Important note:

1. This assignment based on the fact the 1% of the data from streaming and the collection of geo_tagged data from streaming is within the same 1 hour period and only using streaming.filter(location = SG)
2. API Keys are not included in both reddit and twitter crawler, please check the README on where to insert API keys

Introduction

Software used:

Python 3.6, MongoDB.

Python packages used:

tweepy, pymongo, spacy, praw(reddit), Jupyter Notebook (For part 3), NLTK, Matplotlib, sklearn, numpy

Required python commands before running:

Python -m spacy download en

To start Jupyter: jupyter python

References:

Twitter streaming API: https://www.youtube.com/watch?v=wlnx-7cm4Gg

Determining K-means elbow using intertia: https://www.linkedin.com/pulse/finding-optimal-number-clusters-k-means-through-elbow-asanka-perera/

Twitter API: https://developer.twitter.com/content/developer-twitter/en.html

Various Ad-Hoc issues: reference stackoverflow for fixes.

Time and Duration of data collected:

Crawled twitter and reddit for both one hour.

Total twitter collected data: 12471 tweets

Total reddit data collected: 26081 posts

Data Crawl

Part A

For Streaming API, used the Tweepy Streaming class with a bounding box filter to ensure all streaming tweets are gotten from Singapore. To ensure that my Tweepy streaming function is listening all the time throughout the 1 hour, A thread is created just for listening to tweets for streaming and put any data in a queue to be processed in another thread. This is because there might be two streaming tweets that come in quick succession of one another and the second streaming tweet might be ignored if the Tweepy streaming function is processing the data for import into MongoDB.

Another thread is created for the processing of data and importing it into MongoDB. The processing of the data includes checking if the tweet is from Singapore as the bounding box include parts of Malaysia and Indonesia as well as checking if the tweet is in English. Not all tweets are geo-tagged, so tweets without any geo-tagged are assumed to be from Singapore. At this point Named Entity Recognition (NER) is conducted using spacy.

Part B

For REST API, used the Tweepy Cursoring . REST API is run in another thread.

The queries for the REST API come from three sources:

1. NER from streaming tweets
2. Trending topics gotten from Tweepy API
3. Users with the most retweets

The NER from streaming give us a hint on what event might be happening or person that is currently garnering a lot of attention in real-time. But only a small amount can be gotten from the streaming API, so we complement if with REST api to get as much tweet as whatever is happening right now according to the NER.

Trending topics are provided by the Tweepy API and can be used as a tool like NER.

User with the most retweets can be considered most influential and their tweets are queried to see any events that might be happening.

All queried keywords are put in a set to prevent them from being re-queried to reduce the number of redundant tweets.

As rest query data from the past, to ensure that data is as real-time as possible (within the 1 hour of running), Two algorithms are used.

1. Getting the first tweet ID that enters in streaming and using it in the *since_id* argument in the Tweepy Cursoring function
2. Getting the start time of the program and comparing it with every tweet to check if the tweets are after the start time of the program. If yes then proceed on with processing, if not then ignore.

After all the above, the tweets go through similar data processing such as ensuring it is English and checking if it is from Singapore, if no geo tags then assume it is from Singapore.

Part C

Getting as much geo-tagged data for Singapore is used in conjunction with *Part A*.

Part D

The issue with using REST to query real time data was that it always queried data from the past. So, getting any real-time data was an issue as it always returned close to no results. How that was circumvented was by using NER and starting the REST API 5 minutes after Streaming has started. This allowed enough NER to be produced by streaming and give some leeway for REST to query more data.

After querying the first round of NER, Trends and Users, I force the REST API to sleep for 5 minutes before querying from trends as trends are not in real-time. Again, this is to give some leeway for the REST to query more data. Finally querying users with the most retweet count from what we've gotten.

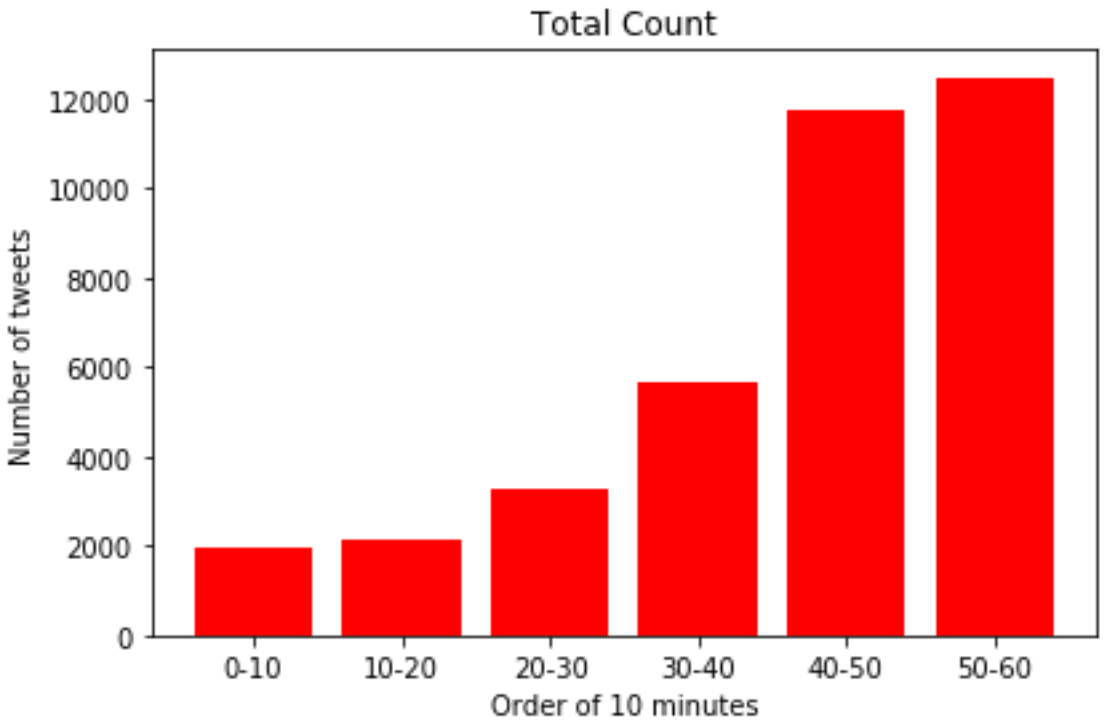
This process repeats until the end of the program. As stated above, all queried NER, trends and users are put in a set so future queries are compared against them to ensure there is no duplicated queries.

Data Analytics

Data analytics is done in a jupyter notebook, please checkout twitter_count.ipynb for more details.

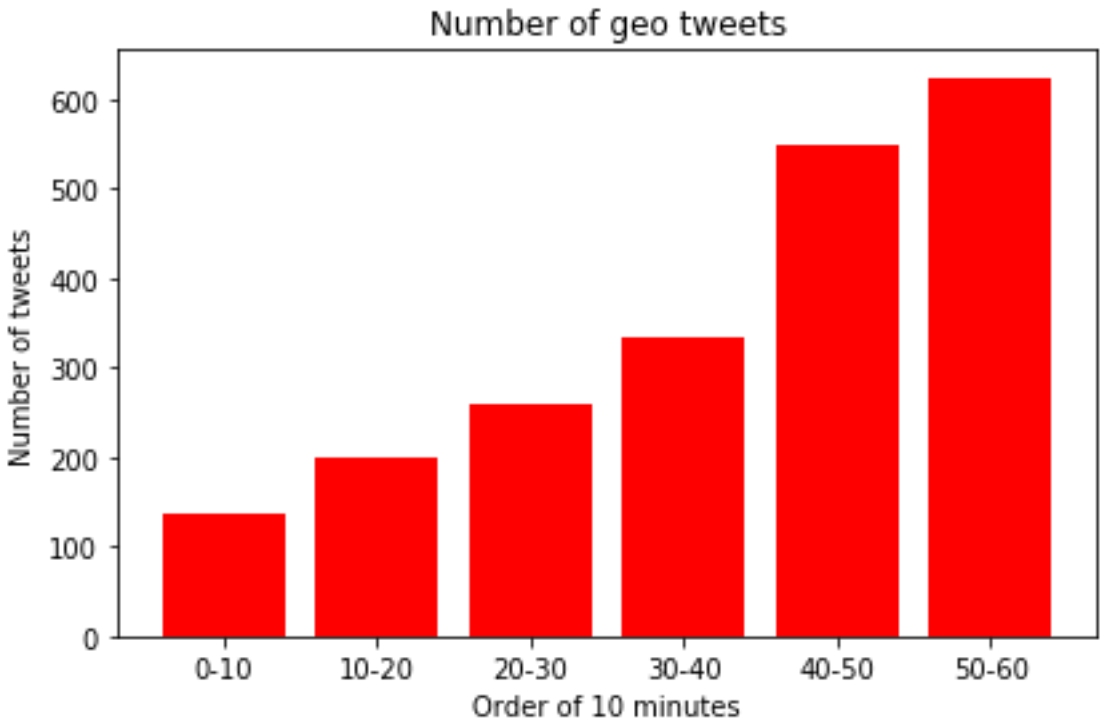
Total Count

0-10 mins: 1932, 10-20 mins: 2132, 20-30 mins: 3262, 30-40mins: 5633, 40-50 mins: 11734, 50-60 mins: 12478



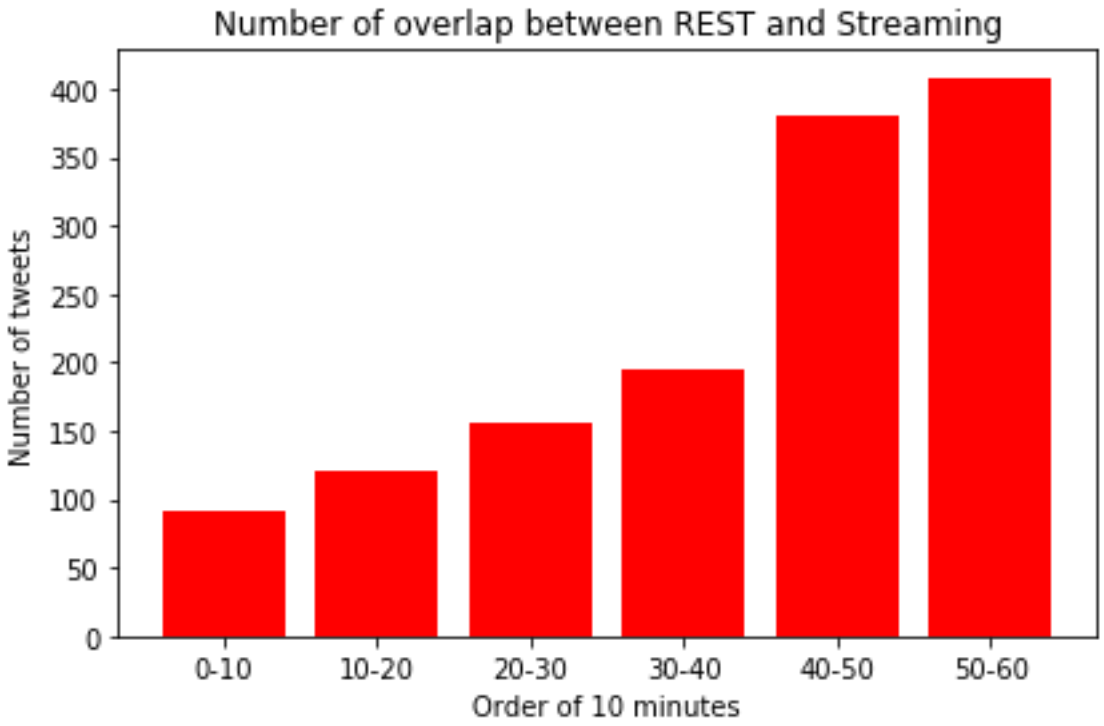
Geo-tagged, and SG geo-tagged data

0-10 mins: 137, 10-20 mins: 199, 20-30 mins: 259, 30-40mins: 333, 40-50 mins: 550, 50-60 mins: 624



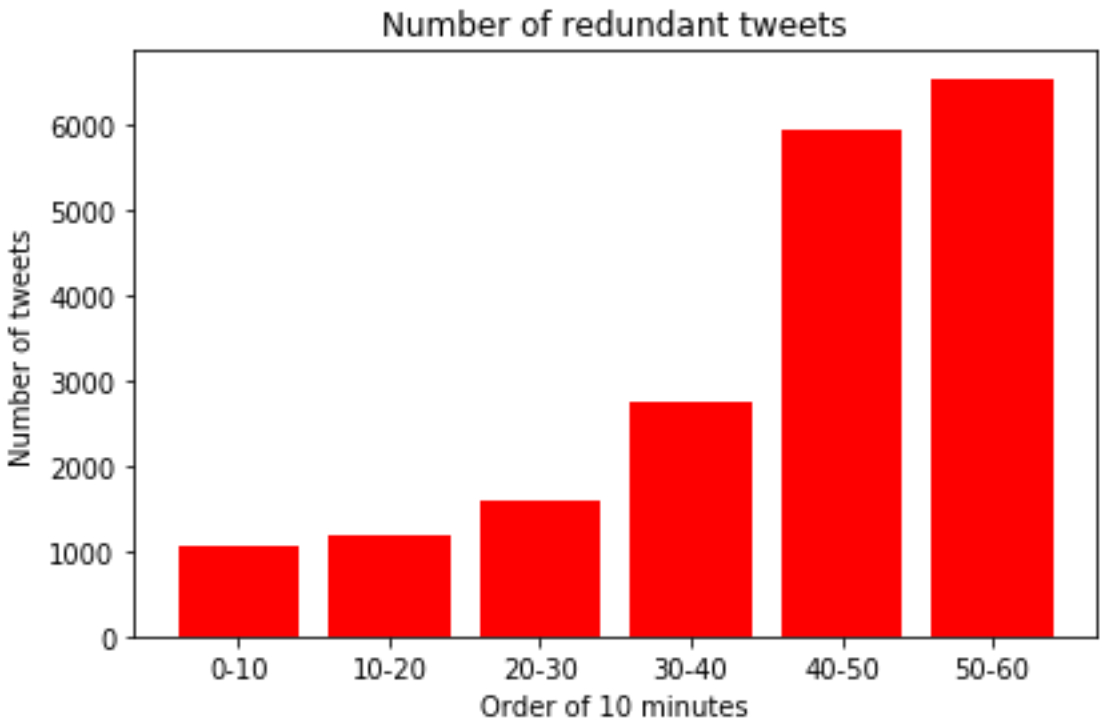
Number of overlaps between REST and Streaming

0-10 mins: 92, 10-20 mins: 120, 20-30 mins: 155, 30-40mins: 194, 40-50 mins: 380, 50-60 mins: 408



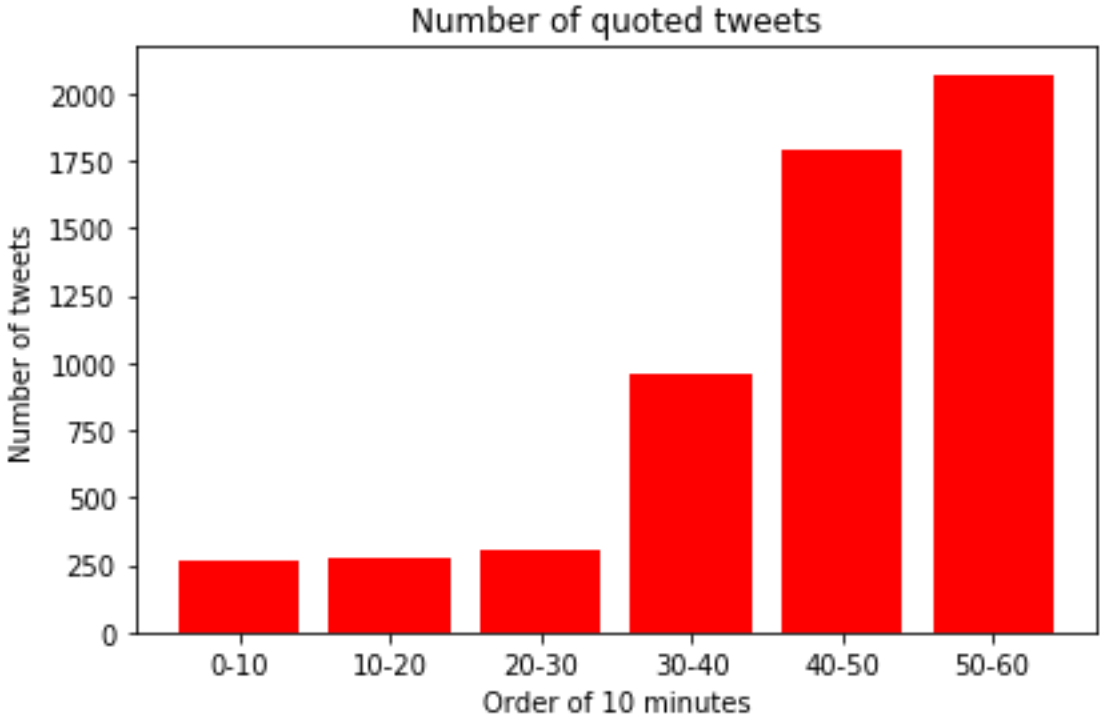
Number of redundant tweets

0-10 mins: 1064, 10-20 mins: 1174, 20-30 mins: 1602, 30-40mins: 2744, 40-50 mins: 5921, 50-60 mins: 6530



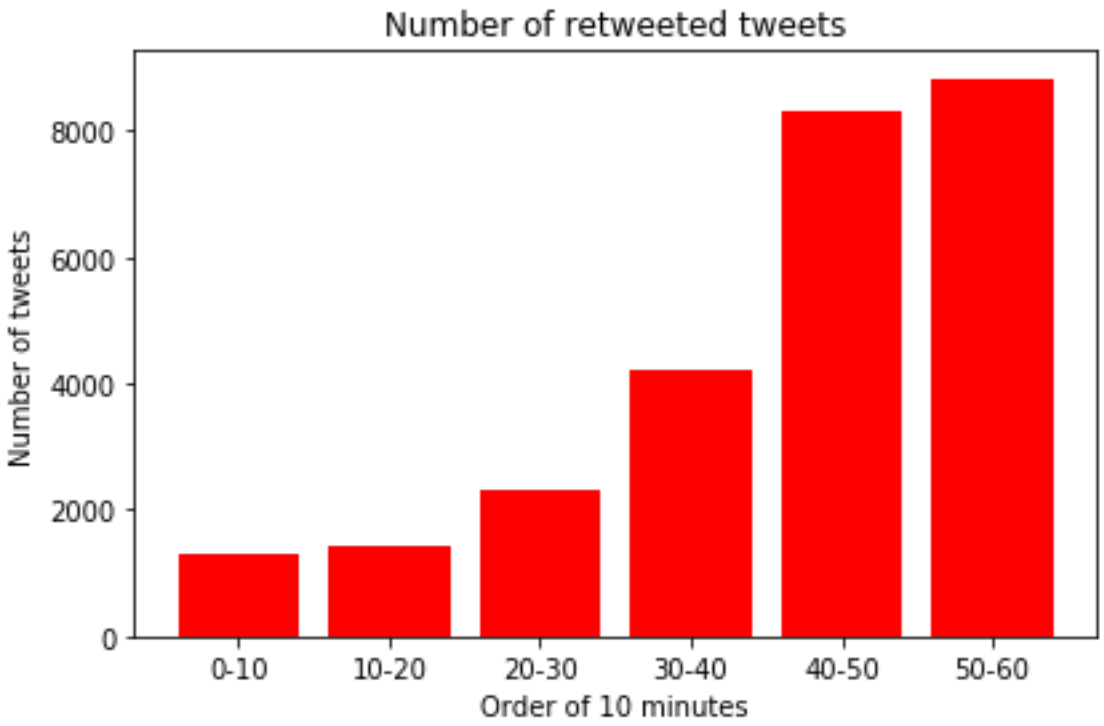
Number of quoted tweets

0-10 mins: 263, 10-20 mins: 276, 20-30 mins: 303, 30-40mins: 957, 40-50 mins: 1789, 50-60 mins: 2069



Number of retweeted tweets

0-10 mins: 1296, 10-20 mins: 1408, 20-30 mins: 2305, 30-40mins: 4210, 40-50 mins: 8321, 50-60 mins: 8828



Enhance the geo-tagged data

Grouping of tweets

For the grouping of tweets, There is basically 3 steps involved

1. Pre-processing of the data
2. Creating bag of words
3. Clustering using Kmeans

Pre-processing of the data

For the pre-processing of the data, the following are done:

1. Removing of links in tweets
2. Removing of non-ascii
3. Removing of digits
4. Removal of text that are less than 3 characters long
5. Removing of stop words
6. Removing of accented words
7. Lowercasing all text
8. Getting only the nouns of each tweet as we are clustering based on events
9. Lemmatizing all words extracted

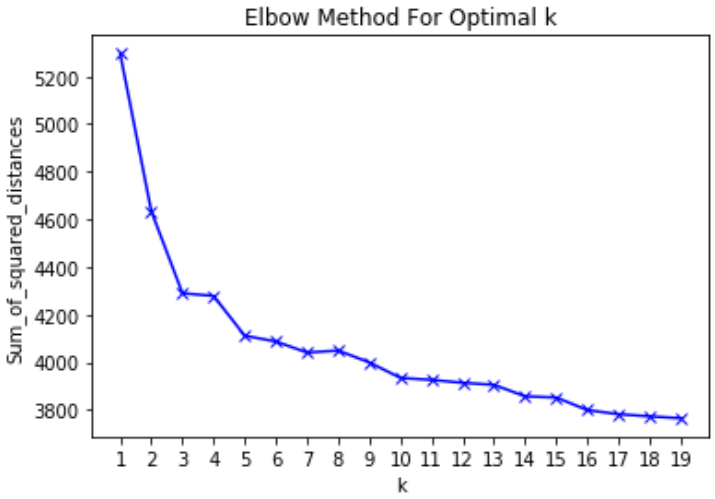
All the above steps are necessary as this is done to remove the words that are considered noise and might skew the clustering of data. We are only extracting the nouns from each tweet because we are clustering events which are basically nouns.

Creating bag of words

Creating a bag of words from every tweet using SKlearn countvectorizer

Clustering using kmeans

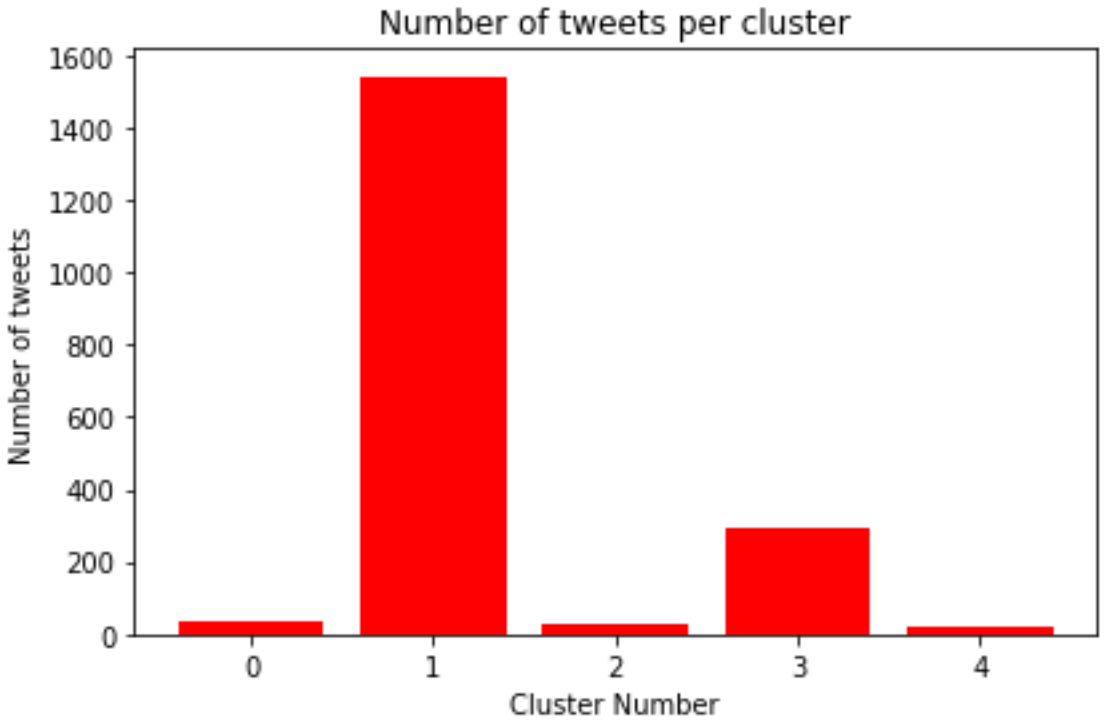
Clustering with kmeans is straightforward using sklearn, however there is a need to determine the elbow of the kmeans. To do this, we run the kmeans on the bag of words on 20 clusters and plot the graph.



The elbow of this graph is either 5 or 10 by visualization, so the kmeans cluster size is determined to be 5, form there the graphs for per cluster, geo_tags, profile location can be retrieved

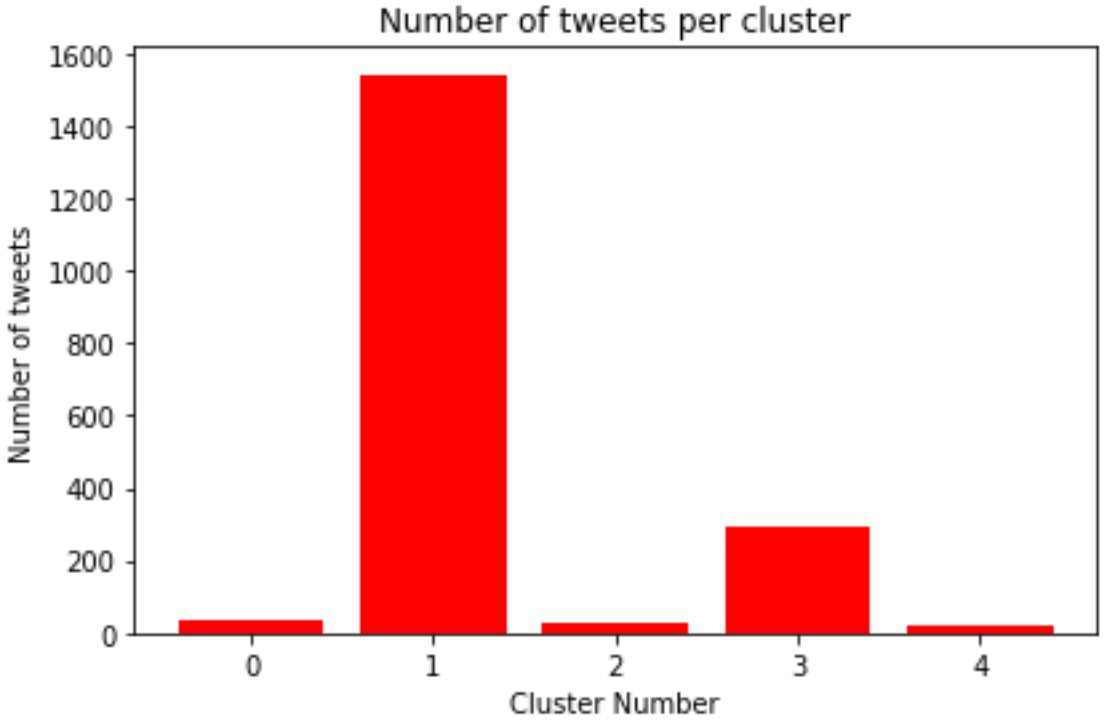
Tweets per cluster

Cluster 0: 32, Cluster 1: 1541, Cluster 2: 28, Cluster 3: 291, Cluster 4: 21



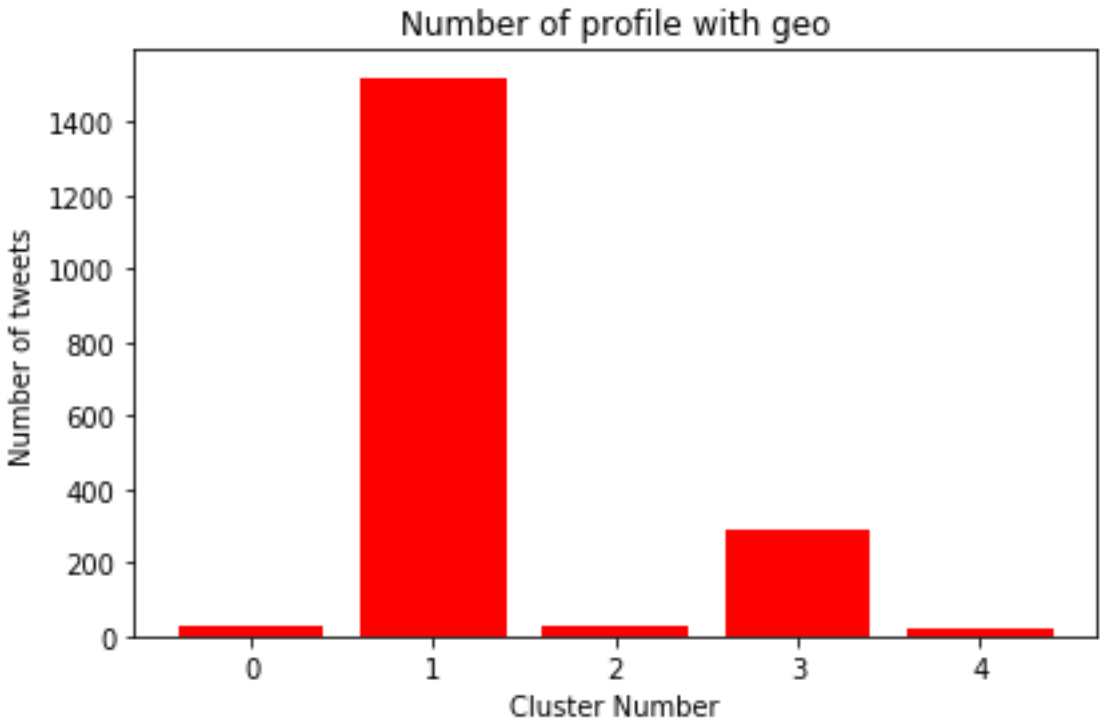
Number of geos tagged per cluster

Cluster 0: 8, Cluster 1: 161, Cluster 2: 4, Cluster 3: 44, Cluster 4: 0



Number of profiles with geo

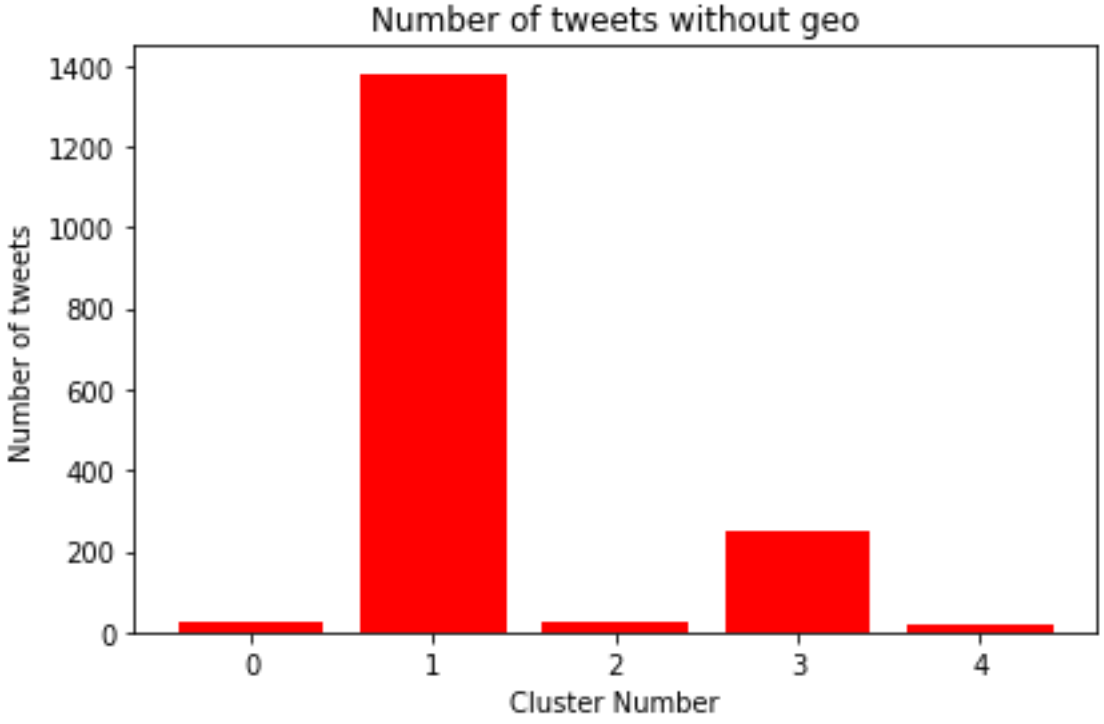
Cluster 0: 29, Cluster 1: 1519, Cluster 2: 26, Cluster 3: 287, Cluster 4: 21



Geo-Location

For the geo location, what I do is get the frequency of each locations in every cluster and create a probabilistic assignment model for assigning non-geo-tagged tweets.

Tweets without geo tags



Evaluation of the method

To evaluate the method, I split the tweets with geo tags 50/50. The first 50 is use it to create the probabilistic model and the second 50 I assign a location using the probabilistic model and check the hit rate. The hit rate of the location is as follows

Acuracy using 50% as training data and 50% as assignment for cluster 0: 50.0%

Acuracy using 50% as training data and 50% as assignment for cluster 1: 33.54037267080746%

Acuracy using 50% as training data and 50% as assignment for cluster 2: 50.0%

Acuracy using 50% as training data and 50% as assignment for cluster 3: 40.909090909090914%

No assignment possible due either 0 or 1 geo tweets for cluster 4

Crawling data from another social media: Reddit

Used PRAW package and crawled API.

Reddit API doesn't provide any top subreddits. To circumvent that, redditlist.com was used to get the subreddits with the top subscribers. Manually highlighted and converted into a json.

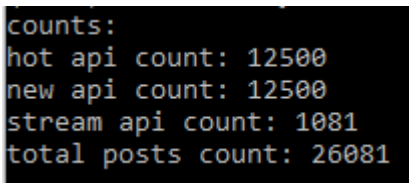
Similar to twitter, there is a Streaming API and REST API.

For the Streaming API, we listen for any new post in the top 125 subreddits and put it into a queue for insert into mongodb.

For the rest, we can query the top 125 subreddits according to "hot" and "new" post

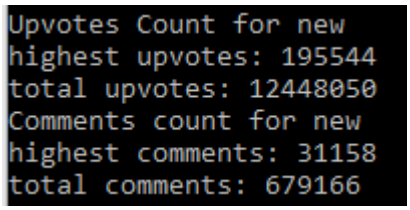
Data analysis

The analysis done on the reddit dataset is counting the number of post gotten from "hot" and "new" along with the total post in streaming



Count the number of upvotes and comments along with the highest upvoted and commented post for "new", "hot" and "streaming

New



Hot

```
Upvotes Count for hot
highest upvotes: 195554
total upvotes: 24069632
Comments count for hot
highest comments: 31158
total comments: 1161156
```

Streaming

```
Upvotes Count for stream
highest upvotes: 14
total upvotes: 1194
Comments count for stream
highest comments: 11
total comments: 121
```

Then count the number of post from streaming and new that made it into hot

```
number of streaming posts that got into hot during the 1 hour: 52
number of new posts that got into hot during the 1 hour: 9686
```