results based on accuracy of identification, to report back the varying levels of accuracy across the various APIs and determine the most and least accurate.

   c. **Description of the Solution/Enhancements**: Using Gaurav Oberoi's open source labeling tool: Cloudy Vision (found here: https://github.com/goberoi/cloudy_vision) to obtain identification tags for each image, I have created a java program that stores that data (tags, for each image from a list of several images, received from each respective image recognition

API) and compares those tags to similar data from my own manual evaluation of the same images in order to compute accuracy from each individual API in comparison to my own data, compare the results amongst APIs, and display those results.

d. **Scope of the Work Performed**:
   - Manually evaluate each test image (290) and provide a series of tags summarizing what a human (in this case, more specifically me) would identify from each image, place in google sheets to log.
   - Study the basics of Python, in order to use Gaurav Oberoi's Cloudy Vision to obtain tags from each test API. Then, setup and run (had to make a few modifications due to outdated code) Cloudy Vision and obtain .html page with each API's tags for each image and log tags for each image into google sheets, and organize by API.
   - Write java code: 1) API class to store data (image name and corresponding identification tags) for each individual API tested, 2) ImageRecognitionAPIs class to search through data provided from each API, create an array of API objects, compare each set of tags with my manual data per image for word matches to determine accuracy, rank APIs based on accuracy, calculate average number of tags provided per image from each evaluated API, and display that data via text and charts printed to an html page.

e. **Summary**: Overall, in order to evaluate the various image recognition APIs, I have created a java program that reads in the data (image name and list of tags) from each API (after the user has placed the data in the 'vendors' folder in the format "API_NAME.txt" and calculates the accuracy of the tags based off of the comparison to the user's manual tags (saved in manualResults.txt), ranks all evaluated APIs, and computes the average number of tags per image from each evaluated API, displaying the results to an output.html file. My initial hypothesis was that the less tags that an API provides, the higher the accuracy would be due to more concise tags and matches. However, despite the fact that microsoft happened to be the most accurate API and the API with the least amount of tags per image, my theory was disproved by the fact that clarifai had the second highest accuracy percentage but provided the greatest number of tags per image.

E. **Complete Description of Tasks Mentioned in the Proposal/Interim Report but Not Accomplished**: For the most part, all tasks mentioned in my proposal and interim report(s) were accomplished. After meeting with Professor Mandel and submitting my second interim report, I decided to add a method that searches through the tags for each image from each API and calculates the average number of tags provided per image. However, despite not being mentioned in the proposal or interim reports, I would have liked to implement a function that would read the tags from each API directly off of the html file provided from Cloudy Vision, so that the user would not have to manually transfer those results to a .txt file placed in the ‘vendors’ folder in order to use my program, yet due to time constraints I was unable to execute this concept. I would definitely like to improve the program to eliminate the use of the specific .txt files for each API and read directly from Cloudy Vision’s generated html output. In addition, I would have liked to add restraints/restrictions that would check the user’s manualResults.txt file to ensure that duplicate words are not counted twice as matching tags and that function words (auxiliary verbs, prepositions, articles, conjunctions, etc.) do not count/contribute to the total of tags evaluated and only content words would contribute to the overall total and matches, in case a user happened to include them in their .txt file. Instead, once again due to time restraints, I provided a disclaimer in the instructional markdown file that states that function words and duplicate words should not be included in the user’s manual evaluation tags provided for each individual image.

F. **COMPLETE PROGRAM AND SYSTEM DOCUMENTATION**:

a. **Programs**:

```
/**
 * Amara Auguste
 * Image Recognition API Evaluator:
 * Evaluates the API results from Gaurav Oberoi's Open Source tool --- Cloudy Vision,
 *  to test the image labeling capabilities of different computer vision API vendors, found here:
 * https://github.com/goberoi/cloudy_vision. Uses the Apache Commons IO library to generate
and write output
 * to an html file and JFreeChart to create and display visual representation of the statistical data.
 */

package imagerecognitionapis;

import java.util.*;
import java.io.*;
import java.util.regex.*;
import javax.swing.SwingUtilities;
import org.apache.commons.io.FileUtils;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;

public class ImageRecognitionAPIs {

    public static char percentSymbol = '%';

    public static void main(String[] args) throws FileNotFoundException, IOException {
        //place names of all images to be evaluated in "images.txt"
        File imageList = new File("info/images.txt");
        Scanner sc = new Scanner(imageList);
        PrintWriter p = new PrintWriter("output.txt");
        ArrayList<Object> images = new ArrayList<>();
        while (sc.hasNext()) {
            images.add(sc.next());//read image names into arraylist
        }
        Object[] imagesArray = images.toArray();//convert arraylist to array
        sc.close();

        File htmlTemplateFile = new File("template.html");
        String htmlString = FileUtils.readFileToString(htmlTemplateFile);
        htmlString = htmlString.replace("$total", "Number of images to evaluate: " +
imagesArray.length);

        p.printf("Number of images to evaluate: %d\n", imagesArray.length);
        p.println("-----------------------------------------------");

        File manualFile = new File("info/manualResults.txt");
        Scanner manual = new Scanner(manualFile);
        Map<String, String> manualEvaluation = new LinkedHashMap<>();//stores manual results
for APIs to compare to
        String imageName = "";
        while (manual.hasNext()) {
          imageName = manual.next();//key
          String s = manual.nextLine();//value
          manualEvaluation.put(imageName, s);
        }

        manual.close();

        htmlString = evaluateAPIs(imagesArray, manualEvaluation, p, htmlString);//evaluates all
.txt files in the 'vendors' folder

        File newHtmlFile = new File("output.html");//html output file
        FileUtils.writeStringToFile(newHtmlFile, htmlString);//writes to html
        p.close();
    }//end of main

    /**
     * Iterates through the 'vendors' directory and compares the manual results to the results from
each API .txt file, calling containsExact to determine
     * whether or not the tags match and keeping count of tags that match, printing results to an
output file.
     */
    public static String evaluateAPIs(Object[] images, Map<String, String> m, PrintWriter p,
String htmlString) throws FileNotFoundException, IOException {
        File folder = new File("./vendors");//folder/directory where input files are located
        File[] listOfFiles = folder.listFiles();//array of files
        int numOfFiles = listOfFiles.length;//# of APIs to be evaluated

        API[] API_List = new API[numOfFiles];
        int[] countResults = new int[numOfFiles];//parallel arrays to store evaluation results
        double[] percent = new double[numOfFiles];//parallel arrays to store evaluation results
        double[] averageTags = new double[numOfFiles];//holds number of average tags per image
via each API
        int totalTagsCount = 0;

        String body = "";
        String result = "";
        System.out.println("Evaluation completed for: ");
        System.out.println("--------------------------");

        int i = 0;
        for (File file : listOfFiles) {
          if (file.isFile()) {
            System.out.println(file.getName());
            API_List[i] = new API(images.length);
            API.read(listOfFiles[i]);//reads data from API file(s)

            for (int j = 0; j < images.length; j++) {//runs for each image to be evaluated
              if (m.containsKey(API_List[i].getImage(j))) {//if image name exists in manual
evaluation
                String temp = m.get(API_List[i].getImage(j));//returns the string of tags found in
manual evaluation
                String[] words = temp.split("\\W+");//split tags into array
                for (int k = 0; k < words.length; k++) {//iterate through array of tags
                  if (containsExact(API_List[i].getTags(j), words[k])) {
                    countResults[i]++;//increment counter
                  }
                  totalTagsCount++;
                }
              }
            }
            averageTags[i] = computeAvgTags(API_List[i], images);//computes average number of
tags provide per image from each API
            result += "Average tags per image for " + listOfFiles[i].getName() + " is: " + (int)
averageTags[i] + "<br><br>";
            percent[i] = ((double) countResults[i] / totalTagsCount) * 100;//computes accuracy
percentage
            p.printf("For '%s', accuracy in comparison to manual evaluation: %d out of %d (tags)
or %.2f%c\n", listOfFiles[i].getName(), countResults[i], totalTagsCount, percent[i],
percentSymbol);
            body += "For " + listOfFiles[i].getName() + ", accuracy in comparison to manual
evaluation: " + countResults[i] + " out of " + totalTagsCount + " (tags) or " +
String.format("%.2f", percent[i]) + percentSymbol + "<br><br>";
            totalTagsCount = 0;

            i++;
         }
      }

      p.println();

      if (numOfFiles >= 1) { //if one or more APIs are to be evaluated, compare results for
accuracy
         htmlString = accuracy(listOfFiles, countResults, percent, p, htmlString, averageTags);
      }

      p.println("\nAverage Tags Per Image: ");
      p.println("--------------------------");
      String[] avg = result.split("<br><br>");
      for (int q = 0; q < avg.length; q++) {
         p.println(avg[q]);
      }

      createChart(listOfFiles, averageTags, "average tags", "Average Tags Per Image", "API",
"Average Tags", "averageChart.png");//create chart of average tags
      p.println();

      htmlString = htmlString.replace("$evaluations", body);
      htmlString = htmlString.replace("$average", result);
```

```
        return htmlString;
    }

    /**
     * Searches String (sentence) for a full instance of a specific String (word)
     */
    public static boolean containsExact(String fullString, String partWord) {
        String pattern = "\\b" + partWord + "\\b";
        Pattern p = Pattern.compile(pattern);
        Matcher m = p.matcher(fullString);
        return m.find();
    }

    /**
     * Orders the results: creates and displays ranked list based on accuracy in descending order
states most and least accurate APIs based on results
     */// double[] averageTags
    public static String accuracy(File[] f, int[] countResults, double[] percent, PrintWriter p, String
htmlString, double[] averageTags) throws IOException {

        p.println("API Rankings (Most to least accurate): ");
        p.println("---------------------------------------");
        int temp;
        double temp2;
        double temp3;
        File tempF;
        String body = "";
        for (int i = 0; i < (countResults.length - 1); i++) {//sorts the results in descending order in
order to rank them
            for (int j = 0; j < countResults.length - i - 1; j++) {
                if (countResults[j] < countResults[j + 1]) {
                    temp = countResults[j];
                    countResults[j] = countResults[j + 1];
                    countResults[j + 1] = temp;

                    temp2 = percent[j];
                    percent[j] = percent[j + 1];
                    percent[j + 1] = temp2;
                    temp3 = averageTags[j];
                    averageTags[j] = averageTags[j + 1];
                    averageTags[j + 1] = temp3;

                    tempF = f[j];
                    f[j] = f[j + 1];
                    f[j + 1] = tempF;

                }
            }
        }

        for (int k = 0; k < countResults.length; k++) {
            int place = k + 1;//rank
            p.printf("%d) '%s': %.2f%c accuracy\n", place, f[k].getName(), percent[k],
percentSymbol);
            body += place + ") " + f[k].getName() + ": " + String.format("%.2f", percent[k]) +
percentSymbol + "<br><br>";
        }

        htmlString = htmlString.replace("$rankings", body);
        p.println();

        int currentMax = countResults[0];
        int bestIdx = 0;
        int worstIdx = 0;
        for (int j = 1; j < countResults.length; j++) {
            if (countResults[j] > currentMax) {//compares results to find the most accurate
                currentMax = countResults[j];
                bestIdx = j;
            } else {
                worstIdx = j;
            }
        }

        createChart(f, percent, "percentage", "API Accuracy", "API", "Percentage",
"percentageChart.png");//creates bar chart displaying accuracy percentages
```

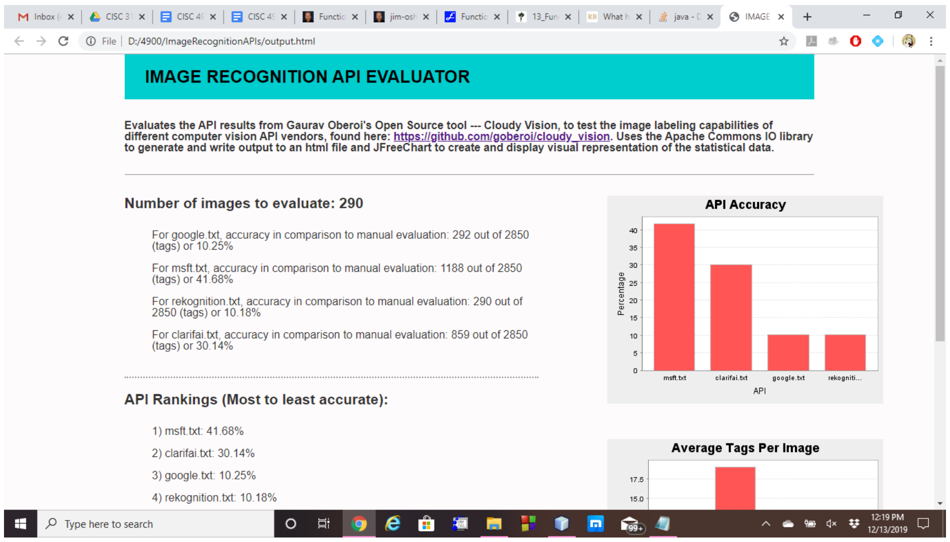
```
        p.printf("Most accurate API is: '%s' at %.2f%c accuracy\n", f[bestIdx].getName(),
percent[bestIdx], percentSymbol);//displays most accurate API
        p.printf("Least accurate API is: '%s' at %.2f%c accuracy\n", f[worstIdx].getName(),
percent[worstIdx], percentSymbol);//displays most accurate API
        htmlString = htmlString.replace("$most", "Most accurate API is: " + f[bestIdx].getName() +
" at " + String.format("%.2f", percent[bestIdx]) + percentSymbol + " accuracy " + "<br><br>");
        htmlString = htmlString.replace("$least", "Least accurate API is: " + f[worstIdx].getName()
+ " at " + String.format("%.2f", percent[worstIdx]) + percentSymbol + " accuracy " +
"<br><br>");

        return htmlString;
    }

    /**
     * Creates bar chart with data from the API accuracy percentage(s) and saves as .png file to be
displayed in the .html file
     */
    public static void createChart(File[] f, double[] data, String dataValue, String chartName,
String X, String Y, String saveImageName) throws IOException {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        for (int i = 0; i < f.length; i++) {
            dataset.addValue(data[i], dataValue, f[i].getName());
        }

        JFreeChart barChart = ChartFactory.createBarChart(
                chartName,
                X,
                Y,
                dataset,
                PlotOrientation.VERTICAL,
                false, true, false);

        ChartUtilities.saveChartAsPNG(new File("images/" + saveImageName), barChart, 400,
300);//saves chart as png image

    }
    /**
     * Counts up tags for each image from each API
     * divides that total by the number of images analyzed
     * to compute average number of tags per image
     */
    public static int computeAvgTags(API API_List, Object[] images) {
        int sum = 0;
        for (int i = 0; i < API_List.tags.length; i++) {
            if (API_List.tags[i] != null || !(API_List.tags[i].isEmpty())) {
                String[] words = API_List.tags[i].split(",");
                sum += words.length;

            }
        }

        return sum / images.length;

    }

}
```

```
/**
 * Amara Auguste
 * Used to create an API object which holds the names of
 * the images analyzed and the corresponding tags
 * provided by each API
 */

package imagerecognitionapis;

import java.io.*;
import java.util.Scanner;

public class API {

   public static String[] imageList;
   public static String[] tags;

   API(int n) {
      imageList = new String[n];
      tags = new String[n];
   }

   public static void read(File f) throws FileNotFoundException {
      Scanner scan = new Scanner(f);

      for (int i = 0; i < imageList.length; i++) {
         imageList[i] = scan.next();
         tags[i] = scan.nextLine();
      }
   }


   public String getImage(int i){
      return imageList[i];
   }

   public String getTags(int i){
      return tags[i];
```

```
    }

    public static void print() {
        for (int i = 0; i < imageList.length; i++) {
            System.out.print(imageList[i] + "\t");
            System.out.print(tags[i] + "\n");
        }
    }

    public static int compare (API a){
        int count = 0;


        return count;
    }
}
```

b. **<u>Description of Data Inputs and Outputs/Sample Outputs</u>**:

- Data Inputs:
  - A .txt file of manually evaluated data (default/"normalized" data, used to compare the results of the APIs to)
  - Individual .txt files of results from each API (each API's results should be contained to one .txt file) stored in the 'vendors' folder
- Outputs/Sample Outputs:
  - An output.txt file displaying:
    - The number of images analyzed
    - The number of matching tags and percentage of accuracy from each API
    - The APIs ranked from most to least accurate
    - The names of the most and least accurate APIs
    - The average number of tags per image from each API evaluated
  - In the images folder:
    - A bar chart displaying the accuracy percentages from each API
    - A bar chart displaying the average number of tags per image from each API evaluated
  - An output.html file neatly displaying the text from the output.txt file and the accompanying bar chart images:



D:/4900/ImageRecognitionAPIs/output.html

IMAGE RECOGNITION API EVALUATOR

Evaluates the API results from Gaurav Oberoi's Open Source tool --- Cloudy Vision, to test the image labeling capabilities of different computer vision API vendors, found here: https://github.com/goberoi/cloudy_vision. Uses the Apache Commons IO library to generate and write output to an html file and JFreeChart to create and display visual representation of the statistical data.

Number of images to evaluate: 290

For google.txt, accuracy in comparison to manual evaluation: 292 out of 2850 (tags) or 10.25%

For msft.txt, accuracy in comparison to manual evaluation: 1188 out of 2850 (tags) or 41.68%

For rekognition.txt, accuracy in comparison to manual evaluation: 290 out of 2850 (tags) or 10.18%

For clarifai.txt, accuracy in comparison to manual evaluation: 859 out of 2850 (tags) or 30.14%

API Rankings (Most to least accurate):

1) msft.txt: 41.68%

2) clarifai.txt: 30.14%

3) google.txt: 10.25%

4) rekognition.txt: 10.18%

c. **User Documentation**:

IMAGE RECOGNITION API EVALUATOR:

Evaluates the API results from Gaurav Oberoi's Open Source tool --- Cloudy Vision, to test the image labeling capabilities of different computer vision API vendors, found here: https://github.com/goberoi/cloudy_vision. Uses the Apache Commons IO library to generate and write output to an html file and JFreeChart to create and display visual representation of the statistical data.

How it works/Usage:

1. After running cloudy_vision.py, record results for each API in a .txt file in the order of:
"image filename"   "list of tags"
(So if evaluating AWS rekognition, Clarifai, and IBM Watson, etc. then each API needs it's own corresponding text file) and save recorded results from Cloudy Vision in individual .txt files and save in 'vendors' folder(if saving in notepad, deselect 'word wrap' under the 'format' menu).
2. Save list of image filenames to be evaluated in a .txt file named: "image.txt" and save manual evaluation (tags that the user wants the APIs to be compared to) in the 'info' folder.
3. Record results from manual evaluation (or whichever data that you would like to initially compare the API results to) in a text file titled 'manualResults.txt' with the same format as the API tags:
"image filename"   "list of tags"
and save in 'info' folder (once again, if saving in notepad, deselect 'word wrap' under the 'format' menu). Limit (try not to feature at all) the use of duplicate terms and function words (prepositions, conjunctions, like: a, an, the, etc.) within the list of tags per each image
4. The data from each vendor .txt file is saved as an Object called 'API' with a String [] imageList and String [] tags respectively (API.java).
5. Runs through array of API objects, comparing tags found in the manual evaluation to tags found from an API (results may vary depending on the user's manual evaluation) and computes accuracy as a percentage.
6. If amount of APIs evaluated is greater than one, ranks the APIs based on descending accuracy and states the most and least accurate APIs and their percentages.

7. Computes accuracy percentage per API and average number of tags provided per image from each API and creates a bar chart displaying the results, saving the resulting charts in the 'images' folder as 'percentageChart.png' and 'averageChart.png' respectively.
8. Displays all resulting data to output.txt file and output.html file in the html format from the template.html file.

G. **Project Log**:

09/22/2019

- 9:06 am - Looked into Gaurav Oberoi's image recognition tool, **Cloudy Vision**, which presents image labeling results from Microsoft, Google, IBM, Clarifai, and Cloud Sight (the main APIs discussed in evaluation of images for this project). **Cloudy Vision** is an open source tool written in Python to run images through various image recognition APIs and provide image labeling results side by side. Due to my personal lack of knowledge in Python, plus newfound interest in the language, I have decided to study (at least for now) the basics. My only experience with Python (as of currently) is a program to create an html webpage in a browser, displaying NASA's Astronomy Picture of the Day (a homework assignment for the course that I am also currently taking CISC 3140 [MW2]: DESIGN & IMPLEMENTATION OF SOFTWARE APPLICATIONS II, assignment can be found here)

- TO DO:
  1) DOWNLOAD + SETUP/INSTALL CLOUDY VISION
  2) BEGIN STUDYING THE BASICS OF PYTHON

- 9:19 am - Started free Python beginner's tutorial at https://www.learnpython.org
- 12:30 pm - Learned print statements, indentation, variables, lists, basic operators and string formatting in Python via https://www.learnpython.org

09/23/2019

- 12:00 pm (until 2pm due to my 2:15pm class) - Continued Python beginner's tutorial: string formatting, basic string operations, conditions, loops, and functions
- 5:00 pm (after classes) - Downloaded (but did not set up yet) **Cloudy Vision**

09/24/2019

- 10:43 am - obtained API keys for Clarifai and IBM
- 3:00 pm (until 4pm due to 4pm work schedule) - Continued Python beginner's tutorial: classes and objects, dictionaries, and modules and packages

09/27/2019

- 12:00 pm - obtained API keys for Google Cloud and Amazon
- 3:14 pm - divided the 290 test images into 10 folders of 29 different images to be evaluated

09/30/2019

- 10:00 am (to 12:00 pm) - Started Python Data Science tutorial: numpy arrays, and pandas basics
- 8:00 pm - Attempted to set up Clarifai according to: https://github.com/Clarifai/clarifai-python#setup (had some trouble setting up via CMD on Windows, need to look into more Python commands for Windows)

10/01/2019

- 4:30 pm - Began manually assessing the first folder (29 images) of test imagery and recorded how many Caribou were found per image

10/02/2019

- 8:00 pm - Looked into installing pip for Python into Windows so I could set up Clarifai and probably the other APIs as well. Set up pip according to: https://matthewhorne.me/how-to-install-python-and-pip-on-windows-10/
- 8:54 pm - Finally installed pip on Windows, will now proceed to set up Clarifai

10/04/2019
- 1:00 pm - Started Python Advanced Tutorials: Generators, and List Comprehensions
- 3:15 pm - Placed all images from Dropbox into ./input_images
- 3:30 pm - Installed dependencies needed to run Cloudy Vision:
- ❏ virtualenv venv
- ❏ source venv/bin/activate
- ❏ pip install -r requirements.txt

10/09/2019
- 5:45 pm - Ran into error attempting to run cloudy_vision.py to test Google, needed to install Jinja2, numpy, watson_developer_cloud, cloudsight, boto3, etc.
- 7:09 pm - Attempted to set up Clarifai for use since Clarifai needs to be configured outside of api_keys.json

10/10/2019
- 8:16 am - Wrote first interim report (due 10/15) and sent to Professor Mandel
- 11:40 am - Meeting with Professor Mandel to touch base on progress so far. Suggested one spreadsheet for manual data alone, decided to drop testing Cloud Sight and test MIcrosoft Azure Computer Recognition instead. Decided to manually check for other variables (besides Caribou) that may be identified by different APIs

10/13/2019
- 5:30 pm - Continued manual evaluation and log of data (next 29 images)

10/14/2019
- 11:00 am - Continued manual evaluation and log of data (next 29 images - folder #3)

10/16/2019
- 5:10 pm - Continued manual evaluation and log of data (next 29 images - folder #4)

10/20/2019
- 8:05 am - Continued Python Advanced Tutorials: Multiple Function Arguments and Regular Expressions

10/22/2019
- 11:10 am - Continued Python Advanced Tutorials: Exception Handling and Sets

10/24/2019
- 4:05 pm - Continued manual evaluation and log of data (next 29 images - folder #5)

10/25/2019
- 3:15 pm - Continued Python Advanced Tutorials: Serialization and Partial Functions

10/27/2019
- 4:15 pm - Continued manual evaluation and log of data (next 29 images - folder #6)

10/28/2019
- 9:53 am - Resumed debugging cloudy vision, obtained this error:

```
Traceback (most recent call last):
 File "cloudy_vision.py", line 64, in <module>
   if settings('resize'):
```

```
 File "cloudy_vision.py", line 59, in settings
   SETTINGS['api_keys'] = json.load(data_file)
 File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\__init__.py", line
296, in load
parse_constant=parse_constant, object_pairs_hook=object_pairs_hook, **kw)
 File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\__init__.py", line
348, in loads
 return _default_decoder.decode(s)
 File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\decoder.py", line
337, in decode
 obj, end = self.raw_decode(s, idx=_w(s, 0).end())
File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\decoder.py", line
355, in raw_decode
 raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 2 column 16 (char 17)
```

- 10:06 am - looked up: “json.decoder.JSONDecodeError” on the internet (Stack Overflow, etc.)
- 10:09 am - added double quotation marks (“ ”) around api keys in the api_keys file, error changed to:

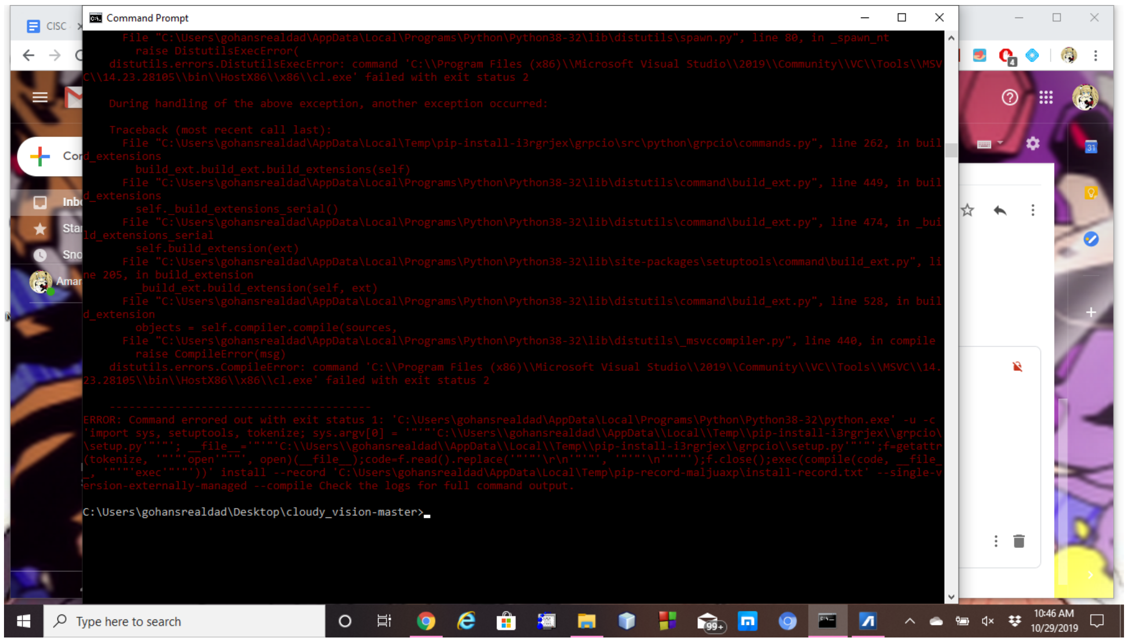
```
Traceback (most recent call last):
File "cloudy_vision.py", line 64, in <module>
if settings('resize'):
File "cloudy_vision.py", line 59, in settings
 SETTINGS['api_keys'] = json.load(data_file)
File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\__init__.py", line
296, in load
parse_constant=parse_constant, object_pairs_hook=object_pairs_hook, **kw)
 File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\__init__.py", line
348, in loads
return _default_decoder.decode(s)
 File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\decoder.py", line
337, in decode
 obj, end = self.raw_decode(s, idx=_w(s, 0).end())
 File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python37\lib\json\decoder.py", line
353, in raw_decode
obj, end = self.scan_once(s, idx)
json.decoder.JSONDecodeError: Expecting property name enclosed in double quotes: line 5
column 1 (char 125)
```

- 10:10 am - installed jsbeautifier in an attempt to fix the error (py -m pip install jsbeautifier), had to upgrade pip from version 19.2.3 to 19.3.1 (py -m pip --upgrade pip)
- 11:03 am - read about using JSON in attempt to debug cloudy vision here: https://realpython.com/python-json/
- 2:14 pm - validated and identified errors in my __init__.py and decoder.py files in JSON, because after some research the compiler errors seem to come from my JSON files opposed to cloudy_vision.py (using https://jsonlint.com/)
- 3:30 pm - compared my __init__.py and decoder.py files to the files that are found here: https://github.com/python/cpython/tree/master/Lib/json although they seem to

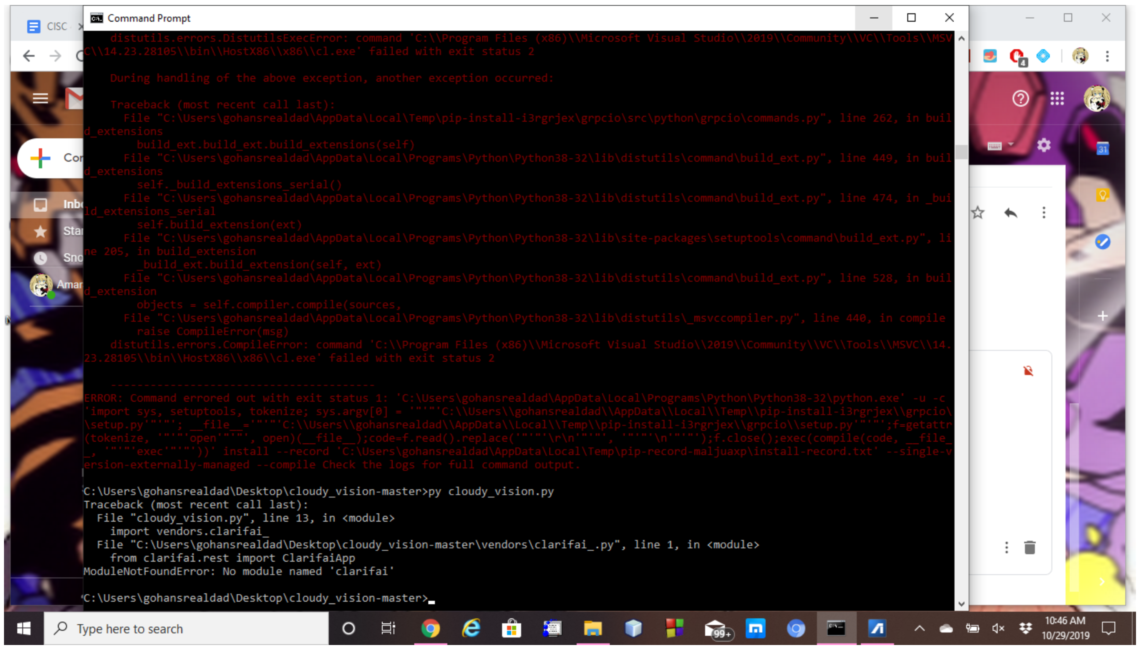
be the same, both set of files (mine and those found on github) have errors according to the JSON validator online

10/29/2019

- 7:35 am - had to uninstall python 3.7.4 (apparently there were errors in installation) and install updated python 3.8.0, now have to install Jinja2 (py -m pip install Jinja2), upgrade pip from version 19.2.3 to version 19.3.1, install numpy, install requests, install clarifai, install wheel, upgrade setuptools
- 8:05 am - ran into error attempting to install clarifai, need to install Microsoft Visual Studio before clarifai set up can be completed
- 10:47 am - Microsoft Visual Studio has finished installing yet I still have errors attempting to install clarifai:



```
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\spawn.py", line 80, in _spawn_nt
        raise DistutilsExecError(
    distutils.errors.DistutilsExecError: command 'C:\\Program Files (x86)\\Microsoft Visual Studio\\2019\\Community\\VC\\Tools\\MSV
C\\14.23.28105\\bin\\HostX86\\x86\\cl.exe' failed with exit status 2

    During handling of the above exception, another exception occurred:

    Traceback (most recent call last):
      File "C:\Users\gohansrealdad\AppData\Local\Temp\pip-install-i3rgrjex\grpcio\src\python\grpcio\commands.py", line 262, in buil
d_extensions
        build_ext.build_ext.build_extensions(self)
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\command\build_ext.py", line 449, in buil
d_extensions
        self._build_extensions_serial()
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\command\build_ext.py", line 474, in _bui
ld_extensions_serial
        self.build_extension(ext)
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\site-packages\setuptools\command\build_ext.py", li
ne 205, in build_extension
        _build_ext.build_extension(self, ext)
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\command\build_ext.py", line 528, in buil
d_extension
        objects = self.compiler.compile(sources,
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\_msvccompiler.py", line 440, in compile
        raise CompileError(msg)
    distutils.errors.CompileError: command 'C:\\Program Files (x86)\\Microsoft Visual Studio\\2019\\Community\\VC\\Tools\\MSVC\\14.
23.28105\\bin\\HostX86\\x86\\cl.exe' failed with exit status 2

    ----------------------------------------
ERROR: Command errored out with exit status 1: 'C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\python.exe' -u -c
'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\\Users\\gohansrealdad\\AppData\\Local\\Temp\\pip-install-i3rgrjex\\grpcio\
\setup.py'"'"'; __file__='"'"'C:\\Users\\gohansrealdad\\AppData\\Local\\Temp\\pip-install-i3rgrjex\\grpcio\\setup.py'"'"';f=getattr
(tokenize, '"'"'open'"'"', open)(__file__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file_
_, '"'"'exec'"'"'))' install --record 'C:\Users\gohansrealdad\AppData\Local\Temp\pip-record-maljuaxp\install-record.txt' --single-v
ersion-externally-managed --compile Check the logs for full command output.

C:\Users\gohansrealdad\Desktop\cloudy_vision-master>
```



```
    distutils.errors.DistutilsExecError: command 'C:\\Program Files (x86)\\Microsoft Visual Studio\\2019\\Community\\VC\\Tools\\MSV
C\\14.23.28105\\bin\\HostX86\\x86\\cl.exe' failed with exit status 2

    During handling of the above exception, another exception occurred:

    Traceback (most recent call last):
      File "C:\Users\gohansrealdad\AppData\Local\Temp\pip-install-i3rgrjex\grpcio\src\python\grpcio\commands.py", line 262, in buil
d_extensions
        build_ext.build_ext.build_extensions(self)
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\command\build_ext.py", line 449, in buil
d_extensions
        self._build_extensions_serial()
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\command\build_ext.py", line 474, in _bui
ld_extensions_serial
        self.build_extension(ext)
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\site-packages\setuptools\command\build_ext.py", li
ne 205, in build_extension
        _build_ext.build_extension(self, ext)
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\command\build_ext.py", line 528, in buil
d_extension
        objects = self.compiler.compile(sources,
      File "C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\lib\distutils\_msvccompiler.py", line 440, in compile
        raise CompileError(msg)
    distutils.errors.CompileError: command 'C:\\Program Files (x86)\\Microsoft Visual Studio\\2019\\Community\\VC\\Tools\\MSVC\\14.
23.28105\\bin\\HostX86\\x86\\cl.exe' failed with exit status 2

    ----------------------------------------
ERROR: Command errored out with exit status 1: 'C:\Users\gohansrealdad\AppData\Local\Programs\Python\Python38-32\python.exe' -u -c
'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\\Users\\gohansrealdad\\AppData\\Local\\Temp\\pip-install-i3rgrjex\\grpcio\
\setup.py'"'"'; __file__='"'"'C:\\Users\\gohansrealdad\\AppData\\Local\\Temp\\pip-install-i3rgrjex\\grpcio\\setup.py'"'"';f=getattr
(tokenize, '"'"'open'"'"', open)(__file__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file_
_, '"'"'exec'"'"'))' install --record 'C:\Users\gohansrealdad\AppData\Local\Temp\pip-record-maljuaxp\install-record.txt' --single-v
ersion-externally-managed --compile Check the logs for full command output.

C:\Users\gohansrealdad\Desktop\cloudy_vision-master>py cloudy_vision.py
Traceback (most recent call last):
  File "cloudy_vision.py", line 13, in <module>
    import vendors.clarifai_
  File "C:\Users\gohansrealdad\Desktop\cloudy_vision-master\vendors\clarifai_.py", line 1, in <module>
    from clarifai.rest import ClarifaiApp
ModuleNotFoundError: No module named 'clarifai'

C:\Users\gohansrealdad\Desktop\cloudy_vision-master>
```

Amara Auguste, CISC 4900, Fall 2019

- 12:30 pm - after consulting Professor Mandel, he suggested installing conda to manage running python on Windows

10/30/2019

- 4:00 pm - Continued Python Advanced Tutorial: Code Introspection and Closures

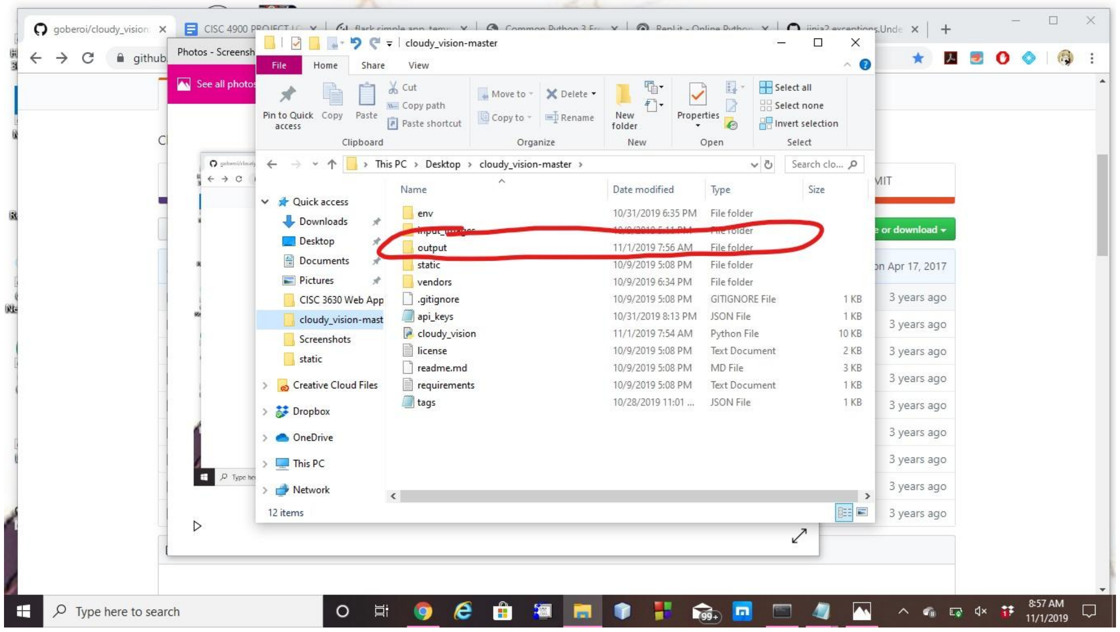
10/31/2019

- 11:15 am - Meeting with Professor Mandel to debug cloudy vision and attempt to get it up and running to begin using it to obtain results from APIs, suggested stop using 'py' command and use 'python' command instead
- 5:20 pm - installed microsoft visual c++ build tools 14.0 and fixed clarifai error (see previous screenshots), still having error:

```
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

11/01/2019

- 8:00 am - rid of previous json error and connected to 'templates.html' file, now output folder is produced (albeit empty):



This PC › Desktop › cloudy_vision-master ›

Name	Date modified	Type	Size
env	10/31/2019 6:35 PM	File folder	
input_images	10/9/2019 5:11 PM	File folder	
output	11/1/2019 7:56 AM	File folder	
static	10/9/2019 5:08 PM	File folder	
vendors	10/9/2019 6:34 PM	File folder	
.gitignore	10/9/2019 5:08 PM	GITIGNORE File	1 KB
api_keys	10/31/2019 8:13 PM	JSON File	1 KB
cloudy_vision	11/1/2019 7:54 AM	Python File	10 KB
license	10/9/2019 5:08 PM	Text Document	2 KB
readme.md	10/9/2019 5:08 PM	MD File	3 KB
requirements	10/9/2019 5:08 PM	Text Document	1 KB
tags	10/28/2019 11:01 ...	JSON File	1 KB

12 items

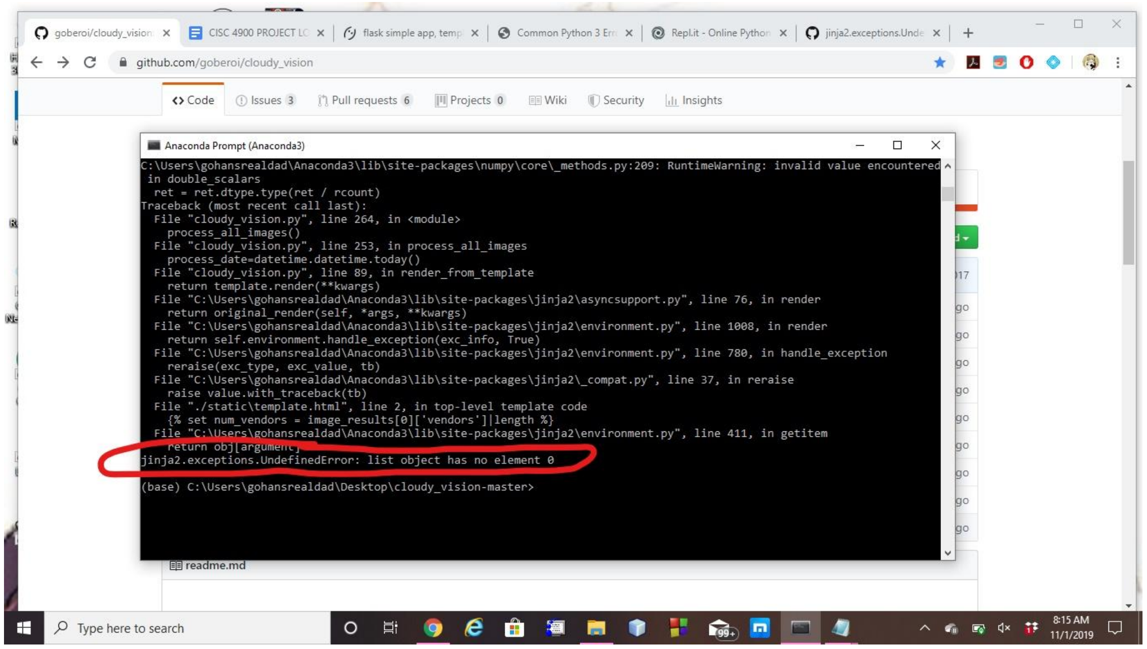
and now obtaining this error:

```
jinja2.exceptions.UndefinedError: list object has no element 0
```

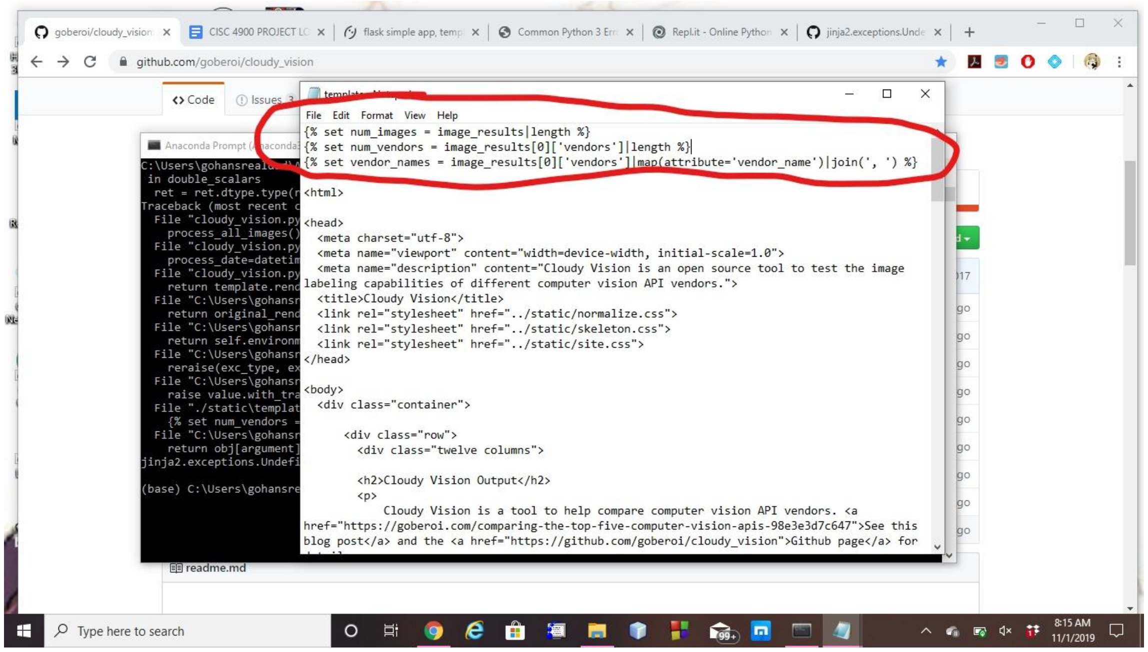
Pertaining to this line in templates.html:

```
{% set num_vendors = image_results[0]['vendors']|length %}
```

Amara Auguste, CISC 4900, Fall 2019

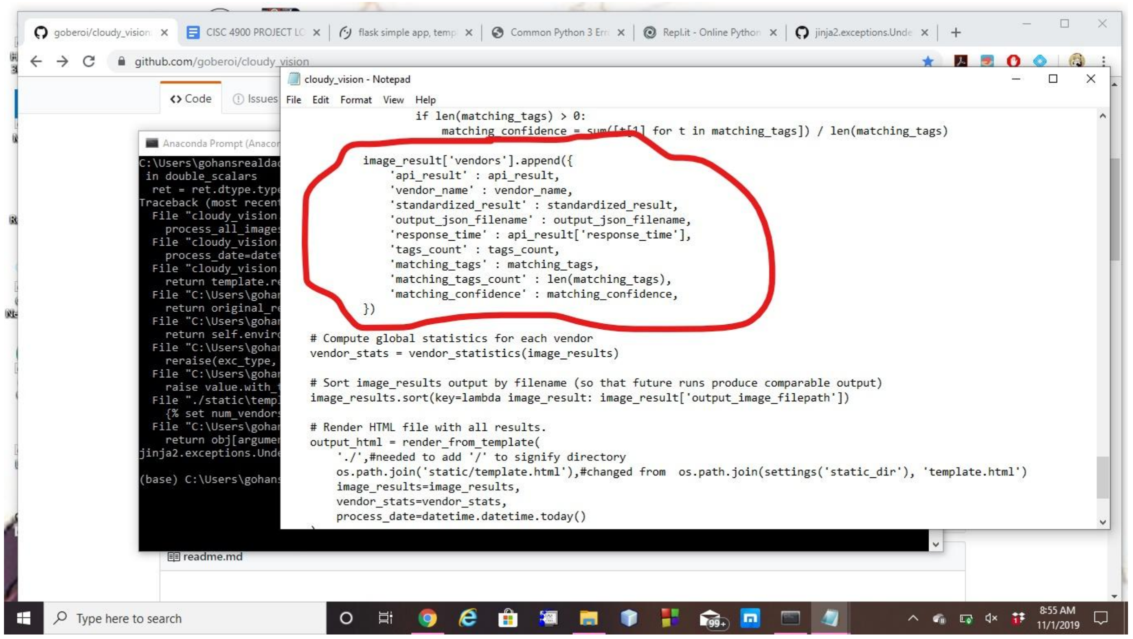


```
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core\_methods.py:209: RuntimeWarning: invalid value encountered
 in double_scalars
  ret = ret.dtype.type(ret / rcount)
Traceback (most recent call last):
  File "cloudy_vision.py", line 264, in <module>
    process_all_images()
  File "cloudy_vision.py", line 253, in process_all_images
    process_date=datetime.datetime.today()
  File "cloudy_vision.py", line 89, in render_from_template
    return template.render(**kwargs)
  File "C:\Users\gohansrealdad\Anaconda3\lib\site-packages\jinja2\asyncsupport.py", line 76, in render
    return original_render(self, *args, **kwargs)
  File "C:\Users\gohansrealdad\Anaconda3\lib\site-packages\jinja2\environment.py", line 1008, in render
    return self.environment.handle_exception(exc_info, True)
  File "C:\Users\gohansrealdad\Anaconda3\lib\site-packages\jinja2\environment.py", line 780, in handle_exception
    reraise(exc_type, exc_value, tb)
  File "C:\Users\gohansrealdad\Anaconda3\lib\site-packages\jinja2\_compat.py", line 37, in reraise
    raise value.with_traceback(tb)
  File "./static\template.html", line 2, in top-level template code
    {% set num_vendors = image_results[0]['vendors']|length %}
  File "C:\Users\gohansrealdad\Anaconda3\lib\site-packages\jinja2\environment.py", line 411, in getitem
    return obj[argument]
jinja2.exceptions.UndefinedError: list object has no element 0

(base) C:\Users\gohansrealdad\Desktop\cloudy_vision-master>
```



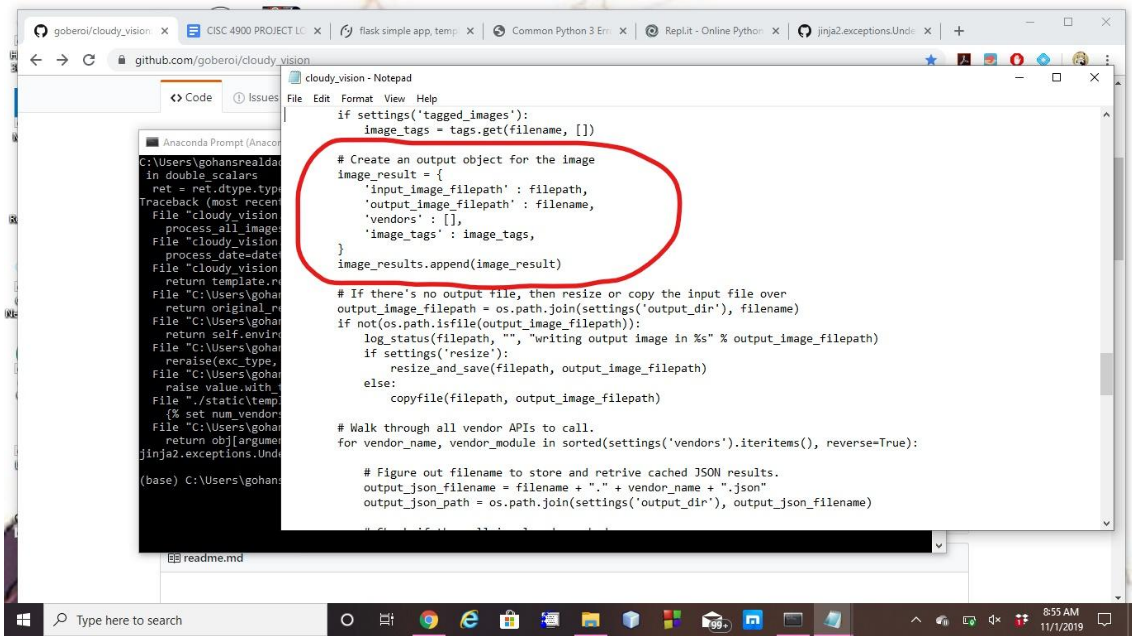
```
{% set num_images = image_results|length %}
{% set num_vendors = image_results[0]['vendors']|length %}
{% set vendor_names = image_results[0]['vendors']|map(attribute='vendor_name')|join(', ') %}

<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Cloudy Vision is an open source tool to test the image
labeling capabilities of different computer vision API vendors.">
  <title>Cloudy Vision</title>
  <link rel="stylesheet" href="../static/normalize.css">
  <link rel="stylesheet" href="../static/skeleton.css">
  <link rel="stylesheet" href="../static/site.css">
</head>

<body>
  <div class="container">

      <div class="row">
        <div class="twelve columns">

        <h2>Cloudy Vision Output</h2>
        <p>
            Cloudy Vision is a tool to help compare computer vision API vendors. <a
href="https://goberoi.com/comparing-the-top-five-computer-vision-apis-98e3e3d7c647">See this
blog post</a> and the <a href="https://github.com/goberoi/cloudy_vision">Github page</a> for
```

Amara Auguste, CISC 4900, Fall 2019



```
C:\Users\gohansrealdad
 in double_scalars
  ret = ret.dtype.type
Traceback (most recent
  File "cloudy_vision
    process_all_images
  File "cloudy_vision
    process_date=datet
  File "cloudy_vision
    return template.re
  File "C:\Users\gohan
    return original_re
  File "C:\Users\gohan
    return self.enviro
  File "C:\Users\gohan
    reraise(exc_type,
  File "C:\Users\gohan
    raise value.with_t
  File "./static\templ
    {% set num_vendors
  File "C:\Users\gohan
    return obj[argumen
jinja2.exceptions.Unde

(base) C:\Users\gohans
```

```
                if len(matching_tags) > 0:
                    matching_confidence = sum([t[1] for t in matching_tags]) / len(matching_tags)

            image_result['vendors'].append({
                'api_result' : api_result,
                'vendor_name' : vendor_name,
                'standardized_result' : standardized_result,
                'output_json_filename' : output_json_filename,
                'response_time' : api_result['response_time'],
                'tags_count' : tags_count,
                'matching_tags' : matching_tags,
                'matching_tags_count' : len(matching_tags),
                'matching_confidence' : matching_confidence,
            })

    # Compute global statistics for each vendor
    vendor_stats = vendor_statistics(image_results)

    # Sort image_results output by filename (so that future runs produce comparable output)
    image_results.sort(key=lambda image_result: image_result['output_image_filepath'])

    # Render HTML file with all results.
    output_html = render_from_template(
        './',#needed to add '/' to signify directory
        os.path.join('static/template.html'),#changed from  os.path.join(settings('static_dir'), 'template.html')
        image_results=image_results,
        vendor_stats=vendor_stats,
        process_date=datetime.datetime.today()
```



```
C:\Users\gohansrealdad
 in double_scalars
  ret = ret.dtype.type
Traceback (most recent
  File "cloudy_vision
    process_all_images
  File "cloudy_vision
    process_date=datet
  File "cloudy_vision
    return template.re
  File "C:\Users\gohan
    return original_re
  File "C:\Users\gohan
    return self.enviro
  File "C:\Users\gohan
    reraise(exc_type,
  File "C:\Users\gohan
    raise value.with_t
  File "./static\templ
    {% set num_vendors
  File "C:\Users\gohan
    return obj[argumen
jinja2.exceptions.Unde

(base) C:\Users\gohans
```

```
        if settings('tagged_images'):
            image_tags = tags.get(filename, [])

        # Create an output object for the image
        image_result = {
            'input_image_filepath' : filepath,
            'output_image_filepath' : filename,
            'vendors' : [],
            'image_tags' : image_tags,
        }
        image_results.append(image_result)

        # If there's no output file, then resize or copy the input file over
        output_image_filepath = os.path.join(settings('output_dir'), filename)
        if not(os.path.isfile(output_image_filepath)):
            log_status(filepath, "", "writing output image in %s" % output_image_filepath)
            if settings('resize'):
                resize_and_save(filepath, output_image_filepath)
            else:
                copyfile(filepath, output_image_filepath)

        # Walk through all vendor APIs to call.
        for vendor_name, vendor_module in sorted(settings('vendors').iteritems(), reverse=True):

            # Figure out filename to store and retrive cached JSON results.
            output_json_filename = filename + "." + vendor_name + ".json"
            output_json_path = os.path.join(settings('output_dir'), output_json_filename)
```

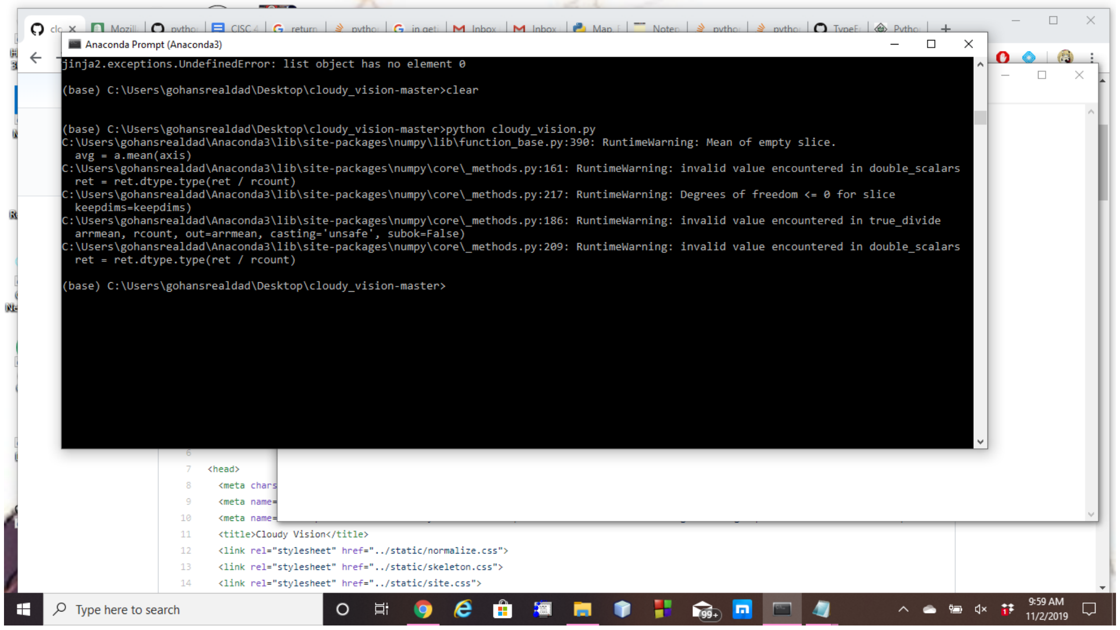
- 3:27 pm - Continued Python Advanced Tutorial: Decorators, and Map, Filter, and Reduce

11/02/2019

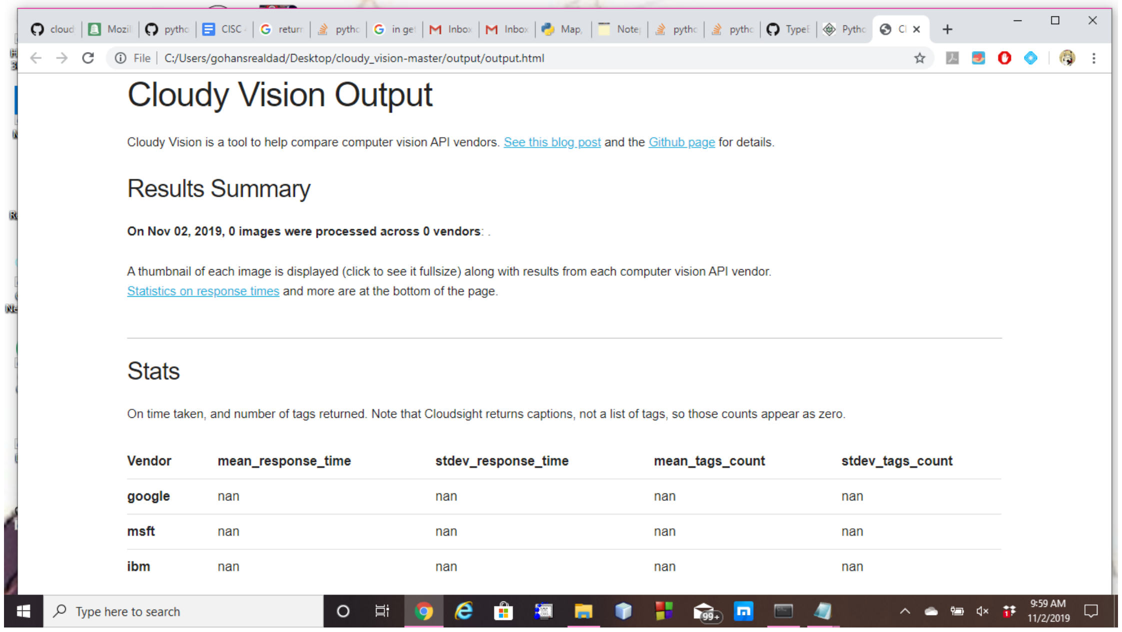
- 9:56 am - managed to get cloudy_vision.py to run, albeit with runtime warnings and the output.html file produced shows that the code does not process any images. Runtime errors shown are:

  C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\lib\function_base.py:390: RuntimeWarning: Mean of empty slice.
  avg = a.mean(axis)

C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core_methods.py:161:
RuntimeWarning: invalid value encountered in double_scalars
ret = ret.dtype.type(ret / rcount)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core_methods.py:217:
RuntimeWarning: Degrees of freedom <= 0 for slice
keepdims=keepdims)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core_methods.py:186:
RuntimeWarning: invalid value encountered in true_divide
arrmean, rcount, out=arrmean, casting='unsafe', subok=False)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core_methods.py:209:
RuntimeWarning: invalid value encountered in double_scalars
ret = ret.dtype.type(ret / rcount)



```
jinja2.exceptions.UndefinedError: list object has no element 0

(base) C:\Users\gohansrealdad\Desktop\cloudy_vision-master>clear


(base) C:\Users\gohansrealdad\Desktop\cloudy_vision-master>python cloudy_vision.py
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\lib\function_base.py:390: RuntimeWarning: Mean of empty slice.
  avg = a.mean(axis)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core\_methods.py:161: RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core\_methods.py:217: RuntimeWarning: Degrees of freedom <= 0 for slice
  keepdims=keepdims)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core\_methods.py:186: RuntimeWarning: invalid value encountered in true_divide
  arrmean, rcount, out=arrmean, casting='unsafe', subok=False)
C:\Users\gohansrealdad\Anaconda3\lib\site-packages\numpy\core\_methods.py:209: RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)

(base) C:\Users\gohansrealdad\Desktop\cloudy_vision-master>
```



Cloudy Vision Output

Cloudy Vision is a tool to help compare computer vision API vendors. See this blog post and the Github page for details.

Results Summary

On Nov 02, 2019, 0 images were processed across 0 vendors: .

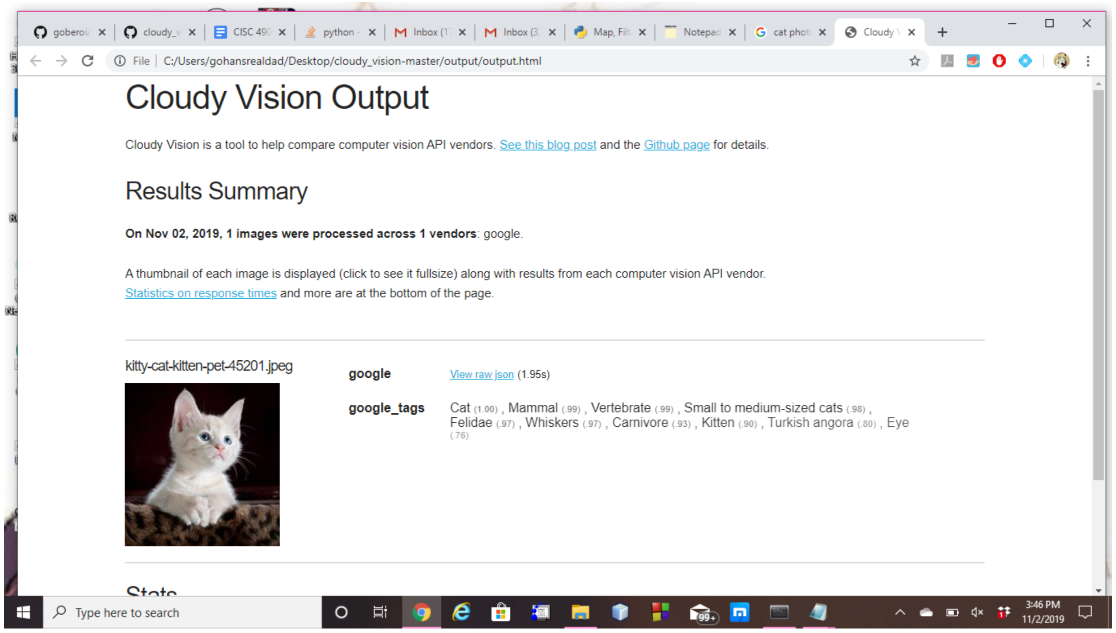
A thumbnail of each image is displayed (click to see it fullsize) along with results from each computer vision API vendor.
Statistics on response times and more are at the bottom of the page.

Stats

On time taken, and number of tags returned. Note that Cloudsight returns captions, not a list of tags, so those counts appear as zero.

Vendor	mean_response_time	stdev_response_time	mean_tags_count	stdev_tags_count
google	nan	nan	nan	nan
msft	nan	nan	nan	nan
ibm	nan	nan	nan	nan

- 3:41 pm - testing cloudy_vision.py with one sample photo and and only google vision api, connection made, need to change all '.iteritems()' calls into '.items()' calls because '.iteritems()' was removed from python 3
- 3:46 pm - cloudy_vision.py seems to work for sample photo using google api:



C:/Users/gohansrealdad/Desktop/cloudy_vision-master/output/output.html

Cloudy Vision Output

Cloudy Vision is a tool to help compare computer vision API vendors. See this blog post and the Github page for details.

Results Summary

On Nov 02, 2019, 1 images were processed across 1 vendors: google.

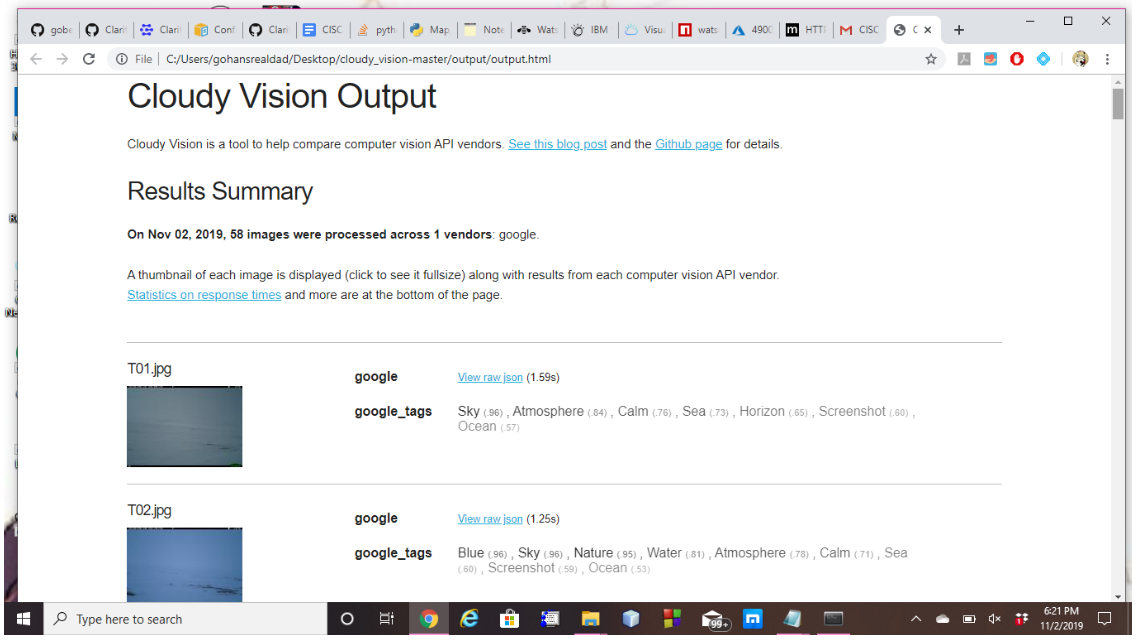
A thumbnail of each image is displayed (click to see it fullsize) along with results from each computer vision API vendor.
Statistics on response times and more are at the bottom of the page.

kitty-cat-kitten-pet-45201.jpeg

google View raw json (1.95s)

google_tags Cat (1.00) , Mammal (.99) , Vertebrate (.99) , Small to medium-sized cats (.98) , Felidae (.97) , Whiskers (.97) , Carnivore (.93) , Kitten (.90) , Turkish angora (.80) , Eye (.76)

- 3:59 pm - However, although it seems to work for my samples, the list attribute error happens when I try to use the test data. Apparently for some reason, despite being listed as JPG files, the test data is not being picked up by the code, to work around this I had to open the images in microsoft paint and save them as JPEG files which seemed to fix the problem



C:/Users/gohansrealdad/Desktop/cloudy_vision-master/output/output.html

Cloudy Vision Output

Cloudy Vision is a tool to help compare computer vision API vendors. See this blog post and the Github page for details.

Results Summary

On Nov 02, 2019, 58 images were processed across 1 vendors: google.

A thumbnail of each image is displayed (click to see it fullsize) along with results from each computer vision API vendor.
Statistics on response times and more are at the bottom of the page.

T01.jpg

google View raw json (1.59s)

google_tags Sky (.96) , Atmosphere (.84) , Calm (.76) , Sea (.73) , Horizon (.65) , Screenshot (.60) , Ocean (.57)

T02.jpg

google View raw json (1.25s)

google_tags Blue (.96) , Sky (.96) , Nature (.95) , Water (.81) , Atmosphere (.78) , Calm (.71) , Sea (.60) , Screenshot (.59) , Ocean (.53)

- 4:10 pm - received error:

DeprecationWarning: watson-developer-cloud moved to ibm-watson. To get updates, use the new package.

11/03/2019

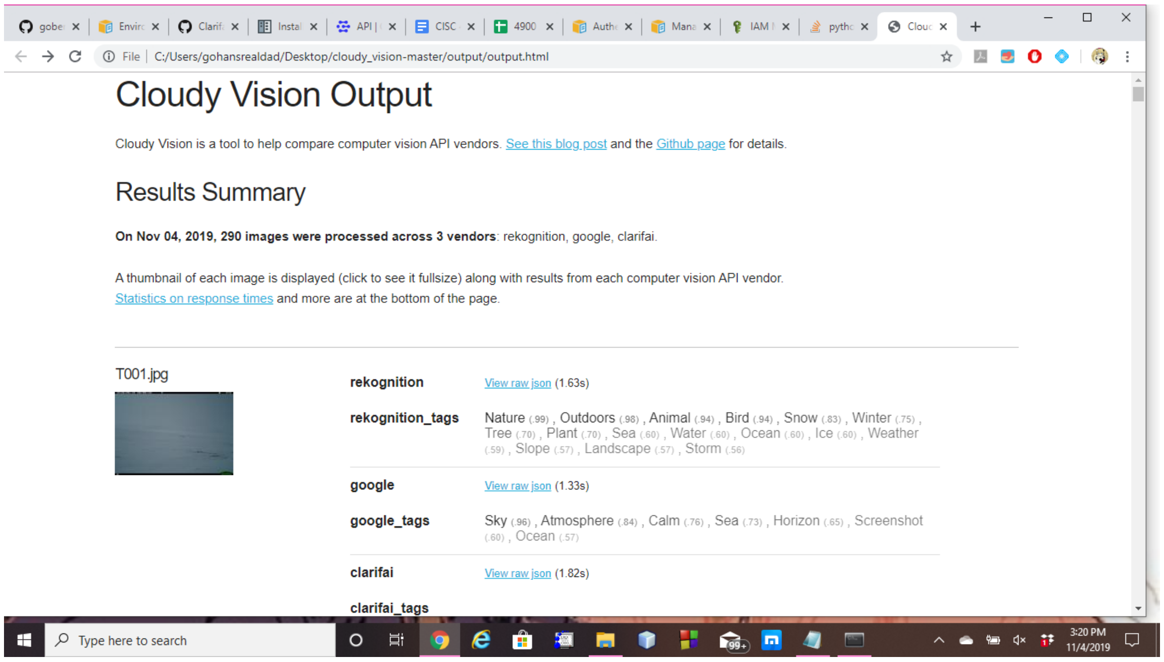
- 8:05 am - commented out the other four APIs in cloudy_vision.py and ran it on the 290 test images for just the google API (will attempt to fix the other API issues later) and began logging the google image results and tags in a spreadsheet

11/04/2019

- 1:00 pm - set clarifai api key as an environmental variable named: 'CLARIFAI_API_KEY', attempted to run cloudy_vision.py for clarifai and received error:

  AttributeError: 'zip' object has no attribute 'sort'

  Had to convert from zip object to list object: list(zip).sort(......) and convert from len(zip) to len(list(zip)
- 3:20 pm - added aws api access key, secret key, and region as environmental variables and ran cloudy_vision.py to successfully generate output:



C:/Users/gohansrealdad/Desktop/cloudy_vision-master/output/output.html

Cloudy Vision Output

Cloudy Vision is a tool to help compare computer vision API vendors. See this blog post and the Github page for details.

Results Summary

On Nov 04, 2019, 290 images were processed across 3 vendors: rekognition, google, clarifai.

A thumbnail of each image is displayed (click to see it fullsize) along with results from each computer vision API vendor.
Statistics on response times and more are at the bottom of the page.

T001.jpg

rekognition	View raw json (1.63s)
rekognition_tags	Nature (.99) , Outdoors (.98) , Animal (.94) , Bird (.94) , Snow (.83) , Winter (.75) , Tree (.70) , Plant (.70) , Sea (.60) , Water (.60) , Ocean (.60) , Ice (.60) , Weather (.59) , Slope (.57) , Landscape (.57) , Storm (.56)
google	View raw json (1.33s)
google_tags	Sky (.96) , Atmosphere (.84) , Calm (.76) , Sea (.73) , Horizon (.65) , Screenshot (.60) , Ocean (.57)
clarifai	View raw json (1.82s)
clarifai_tags	

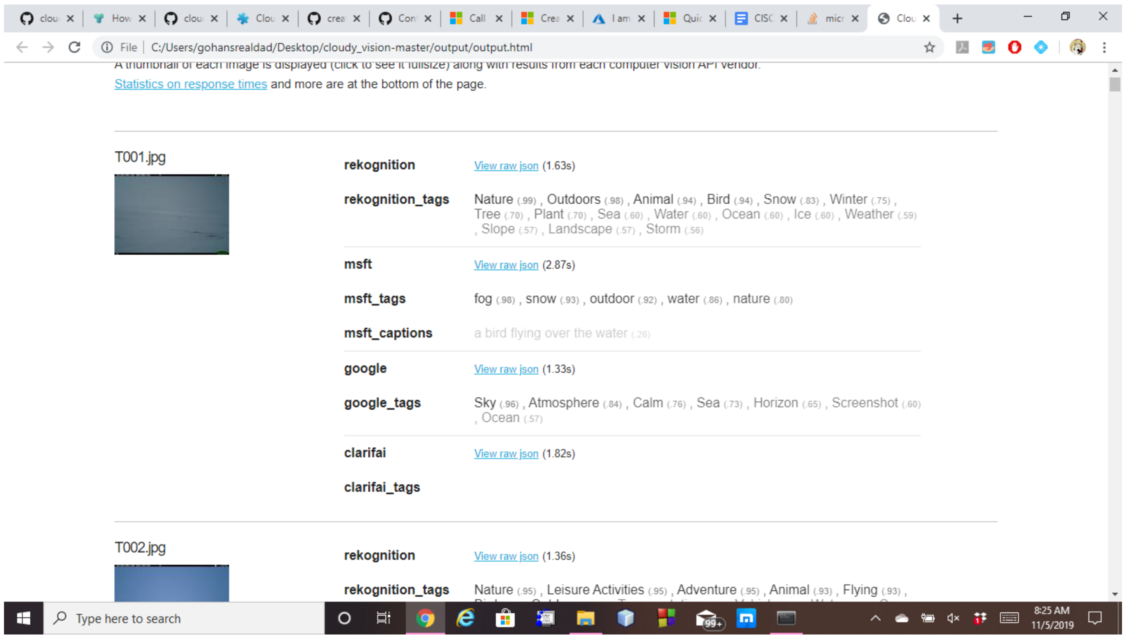
Began logging tags for AWS REKOGNITION in spreadsheet

11/05/2019

- 7:35 am - changed post_url = "https://api.projectoxford.ai/vision/v1.0/analyze?visualFeatures=Categories,Tags,Description,Faces,ImageType,Color,Adult&subscription-key=" + api_key line to: post_url = "https://westus.api.cognitive.microsoft.com/vision/v2.1/analyze?visualFeatures=Description,Tags&subscription-key=" + api_key in Cloudy Vision's 'microsoft.py' file in the vendors folders and successfully re-ran cloudy_vision.py for the microsoft computer vision API
- 7:45 am - received: requests.exceptions.HTTPError: 429 Client Error: Too Many Requests for url: https://westus.api.cognitive.microsoft.com/vision/v2.1/analyze?visualFeatures=Description,Tags&su

bscription-key=aef101317b7445c6b07166ccb42a0fdb ,waited a couple minutes and then re-ran (unlike google and rekognition, microsoft needed to run in several instances)

- 8:27 am - finished running cloudy_vision.py for microsoft



File | C:/Users/gohansrealdad/Desktop/cloudy_vision-master/output/output.html

A thumbnail of each image is displayed (click to see it fullsize) along with results from each computer vision API vendor.
Statistics on response times and more are at the bottom of the page.

T001.jpg

rekognition	View raw json (1.63s)
rekognition_tags	Nature (.99) , Outdoors (.98) , Animal (.94) , Bird (.94) , Snow (.83) , Winter (.75) , Tree (.70) , Plant (.70) , Sea (.60) , Water (.60) , Ocean (.60) , Ice (.60) , Weather (.59) , Slope (.57) , Landscape (.57) , Storm (.56)
msft	View raw json (2.87s)
msft_tags	fog (.98) , snow (.93) , outdoor (.92) , water (.86) , nature (.80)
msft_captions	a bird flying over the water (.26)
google	View raw json (1.33s)
google_tags	Sky (.96) , Atmosphere (.84) , Calm (.76) , Sea (.73) , Horizon (.65) , Screenshot (.60) , Ocean (.57)
clarifai	View raw json (1.82s)
clarifai_tags	

T002.jpg

rekognition	View raw json (1.36s)
rekognition_tags	Nature (.95) , Leisure Activities (.95) , Adventure (.95) , Animal (.93) , Flying (.93) ,

Type here to search

8:25 AM
11/5/2019

Will begin logging microsoft image tags/results into spreadsheet

11/06/2019

- 10:00 am - typed up and sent second interim report to Professor Mandel to look over and sign (to be submitted 11/11/2019)
- 4:15 pm - met with Professor Mandel via Google Hangouts to discuss progress so far, discussed the idea to possibly write code to compare and evaluate APIs
- 6:07 pm - Began writing test Java code to read the information into arrays and compare the results

11/07/2019

- 6:30 pm - continued working on Java code to compare results, created class to create objects for each image recognition API that the user would want to evaluate

11/10/2019

- 3:30 pm - continued manual evaluation of test images, attempted to read manual results into a hashmap with the image file name as the key and the tags as value(s). Changed from HashMap to LinkedHashMap so that data is in sequence of input

11/11/2019

- 8:30 pm - started logging clarifai tag results into spreadsheet

11/14/2019

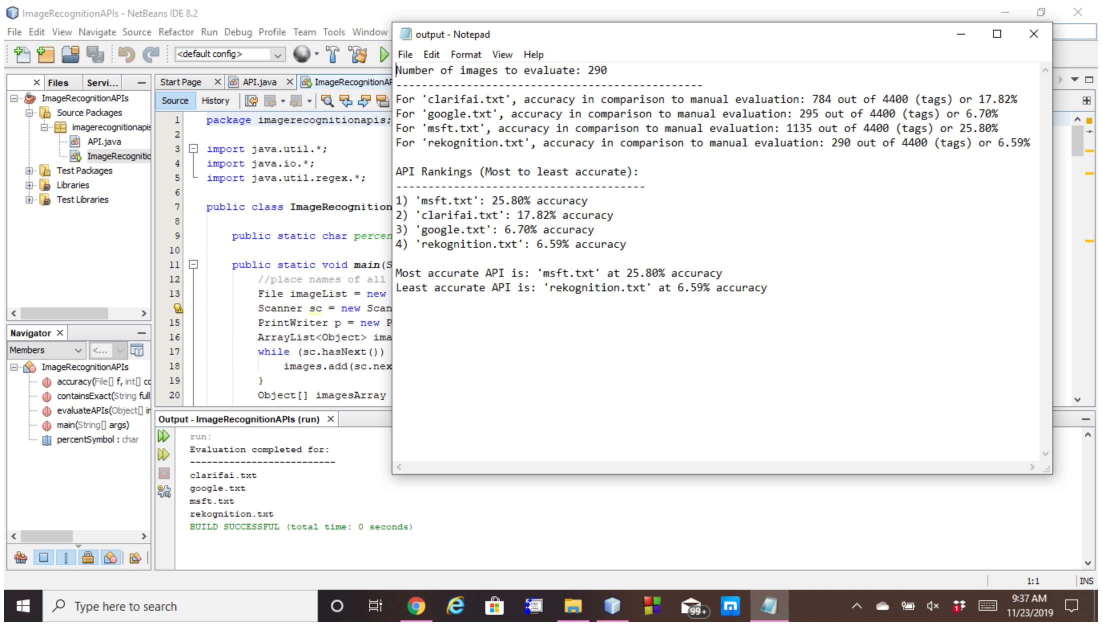
- 7:45 am - resumed working on Java program, attempting to find a way to store the manual results into a data structure like a LinkedHashMap where the image name can be the key and the descriptive tags would be the values
- 10:12 pm - wrote code to iterate through each .txt file of data and compare to manual results found in LinkedHashMap. Only error is that the count is always set to the last API evaluated, probably error within the for loop

11/18/2019

- 9:43 am - fixed java code so that the count is not fixed to one value for each API evaluated by changing the order of the loops so that it works clearly now and computes the accuracy results of each API and compares those results to display both the most and least accurate. Will continue logging clarifai results to test

11/22/2019

- 8:55 am - updated java code so that instead of asking for the number of files/APIs to evaluate, it runs through the number of .txt files in the 'vendors' folder and evaluates all of them (no need for the user to do more work than necessary)



```
package imagerecognitionapis;

import java.util.*;
import java.io.*;
import java.util.regex.*;

public class ImageRecognition

    public static char percen

    public static void main(S
        //place names of all
        File imageList = new
        Scanner sc = new Scan
        PrintWriter p = new P
        ArrayList<Object> ima
        while (sc.hasNext())
            images.add(sc.nex
        }
        Object[] imagesArray
```

```
run:
Evaluation completed for:
--------------------------
clarifai.txt
google.txt
msft.txt
rekognition.txt
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
Number of images to evaluate: 290
-----------------------------------------------
For 'clarifai.txt', accuracy in comparison to manual evaluation: 784 out of 4400 (tags) or 17.82%
For 'google.txt', accuracy in comparison to manual evaluation: 295 out of 4400 (tags) or 6.70%
For 'msft.txt', accuracy in comparison to manual evaluation: 1135 out of 4400 (tags) or 25.80%
For 'rekognition.txt', accuracy in comparison to manual evaluation: 290 out of 4400 (tags) or 6.59%

API Rankings (Most to least accurate):
--------------------------------------
1) 'msft.txt': 25.80% accuracy
2) 'clarifai.txt': 17.82% accuracy
3) 'google.txt': 6.70% accuracy
4) 'rekognition.txt': 6.59% accuracy

Most accurate API is: 'msft.txt' at 25.80% accuracy
Least accurate API is: 'rekognition.txt' at 6.59% accuracy
```

11/23/2019

- 9:45 am - emailed Professor Mandel about my current progress, would like to modify the project to write to an .html file and use the JFreeChart library to display a chart that illustrates these results visually and then use that data to write up my final report

11/26/2019

- 1:00 pm - met with Professor Mandel for status update and to drop off evaluation forms

12/02/2019

- 8:05 pm - installed apache commons IO and added to java project to begin creating html file

12/03/2109

- 8:45 am - successfully transferred and arranged output text results to html file, so that it displays the number of images evaluated, the number of matching tags/percent accuracy per API and ranks APIs based on accuracy in descending order and identifies the most and least accurate. Will now proceed to implement JFreeChart in order to display a visual representation to accompany data.
- 10:15 pm - successfully created bar chart to represent statistics visually, convert to .png and display .png on .html file

Amara Auguste, CISC 4900, Fall 2019

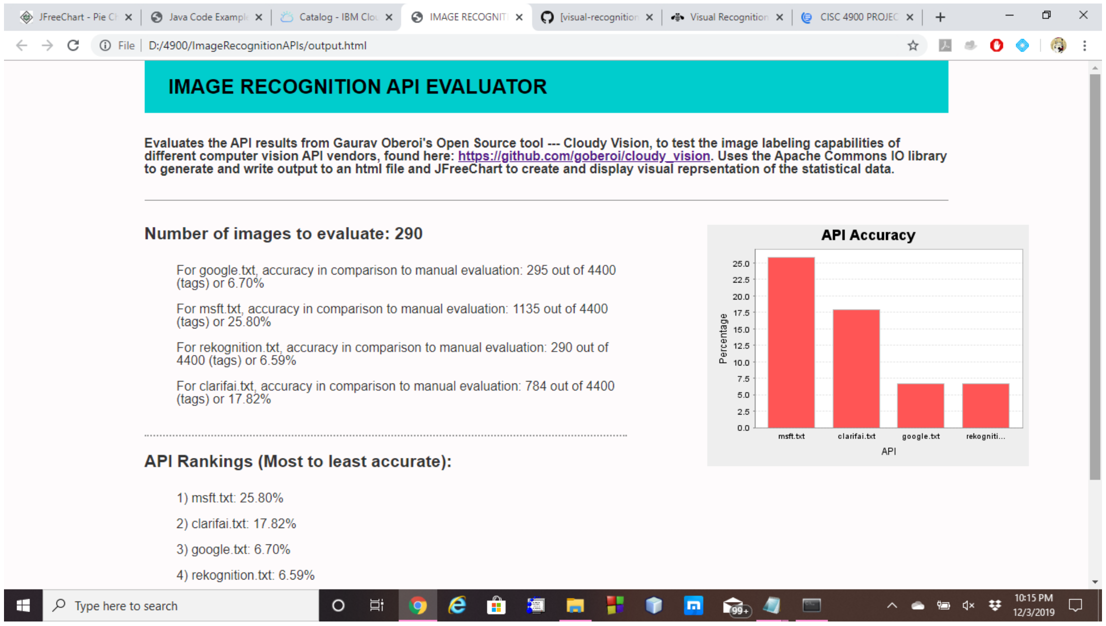


IMAGE RECOGNITION API EVALUATOR

Evaluates the API results from Gaurav Oberoi's Open Source tool --- Cloudy Vision, to test the image labeling capabilities of different computer vision API vendors, found here: https://github.com/goberoi/cloudy_vision. Uses the Apache Commons IO library to generate and write output to an html file and JFreeChart to create and display visual reprsentation of the statistical data.

Number of images to evaluate: 290

For google.txt, accuracy in comparison to manual evaluation: 295 out of 4400 (tags) or 6.70%

For msft.txt, accuracy in comparison to manual evaluation: 1135 out of 4400 (tags) or 25.80%

For rekognition.txt, accuracy in comparison to manual evaluation: 290 out of 4400 (tags) or 6.59%

For clarifai.txt, accuracy in comparison to manual evaluation: 784 out of 4400 (tags) or 17.82%

API Rankings (Most to least accurate):

1) msft.txt: 25.80%
2) clarifai.txt: 17.82%
3) google.txt: 6.70%
4) rekognition.txt: 6.59%

12/06/2019

- 9:45 pm - added method to java code to compute the average number of tags per image provided by each tested API and create graph displaying that data

12/13/2019

- 10:15 am - modified manual evaluation text file to have more concise tags and no use function words (articles, conjunctions, etc.) since none of the APIs would generate them, bringing total tags down from 4400 to 2850 and calculating a better scaled form of accuracy but overall not changing the fact that microsoft has the most matches and AWS rekognition has the least.

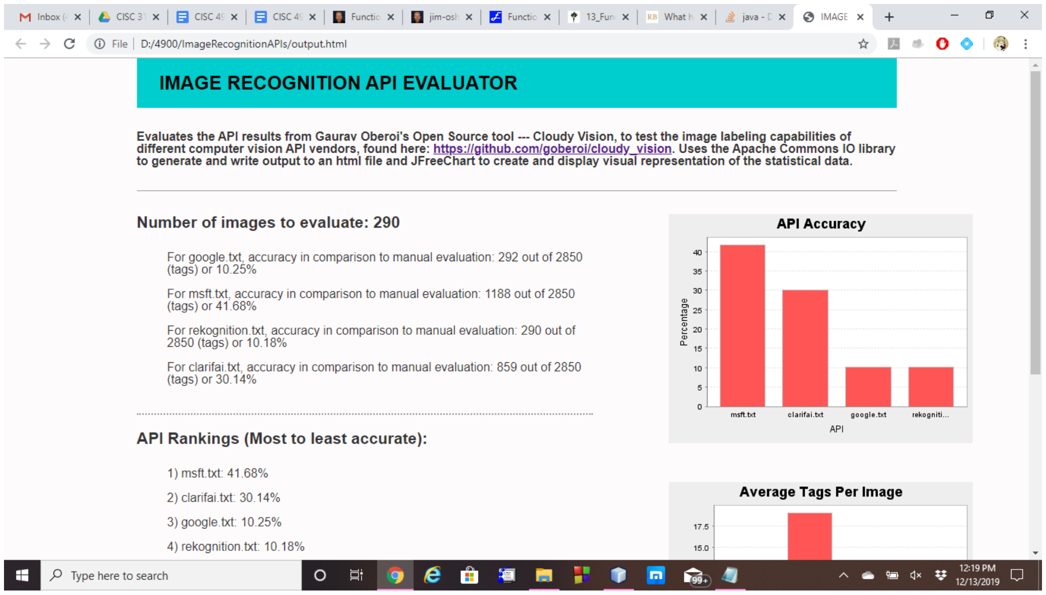


IMAGE RECOGNITION API EVALUATOR

Evaluates the API results from Gaurav Oberoi's Open Source tool --- Cloudy Vision, to test the image labeling capabilities of different computer vision API vendors, found here: https://github.com/goberoi/cloudy_vision. Uses the Apache Commons IO library to generate and write output to an html file and JFreeChart to create and display visual representation of the statistical data.

Number of images to evaluate: 290

For google.txt, accuracy in comparison to manual evaluation: 292 out of 2850 (tags) or 10.25%

For msft.txt, accuracy in comparison to manual evaluation: 1188 out of 2850 (tags) or 41.68%

For rekognition.txt, accuracy in comparison to manual evaluation: 290 out of 2850 (tags) or 10.18%

For clarifai.txt, accuracy in comparison to manual evaluation: 859 out of 2850 (tags) or 30.14%

API Rankings (Most to least accurate):

1) msft.txt: 41.68%
2) clarifai.txt: 30.14%
3) google.txt: 10.25%
4) rekognition.txt: 10.18%

H. **Personal Evaluation**: Overall, I feel like I have learned a lot from working on this project. Due to this project, I have now learned, at least the basics of the Python programming language, plus I have both refreshed and advanced my current skills in Java by using not only what I had learned directly from my courses at the college but outside libraries/information such as Apache Commons IO and JFreeChart. In working on this project, I learned the importance of time management (through splitting up my work effectively to meet deadlines), studying and implementing programming languages, and planning/outlining the necessary tasks needed to successfully execute a project. It was very important for me to make time in my schedule to complete tasks by specific deadlines and keep in contact with my supervisor(s) pertaining to my progress. I also learned a lot about how to properly research information because whenever I reached an obstacle/error along the way and could not meet with my supervisor(s) directly, I had to solve the problem on my own with the help of research. The project definitely helped me gain a better understanding of some of the work entailed in the field of computer science, which is not just limited to programming. I feel that this project for this type of course taught me the importance of not only writing code but testing and debugged code, logging/recording data, writing reports, and reporting to my supervisor(s) in time to meet important time constraints which will help me, and other students like me, effectively transition into the workforce, post graduation.