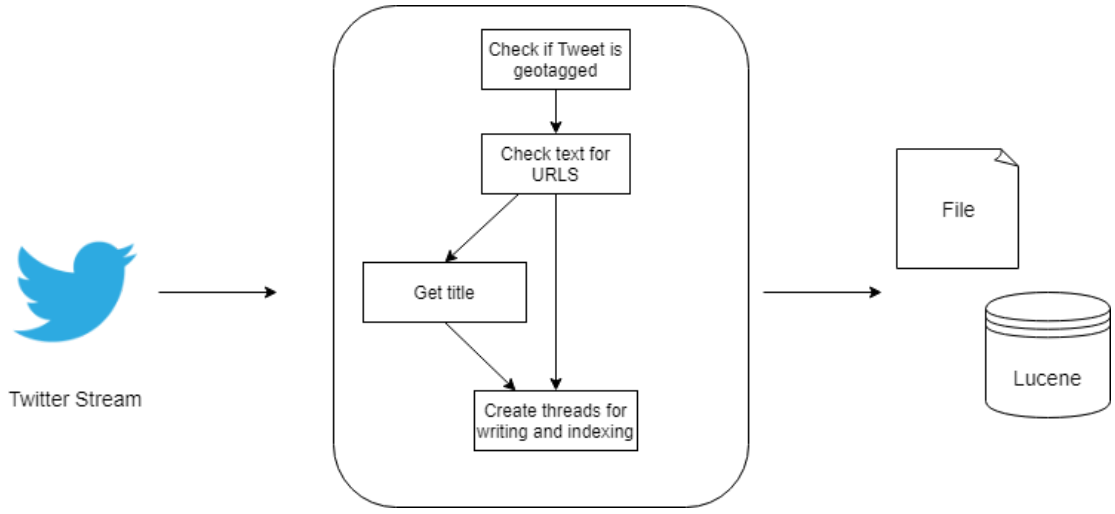
Ryan Gray
Jesse Reyes
Joshua Riley
Melissa Santos

CS172 Final Project Report

Part 1 - Crawler

1. Collaboration Details
    - Melissa: Obtained auth keys. Made the code for connecting to Twitter Stream, coordinate checking, and webpage title retrieval. Implemented threading.
    - Jesse: Created live Tweet indexing function. Came up with the threading idea to minimize data loss.
2. System Overview
    - Architecture



Data collection strategy

i. First, we connect to the Twitter Stream and sample real time Tweets. If a Tweet has a ‘coordinates’ field, we write it to file. If the Tweet’s ‘text’ field has a valid link, we create a ‘title’ field for the Tweet object and set the value to the webpage’s title. If a Tweet has more than one url, we take the title of the last url. Then spawn a threads to index and write the Tweet. If the program disconnects, it’ll wait 20 seconds before attempting to reconnect. The program will exit after three unsuccessful reconnect attempts.

ii. We also implemented indexing live geotagged Tweets. We create another thread for the indexing to minimize data loss.

    - Data Structures employed
        i. JSON
3. Limitations

- We only accept Tweets that are tagged with exact location. Each Tweet's title is retrieved as it comes in, so we can experience data loss if there are a lot of incoming geotagged Tweets.

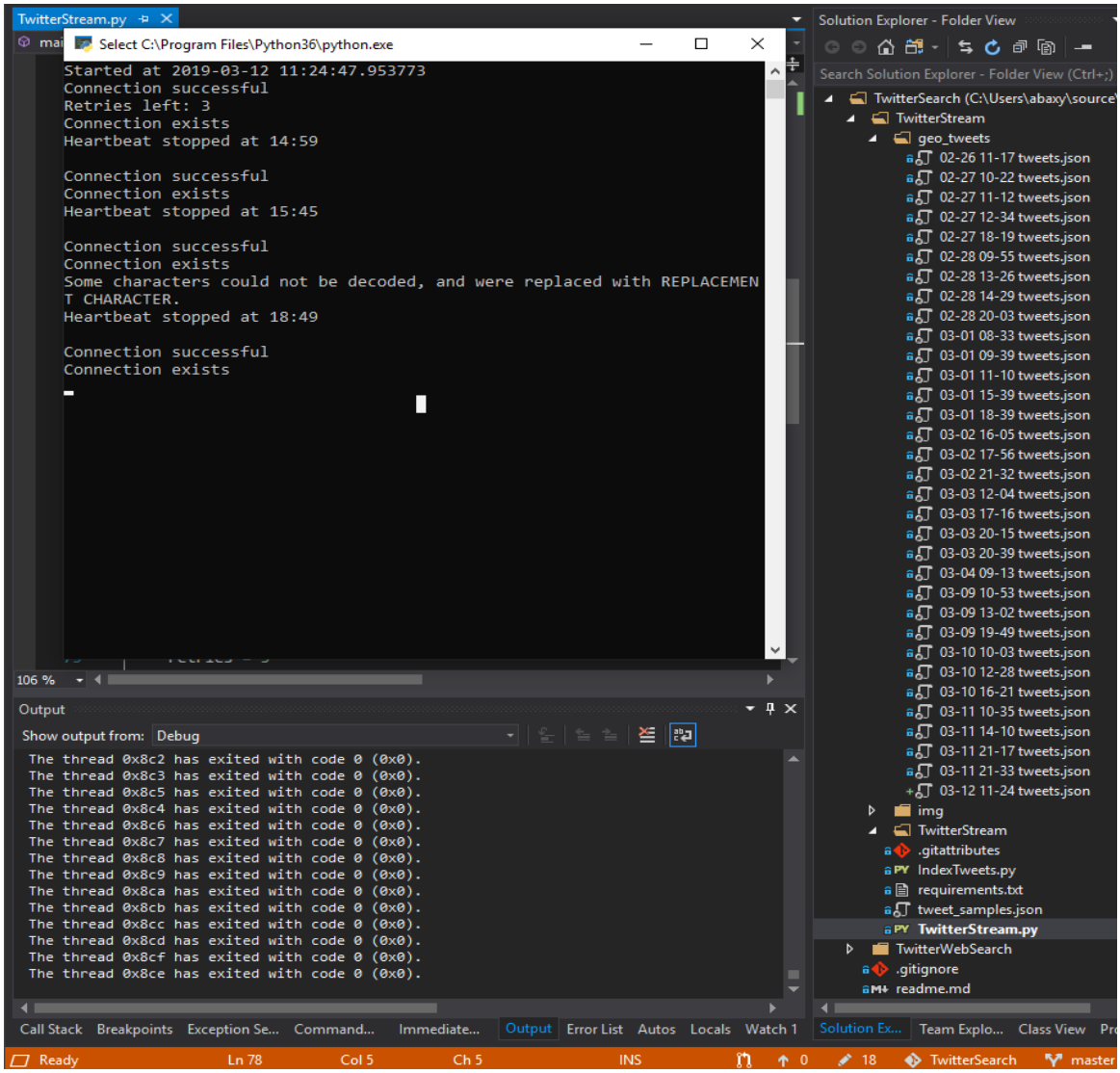
4. How to deploy
    1. Download/clone code from Github
    2. Edit the 'TwitterStream.py' file in the TwitterStream directory and input keys and tokens:

        ```
        consumer_key = <API key>
        consumer_secret = <API secret key>
        token = <Access token>
        token_secret = <Access token secret>
        ```

    3. Have Python 3.X and install the items in 'requirements.txt'
    4. Open the command line, navigate to the TwitterStream directory, and run:

        ```
        $ python TwitterStream.py
        ```

    5. Stream should connect and will automatically start indexing and writing to file geotagged Tweets.
5. Screenshots



Pictured: The debug terminal showing that the indexing and writing threads are successfully terminating. Command prompt showing that stream is reconnecting after disconnects.

Part 2 - Indexer

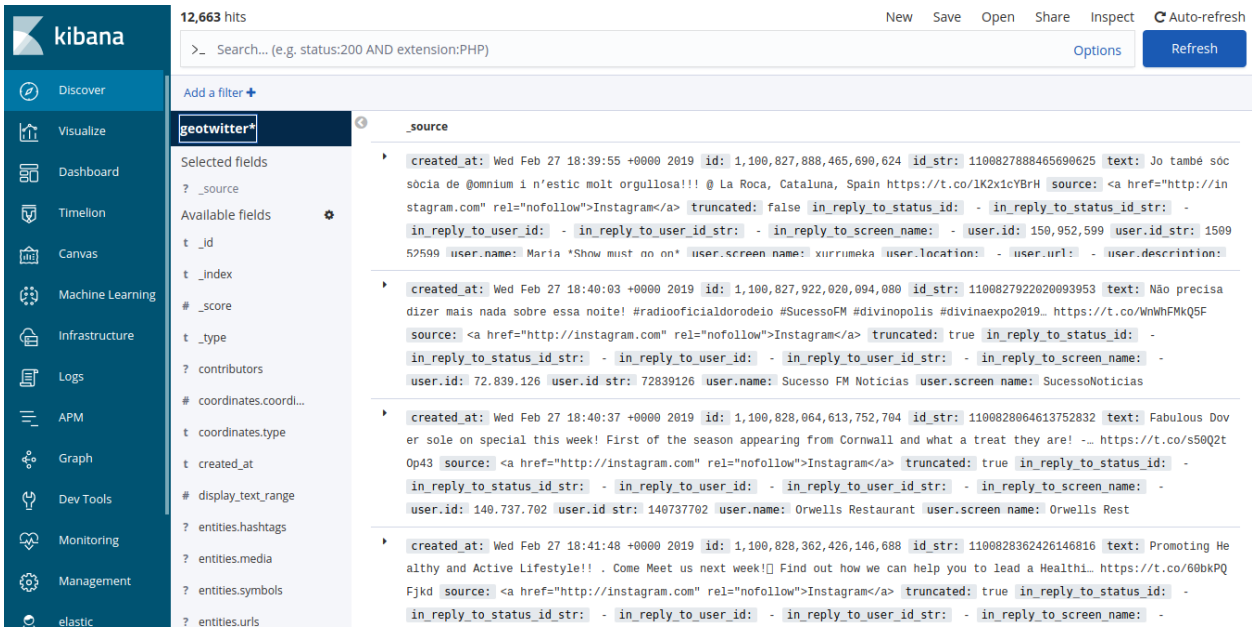
1. Collaboration Details:
    - Jesse: Indexed entirety of the tweet collection, deployed the Elastic Cloud to host index on a remote server, created a Python script for indexing/retrieval of JSON object tweets
2. System Overview
    - Architecture
        i. Python Script to connect to the server
        ii. Use ES object in Python to interface with the API
    - Index Structures
        i. Elasticsearch Index
    - Search Algorithm

i. Use of the Elasticsearch GET request to _search the index that our team built

3. How to deploy
    - [Elastic Cloud Login Page]
    - email: jreye039@ucr.edu
    - password: x7tK33uDxNcdBCB

    - Launching Elasticsearch Server
    - username: elastic
    - password: thvjOYqdxl4INNEDvxOFXC3R

    - IndexTweets.py has a class which can be used to either index a directory with tweets that have been previously collected or to index tweets at the time of them being crawled.
4. Screenshots



Pictured: Kibana interface displaying most recently indexed tweets

Part 3 - Extension

1. Collaboration Details:
    - Ryan: Initialized the MVC web application. Created distance filter, map/pin display. Built functionality for querying elasticsearch server given search text using NEST c# library.
    - Josh: Designed the web UI. Embedded tweets into results column on the left. Enabled tweet results to be sorted by rank or sorted by popularity.
2. Description:
    - We designed a web app that displays the results of our search engine. In addition to displaying the top 10 tweets for any given query, we display the geo-coordinates of those tweets using Google's API. Our web app features a query search bar, a distance filter for limiting results to within the user's radius, and a google maps display with pins for tweet locations. Clicking a map pin displays the associated user, tweet rank, and link to tweet.
    - This app was built using ASP.NET MVC. We used the ElasticSearch NEST library to return the json converted into c# objects given a user-inputted query. We also used System.Device GeoCoordinate class to calculate tweet locations and distances from the user's location. The user location is calculated using HTML5 geolocation.
3. Screenshots

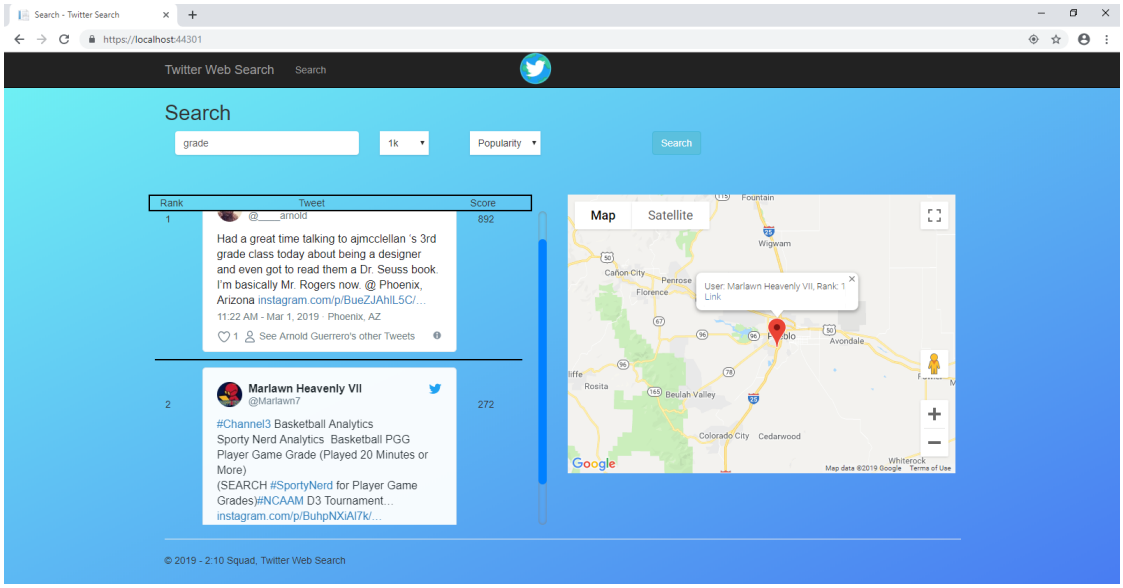


Figure 1: Results for query "grade" with distance maximum set to 1000 miles and tweets sorted by friend count.

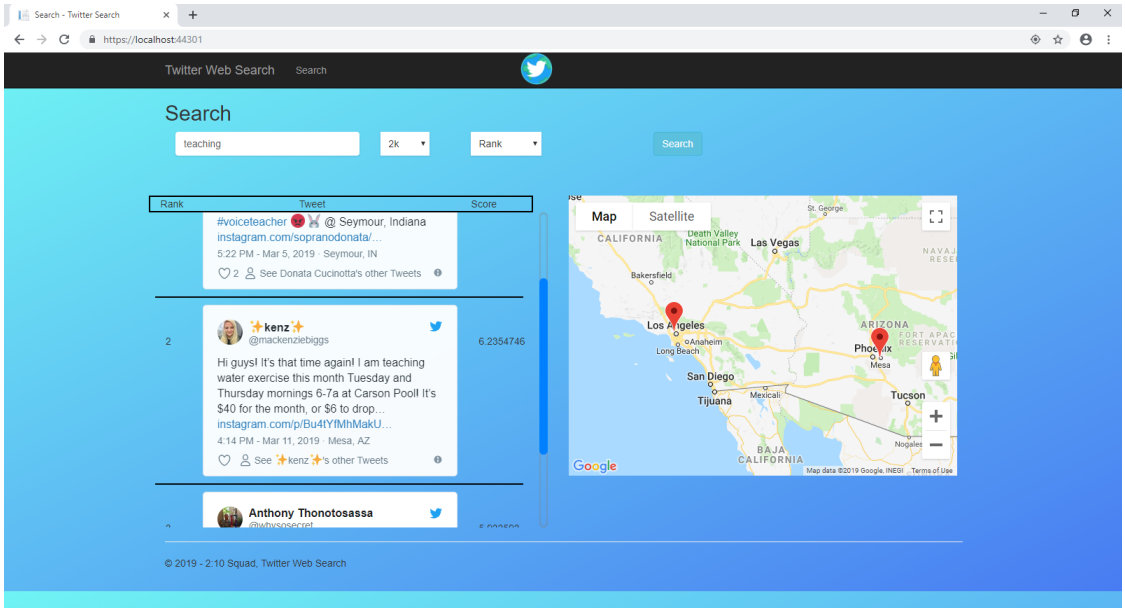


Figure 2: Results for query "teaching" with distance maximum set to 2000 miles and tweets sorted by rank scores.