Group Members: Connor Taylor, Jonathan Gilad, Brian Cefali

Team Name: Reap-Post - You reap what you post

Project Idea: Reap-post is a reddit tool that finds duplicate reddit image posts and analyzes their performance (Karma). We will build a simple site that allows users to input a reddit image link and a specific subreddit. We will then take that data and see if that image has already been posted in that subreddit within the past 30 days. We will then make a visualization of how the identical image posts rank against each other.

Minimum Deliverable:

The minimum that we will do is have a basic webpage set up for user input. Our program will scrape reddit for the past 30 days (might be smaller based on rate limiting) of data. Only reddit posts with imgur URLs will be handed off to the image analyzer. The image analyzer will compare two images and find if they are identical. This will be done with multiple processes. After analyzing all of the reddit posts, our website will display the results to the user.

Maximum Deliverable: Time permitting, we would like to expand the core functionality. Specifically, we would like to mine reddit for data over time to increase our sample size, add text analysis in addition to image analysis in order to map how differing text affects the performance of a post, and potentially making this app a plugin for reddit or Chrome as opposed to a standalone site.

First Step: The first step will be to learn how to use the Reddit API with python. Once we can get the data that we need, we can begin to work on the real meat of the project, the image analyzer.

Biggest Problem: Staggering our network requests to comply with reddit's rate limiting and still testing a large enough set of data for the concurrency to make a difference

System Design

1. Reddit Scraper:
    a. Uses the Reddit API (or a python library for the Reddit API like PRAW)
    b. Scrapes the specified subreddit for new content
    c. If a post has an imgur link, we pass it off to the image analyzer
    d. Otherwise, we ignore the post
    e. **Important note**: We are only going to scrape small subreddits in real time. Larger subreddits will have pre-scraped data (time permitting), as it will take too long to scrape them with Reddit's limits on request rates and returned data
    f. Sample code for flow:

```
def main_loop():
      user_input = get_user_input()

      for day in range(0,29): // where i is the age of the posts
            data = scrape_reddit(day, dispatch_img_procs)

def dispatch_img_procs(data):
      for post in data:
            if has_imgur_link(post):
                  pid = spawn_image_processer(post)
```

2. Image Analyzer:
    a. Takes in imgur links, obtain image, and compare it to the input (Original) post
    b. Image comparison will be "smart"
        i. This is going to be the most computationally intensive part of the system
        ii. We don't want to test pixel by pixel for each image as that wastes a lot of time
        iii. First, we check the urls of the images. If they are the same, we know that the images are equal. Second, check dimensions. If they aren't the same, reject the link.
        iv. After the initial tests, we test small subset of randomly chosen pixels for equality. If they are not equal within 95%, we discard it.
        v. If an image has made it this far, then we do a pixel by pixel analysis.
        vi. As there may be slight variations, we will allow any image that has a 95% similarity rate (due to different image compression; subject to change as we learn more).
        vii. Any matching images will be saved, along with the rest of the reddit-post data, and returned.
    c. Image comparison will be done concurrently with Python multiprocessing
        i. A couple threads will read in the links from the text file and spawn threads to process them

ii. Most threads will be image processing threads that test images for equality
iii. One thread will listen for updates to the repost count, files processed, and files checked to update the site in real time

d. Sample code for flow:

```
// returns 1 for match, 0 for no
def process_image(original_image, post):
     if original_image.link == post.imgur_link:
          return 1
     new_image = pull_from_imgur(post.imgur_link)
     if new_image.dimensions != orignal_image.dimensions:
          return 0
     for i in range(0, RAND_PIXELS):
          orig_pixel = get_rand_pixel(original_image)
          test_pixel = get_rand_pixel(new_image)

          // Will account for variance later
          if orig_pixel != test_pixel:
               return 0

     // images match sufficiently, run pixel by pixel analysis
     return full_image_test(original_image, test_image)
```

3. Website for UI/UX:
    a. Simple Flask site to display returned data
    b. Duplicate posts will be compared side by side
        i. Still not 100% sure what we want to display, but some possibilities are:
            1. Post title
            2. Date submitted
            3. Post upvotes
            4. Number of comments
            5. Comment upvotes
            6. Total Karma (total upvotes for post/comments)
    c. Instantiates Reap-Post Python instance via PHP's `exec()`

4. Tools/Libraries Used:
    a. Python Multiprocessing Module (not multithreading module)
    b. PIL
    c. Reddit API or Reddit API wrapper

Development Plan

1. Nov 3rd - Nov 10th:
    a. Set up Reddit Scraper (both live version with timeout and cron-job version for data accumulation)
    b. Set up basic site for grabbing user input
2. Nov 10th - Nov 17th:
    a. Implement basic image processor module
        i. Steps iii, iv, v listed above
3. Nov 17th - Nov 24th:
    a. Finalize image processor and begin connecting modules together
        i. Debugging and tuning the allowed variance
        ii. Optimizing the concurrent model
        iii. Latency hiding for live-scrape use cases
    b. Implement data visualization
4. Nov 24th - Dec 1st:
    a. Debug
    b. Prepare presentation
    c. Make data visualization happen in real-time
        i. Show the number of pulled links vs. checked links
        ii. Show duplicates as they appear

REAP-POST MODULE DIAGRAM

Web UI

- Takes user request for the post to search for and the subreddit in which to search
- Starts a reap-post python instance w/ the requisite info
- Receives and displays reap-post results

Run via PHP's exec()

Results buffered in temp file, (displayed on completion)

Reap-Post Python Prgm.

- Receives user data from web UI and instantiates the scraper/analyzer processes
- Compiles results from child processes and sends the final result to the Web UI component

Result of comparison via pipe

Main proc. dispatches image analyzer workers

Image Analyzer Process

- Given the original image and an imgur link, compare the two
- Image comparison is done w/ the following steps:
  1. URL equality check
  2. Image dimension check
  3. N random pixel check
     → 95%* equality tolerance
  4. Pixel by pixel comparison
     → 95%* tolerance again
- Results sent to R-P Py Prgm. main process via pipe

*Threshold subject to change

Reap-Post

- Brian Cetoli
- Johnathan Gilad
- Connor Taylor