CSE 546 Cloud Computing
Plagiarism check on Image Assignment Submissions

Abhijith Shreesh (1213204276 - ashreesh@asu.edu)
Abhinethri Tumkur Umesh (1213187714 - atumkuru@asu.edu)
Channabasava Gola (1213222619 - cgola@asu.edu)

1. Problem Statement

1.1. Motivation

In any educational institution, it is required to perform the plagiarism check on the submitted documents (Mainly assignments and projects). But it is difficult to perform the plagiarism check when submission is handwritten (i.e. a hard-copy). With this application, we intend to make text search on documents such as large repositories of handwritten assignments/projects and provide the list of student names and the corresponding plagiarism percentage. This application can also be used by students before submitting the assignment(s).

In order to know whether a particular student has copied someone else works, this software will be helpful. Professor/Grader/TA/Student can take the picture of the assignments and upload them to our GAE website. After uploading, plagiarism check will be performed and a list of top 5 students along with the plagiarism percentage will be displayed. This information will help the graders to know the list of students who have violated academic integrity.

2. Design and Implementation

2.1. Architecture

Components

Google App Engine (GAE):

This is a web framework from Google which has been used in our project to deploy the web application that has been developed using Flask. This platform auto scales when the number of requests increases. Configuring parameters such as target CPU utilization, Requests per second (RPS), the capacity of the load balancer effect the autoscale behavior. Autoscaling has been explained in detail in section 2.2.

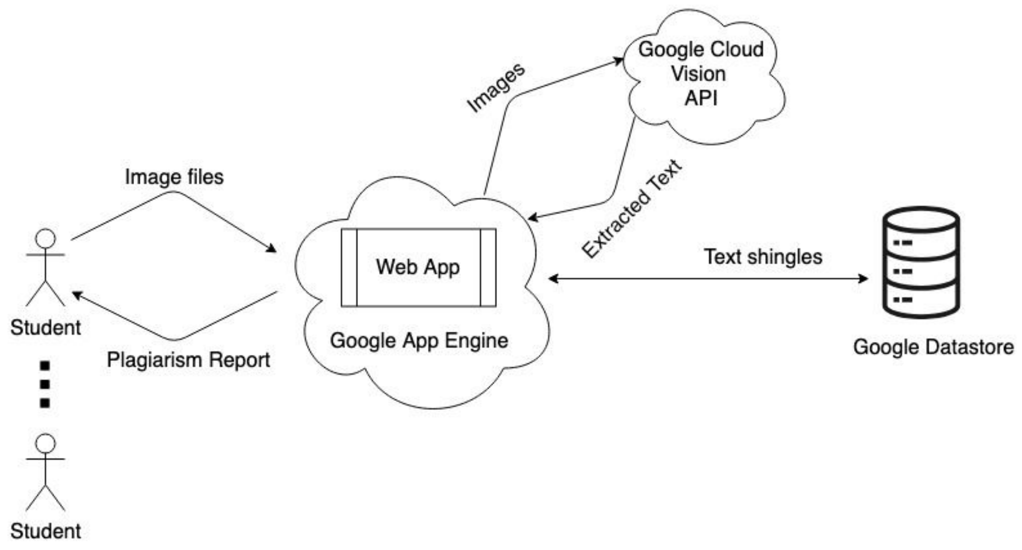


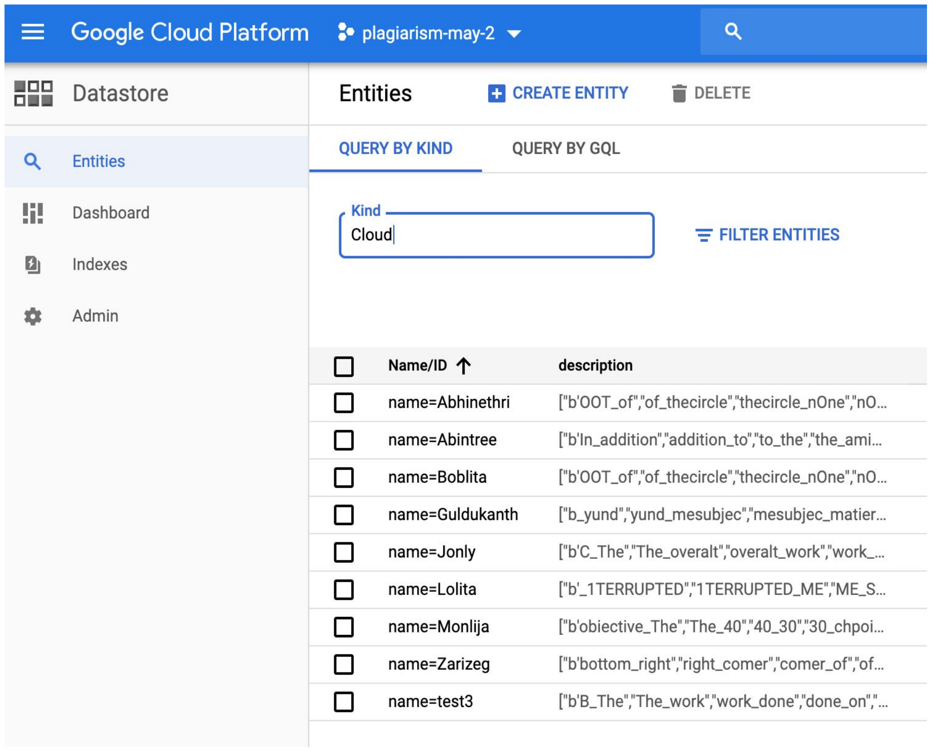
Fig 1. System architecture

Google cloud vision API:

This API renders strong machine learning models which can be accessed through REST/Remote procedure API calls. These APIs have the capability to detect objects, build metadata, identify the position within an image. In our project, we are using this API for document text detection which performs recognition of the characters and then extracts the data.

Google Datastore:

It is NoSQL as a service which is highly scalable and reliable. This has been built on top of Bigtable (Google's). The data has been stored in the form of key-value pair. { Subject name, Student name } forms the key and shingles extracted from the text present in the image forms values. You can query the data store based on the subject name.



Name/ID ↑	description
name=Abhinethri	["b'OOT_of","of_thecircle","thecircle_nOne","nO…
name=Abintree	["b'In_addition","addition_to","to_the","the_ami…
name=Boblita	["b'OOT_of","of_thecircle","thecircle_nOne","nO…
name=Guldukanth	["b_yund","yund_mesubjec","mesubjec_matier…
name=Jonly	["b'C_The","The_overalt","overalt_work","work_…
name=Lolita	["b'_1TERRUPTED","1TERRUPTED_ME","ME_S…
name=Monlija	["b'obiective_The","The_40","40_30","30_chpoi…
name=Zarizeg	["b'bottom_right","right_comer","comer_of","of…
name=test3	["b'B_The","The_work","work_done","done_on","…

Fig 2. sample data in DataStore

Technologies and tools:

1. Bootstrap
2. JavaScript
3. HTML
4. CSS
5. Flask (Python)
6. Gunicorn
7. Postman and curl

Challenges:

The biggest challenge was to find the right documentation for Google App Engine. We went through different rounds of setup referring to different documentation, but nothing seemed to work. At last, we found one guide that worked and we deployed our application. Furthermore, we also tried different methods of producing plagiarism metrics with the help of nltk. Deciding the best scoring model was another challenge that we faced. We selected our current technique based on trial and error.

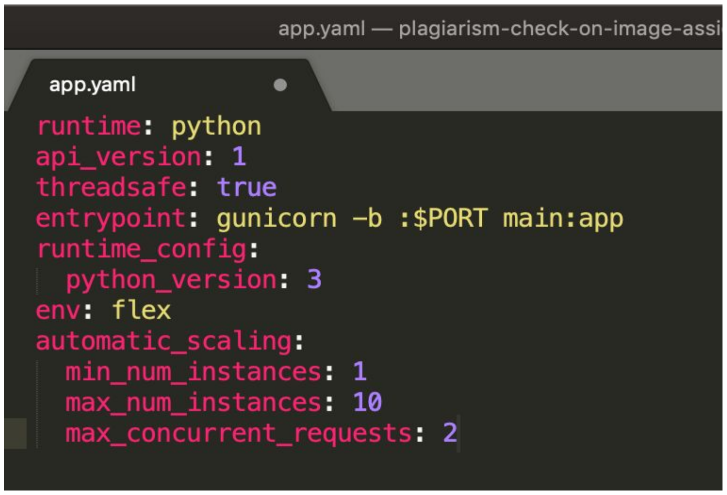
Future Work:

This project was built in mind with a startup in mind. This is just a proof of concept. As of now, this serves students, given time and resource this can be extended to serve professors. This project needs login authentication to classify professor and student. This application can be modified to return the actual text and also work on the frontend to display the image submitted. This application currently works for a single image but can be extended to work for multiple images in a given assignment.

2.2. Autoscaling

This is the hardest part because if it scales slowly then it affects the performance of the application and if it scales at a high rate then it consumes more cloud resources then expected and costs in terms of money. Every instance in GAE has its own queue for the requests and it is monitored by app engine.

The app.yaml helps us configure the environment while deploying the application. The maximum and a minimum number of instances that can be used by the application at any point in time. The auto-scaling can be configured to depend on different factors. The parameters such as minimum instances, maximum instances, CPU and throughput utilization can be set for autoscaling if cloud SDK has been used for deployment. This cannot be used if appcfg tool is used for deployment.



```yaml
runtime: python
api_version: 1
threadsafe: true
entrypoint: gunicorn -b :$PORT main:app
runtime_config:
  python_version: 3
env: flex
automatic_scaling:
  min_num_instances: 1
  max_num_instances: 10
  max_concurrent_requests: 2
```

Fig 3. App.yaml

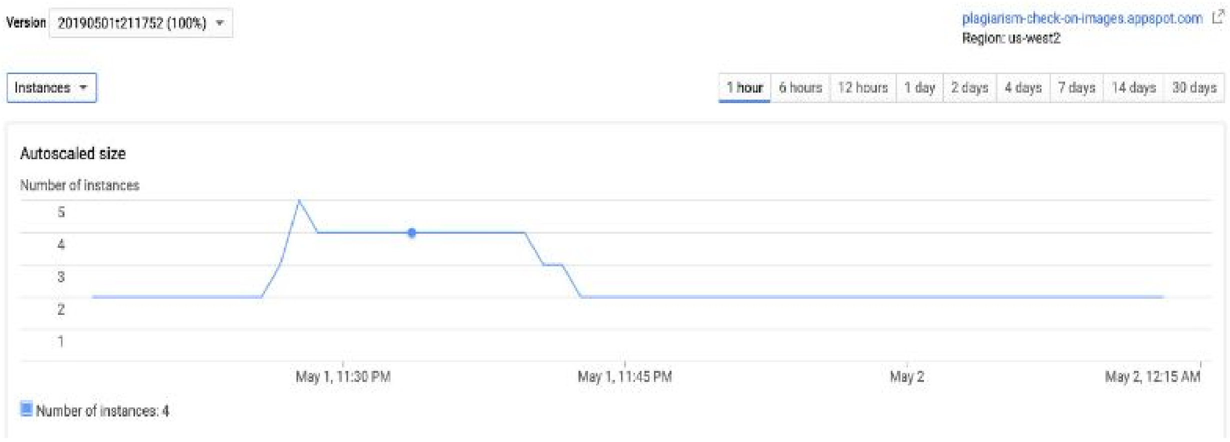
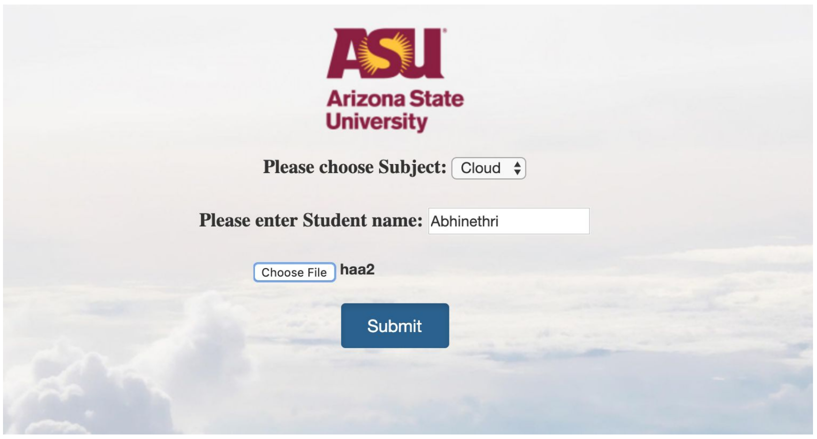


Fig 4. Autoscaling of instances

3. Testing and evaluation

Input:

Course name: Name of the course for which assignment is being submitted.
Student name: Name of the student as in the records.
Image document: Image of handwritten assignment or hard-copy submission.
All the inputs are mandatory so user can't skip any of the inputs.



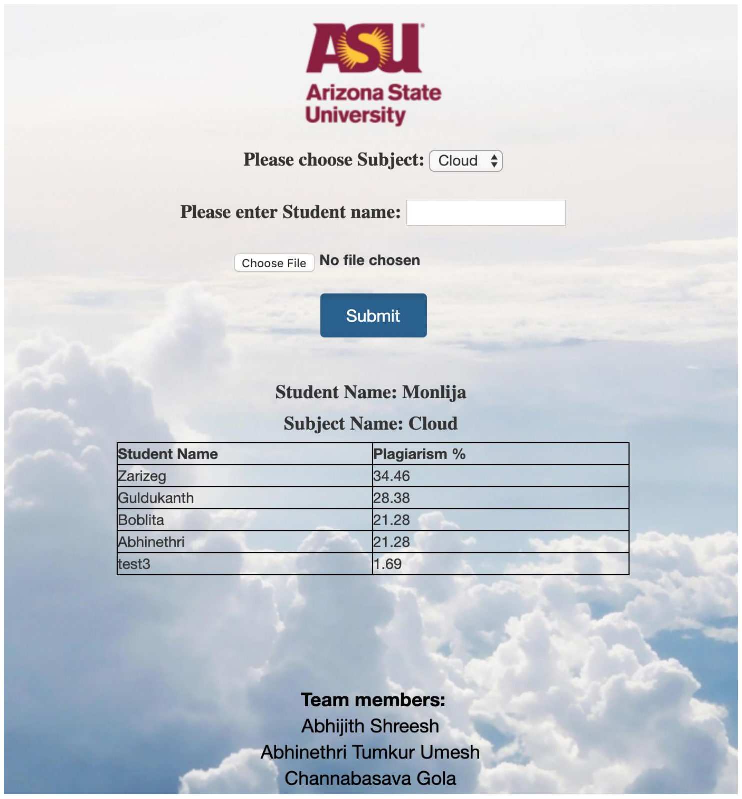
Please choose Subject: Cloud

Please enter Student name: Abhinethri

Fig 5. Input

Output(s):

The output contains a list of top 5 students who have similar content as in the assignment submitted by the current student.



Please choose Subject: Cloud

Please enter Student name:

Student Name: Monlija

Subject Name: Cloud

Student Name	Plagiarism %
Zarizeg	34.46
Guldukanth	28.38
Boblita	21.28
Abhinethri	21.28
test3	1.69

Team members:
Abhijith Shreesh
Abhinethri Tumkur Umesh
Channabasava Gola

Fig 6. Output

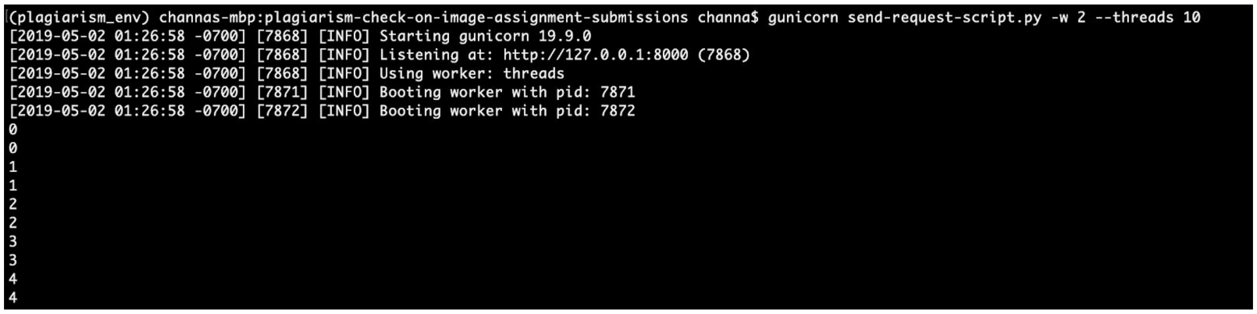
Above are the sample outputs of plagiarism check for few of the user inputs. It displays the top 5 results for the uploaded assignment by performing the plagiarism check.

Testing:

For the purpose of testing, following python script has been developed:

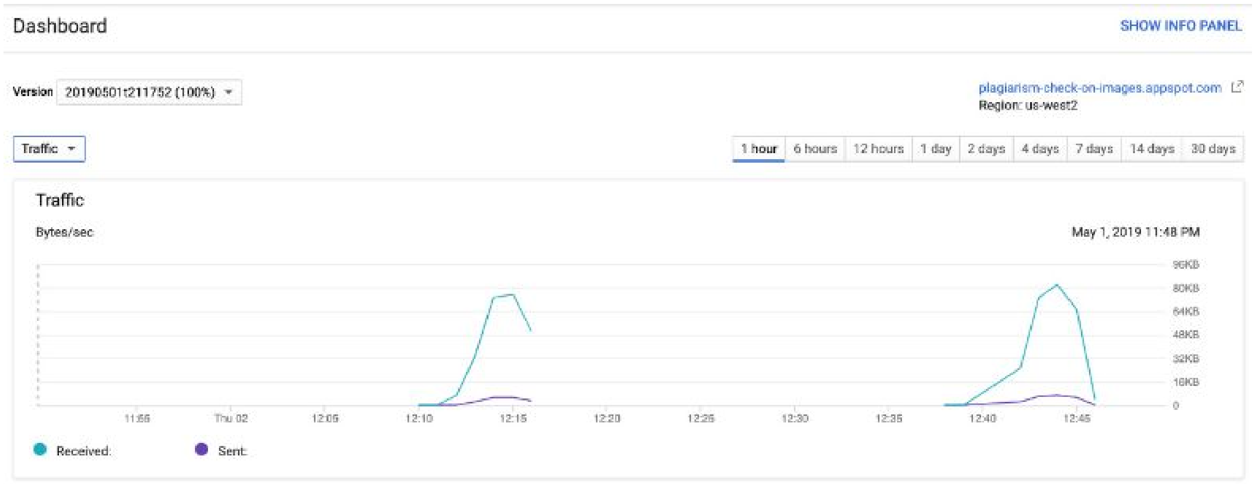
```
import requests
url = "https://plagiarism-check-on-images.appspot.com"
url = "https://plagiarism-may-2.appspot.com"
files = {'assignment-file': open('/Users/channa/Desktop/text_3.png','rb')}
for i in range(5000):
      if i % 10 == 0:print(i)
      values = {'subject-name': 'MC', 'student-name': 'test' + str(i)}
      r = requests.post(url, files=files, data=values)
```

This script sends 5000 requests sequentially to our web application. In order to send concurrent requests, 2 processes and 10 threads have been created using gunicorn. The multiple requests make the GAE app to scale.



```
(plagiarism_env) channas-mbp:plagiarism-check-on-image-assignment-submissions channa$ gunicorn send-request-script.py -w 2 --threads 10
[2019-05-02 01:26:58 -0700] [7868] [INFO] Starting gunicorn 19.9.0
[2019-05-02 01:26:58 -0700] [7868] [INFO] Listening at: http://127.0.0.1:8000 (7868)
[2019-05-02 01:26:58 -0700] [7868] [INFO] Using worker: threads
[2019-05-02 01:26:58 -0700] [7871] [INFO] Booting worker with pid: 7871
[2019-05-02 01:26:58 -0700] [7872] [INFO] Booting worker with pid: 7872
0
0
1
1
2
2
3
3
4
4
```

Fig 7. Gunicorn to spawn 10 threads



Dashboard

Version 20190501t211752 (100%)

plagiarism-check-on-images.appspot.com
Region: us-west2

Traffic

Fig 8. Increase in the traffic

4. Code

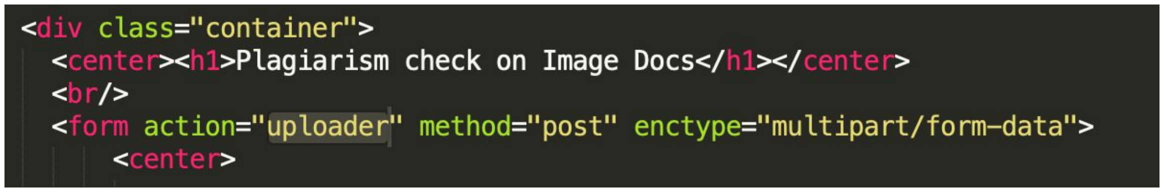
Complete process:

The user accesses our website by hitting the GAE URL https://plagiarism-may-2.appspot.com/. After accessing the URL, he/she has to submit the subject name, student name, and image of the assignment as the input. This invokes the python script running on GAE which will in turn process the request and make appropriate API calls to calculate the results. The result is rendered back to the user using HTML. This application can be accessed from both mobile and desktop.

Front end:

The frontend has been rendered using HTML elements, CSS and bootstrap. Bootstrap is an open source framework with pre-defined classes, templates, and other interface elements. index.html has the text box, dropdown and file tags to receive the inputs from the user. Input data will be sent to the backend for processing through a POST request. The results are rendered in the form of the table in the same index.html file.

Below is the snippet from index.html:



```
<div class="container">
  <center><h1>Plagiarism check on Image Docs</h1></center>
  <br/>
  <form action="uploader" method="post" enctype="multipart/form-data">
      <center>
```

Backend:

Backend is developed using Flask microframework for Python. It is based on Jinja 2 and Werkzeug. It has built-in debugger, development server and supports secure cookies. Our application is developed in a RESTful architecture style. We choose Python as it is widely used high-level language and is one of the best tools for text processing which is extensively done in our project. Python and Flask have good support from Google App Engine with detailed documentation for the deployment of the application.

We use gunicorn to start the application in multiple workers and threads. Our applications load index.html on load. And on submitting with 3 mandatory parameters, i.e., subject name, student name and assignment submission in image /uploader endpoint is called. /uploader is a POST end-point. The application first saves the image. Then calls cloud vision API and gets the text from the image. Then, the text is pre-processed, it is converted to shingles. Shingles can be of any length. Our proof of concept is built with 3 grams. We store this shingled data to google datastore against the subject name and student name. Then for this subject, the application queries prior submissions from google datastore. Then compare and calculate the plagiarism value against each prior submission. Plagiarism is the number of shingle matches over the total number of shingles in this submission. We store the plagiarism value against each prior submission in a heap and return the top 5 largest value for plagiarism values from the heap and the student name of prior submissions. We lastly delete the image from the temporary storage for minimal space usage.

Deployment and execution:

The application was first constructed in a Linux based environment and locally tested. After struggling to find the right documentation from Google Cloud Platform, we stumbled upon a document that enabled us to deploy our application on Google App Engine. The following commands detail the steps involved in this process:

1. git clone https://github.com/channagithub/plagiarism-check-on-image-assignment-submissions.git
2. cd plagiarism-check-on-image-assignment-submissions/
3. virtualenv --python=python env
4. cd plagiarism-check-on-image-assignment-submissions/
5. pip install -r requirements.txt
6. gcloud auth list project plagiarism-check-on-images
7. gcloud config list project
8. gcloud services enable vision.googleapis.com
9. gcloud services enable storage-component.googleapis.com
10. gcloud services enable datastore.googleapis.com
11. export PROJECT_ID=plagiarism-check-on-images
12. gcloud iam service-accounts create codelab --display-name "Plagiarism May 1 deployment"
13. gcloud projects add-iam-policy-binding ${PROJECT_ID} --member serviceAccount:codelab2@${PROJECT_ID}.iam.gserviceaccount.com --role roles/owner
14. gcloud iam service-accounts keys create ~/key.json --iam-account codelab2@${PROJECT_ID}.iam.gserviceaccount.com
15. export GOOGLE_APPLICATION_CREDENTIALS="/home/${USER}/key.json"
16. gcloud app create
17. gcloud app deploy

References

- http://flask.pocoo.org/
- https://en.wikipedia.org/wiki/Google_Cloud_Datastore
- https://cloud.google.com/appengine/
- https://en.wikipedia.org/wiki/Plagiarism_detection
- https://en.wikipedia.org/wiki/N-gram