Web Science (H)
Course Work – Level 4

Event Detection

Name: Sea Hao Feng, Sam
GUID: 2355374S

1. Introduction

a) Describe the software developed with appropriate details; if you have used code from elsewhere please specify it

The software for Part 2 to 4 consists of 2 portions – Crawler Script (for data collection) and Analytics Script (for data analysis). All codes were developed from the documentation of their respective packages.

The following table shows the list of python packages utilised for this project:

Part 2	Part 3 & 4
Crawler (twitter_crawler.py)	Analytics (twitter_analytics.ipynb)
Packages: • threading • tweepy • json • pymongo • pandas • datetime • time • re	Packages: • pymongo • pandas • matplotlib • datetime • numpy • os

b) Specify the time and duration of data collected

The data was collected between 19th November 2018 2230HR to 19th November 2018 2330HR.

A total of 2 hours was spent running the twitter_crawler.py because the Streaming and REST API cannot be run in parallel. If ran concurrently, there will be no overlapped data from both Streaming API and REST API as the former takes live data and the latter retrieves historical data. (see Fig. 1 below).

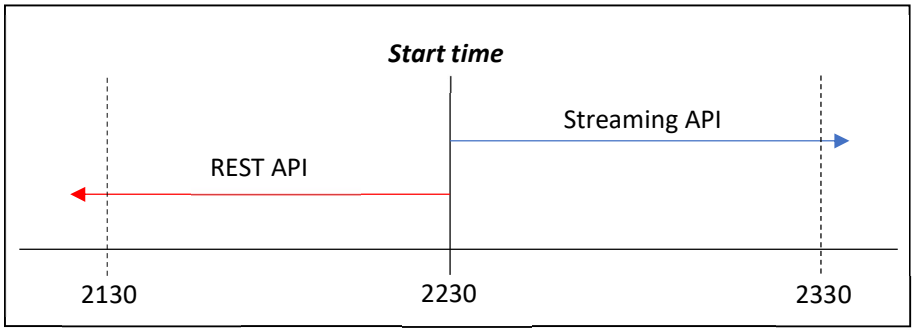


Fig. 1 – Concurrent Start time for Streaming and REST API

The Streaming API thread will fetch tweets created after 2230HR while the REST API thread will retrieve tweets created before 2230HR. Hence, to achieve an overlap of data we must run both Streaming and REST API in sequential order (refer to Fig. 2 below).

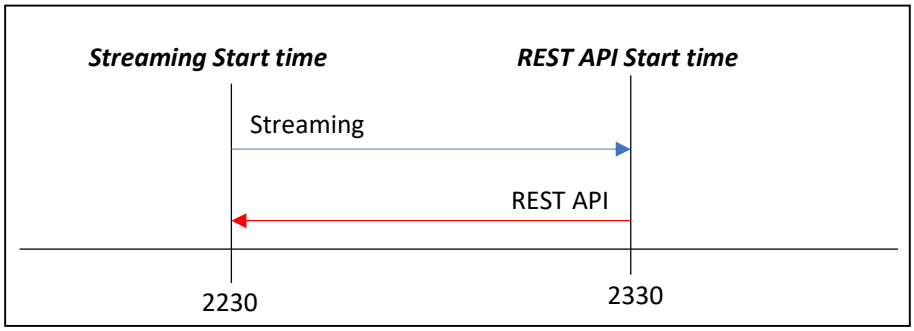


Fig. 2 – Sequential run of Streaming & REST API

Thus, the Streaming API thread will run for 1 hour followed by another 1 hour for the REST API thread.

2. Data Crawl

a) Use Streaming API for collecting 1% data

APIs used:

- tweepy.OAuthHandler
- auth.set_access_token
- tweepy.API
- tweepy.Stream
- tweepy.StreamListener
- streamer.filter

tweepy.OAuthHandler and auth.set_access_token creates the access for connecting to Twitter's API services
tweepy.API sets up the listener
tweepy.Stream and tweepy.StreamListener for accessing the Twitter's streaming API
streamer.filter for filtering the streaming based on language, specific keywords and/or geographical bounding box

b) Enhance the crawling using Streaming and REST API

APIs used:

- Streaming API
  - tweepy.OAuthHandler
  - auth.set_access_token
  - tweepy.API

- tweepy.Stream
- tweepy.StreamListener
- streamer.filter

tweepy.OAuthHandler for establishing user authentication
auth.set_access_token creates the access token for connecting to Twitter's API services
tweepy.API sets up the listener
tweepy.Stream and tweepy.StreamListener for accessing the Twitter's streaming API
streamer.filter for filtering the incoming streams based on language, specific keywords and/or geographical bounding box

- REST API
  - api.search
  - tweepy.cursor
  - cursor.items
  - tweet._json
  - api.trends

api.search returns a collection of relevant tweets that matched a specific query
tweepy.cursor for pagination loops
cursor.items to iterate all the tweets in the cursor object
tweet._json for decoding the JSON object from Twitter
api.trends to collect trending topics in a specific woeid (Where On Earth ID) region

c) Grab as much geo-tagged data for Singapore for the same period

In the Streaming API thread, the geographical bounding box of Singapore was added to the location parameter of the streamer.filter API

In the REST API thread, the geocode of Singapore was added to the location parameter of the api.search API

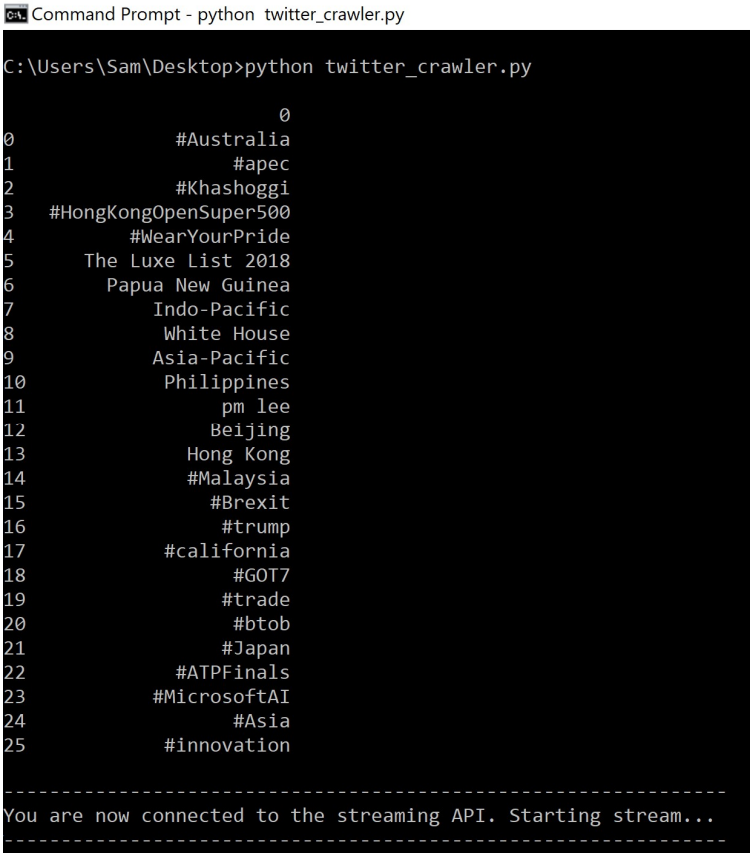
Even though the location parameters were in place, it does not accurately capture only Singapore tweets as the bounding box and geocode are just approximations of Singapore's international boundaries. Tweets from neighbouring countries (e.g. Malaysia and Indonesia) might be captured if their cities are within the coordinates. However, this method ensures that most of the tweets captured are from Singapore.

Also, the api.trends returns trending topics for a specific region and this will help to gather more Singapore tweets as the list of keywords returned from api.trends will be from the Singapore region.

d) Discuss your data access strategies and how did you address twitter data access restrictions

An additional parameter *wait_on_rate_limit = True* was passed to tweepy.API to manage the rate limit on Twitter API.

Another strategy is to include as much keywords via api.trends into the api.seach query parameter so that we can achieve a broader search spectrum. The following was the list of trending words when the twitter_crawler.py program was executed on 19th November 2018 2230HR:



Next, with a social media penetration rate of 25% for Twitter in Q3 2017 (https://www.statista.com/statistics/284466/singapore-social-network-penetration/) and 76% of 1000 survey participants residing in Singapore checking their mobile devices before bedtime (https://www.straitstimes.com/singapore/12hr-42min-connected-for-hours), this crawler was executed during night time from 2230HR to 2330HR (Streaming API) and 2330HR to 0030HR (REST API) to capitalise on Singapore's 'peak hour' internet traffic.

3. Basic Data Analytics

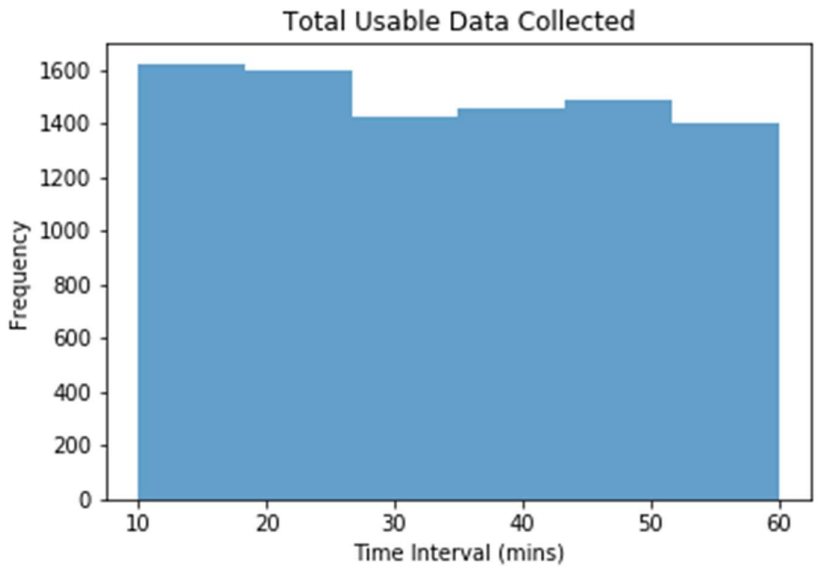
a) Count the amount of data collected (5 marks)

```
windowTime = (set1['created_at'] >= '2018-11-17 22:30:00') & (set1['created_at'] <= '2018-11-17 23:30:00')
data = (pd.DataFrame(set1.loc[windowTime])).reset_index(drop=True)

totalCollected = set1.API.value_counts()

print(f"Total no. of Tweets collected: \n{totalCollected}")
print(f"Total no. of USABLE Tweets collected: {len(data)}")
```

```
Total no. of Tweets collected:
REST         77780
STREAMING      236
Name: API, dtype: int64
Total no. of USABLE Tweets collected: 8993
```

*USABLE tweets = tweets' created_at field which falls between the start and end time of the crawler script.

i. Show a histogram of 10 minutes periods (x –axis duration of 10 minutes – y-axis count)

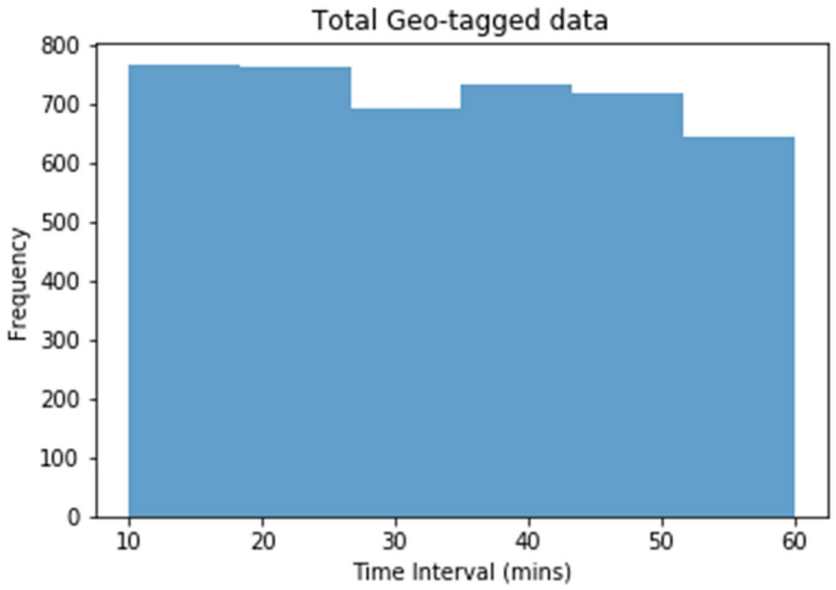


b) Count the amount of geo-tagged data from Glasgow / Singapore

```
trueCounter = 0
falseCounter = 0

for x in range(len(data)) :
    try:
        y = data['place'][x]['country_code']
    except TypeError:
        y = "NA"

    if data['user'][x]['geo_enabled'] and y == "SG":
        trueCounter += 1
    else :
        falseCounter += 1

print(f"Total no. of geo-tagged enabled users in Singapore: {trueCounter}")
```

```
Total no. of geo-tagged enabled users in Singapore: 441
```

i. Show a histogram of 10 minutes periods (x –axis duration of 10 minutes – y-axis count)

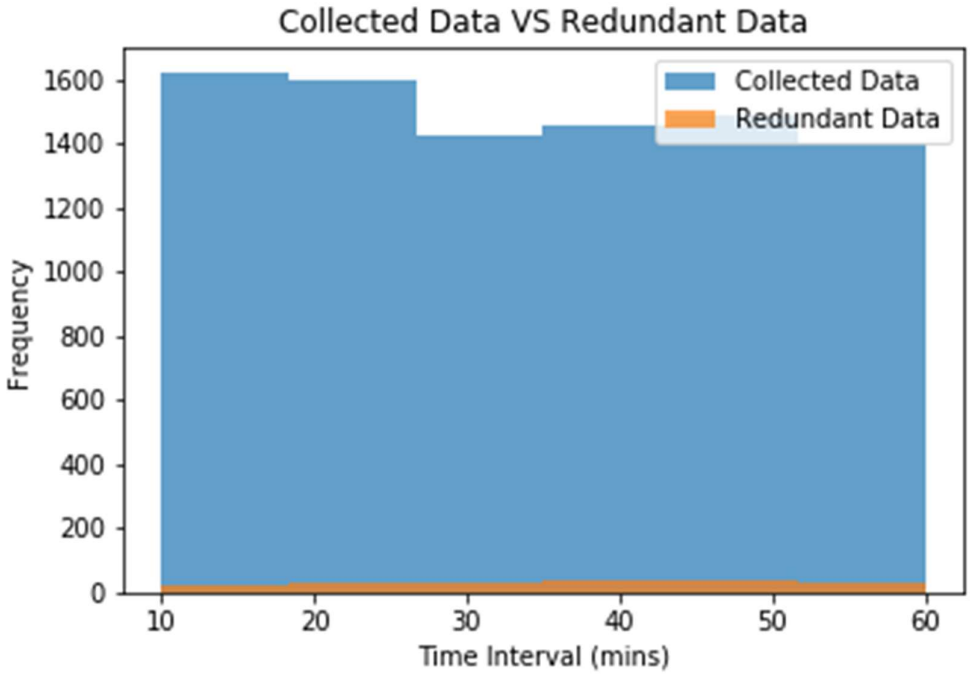


c) Count redundant data present in the collection (you may end up collecting the same tweets again through various APIs)

```
z = data['id_str'].duplicated()
dupCount = z[z == True].count()

print(f"Total no. of duplicated tweets: {dupCount}")
```

```
Total no. of duplicated tweets: 205
```

i. Show a histogram of 10 minutes periods (x –axis duration of 10 minutes – y-axis count) for both collected data and redundant data for same period

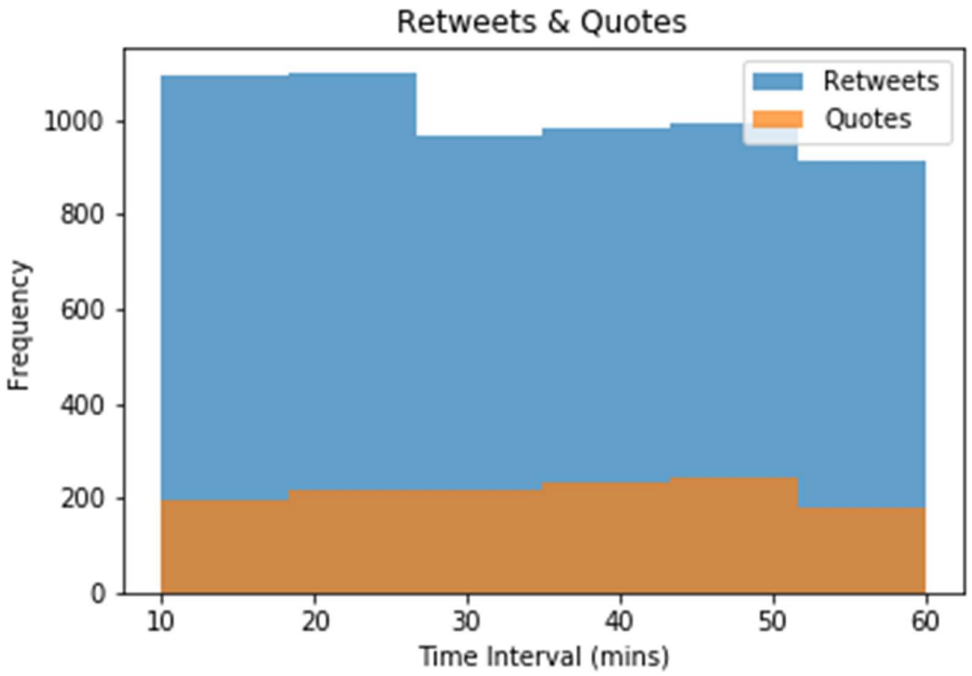


d) Count the re-tweets and quotes

```
w = data['retweeted_status'].notnull()
retweets = w[w == True].count()

q = data['is_quote_status']
quotes = q[q == True].count()

print(f"Total no. of retweets: {retweets}")
print(f"Total no. of quotes: {quotes}")
```

```
Total no. of retweets: 6047
Total no. of quotes: 1286
```

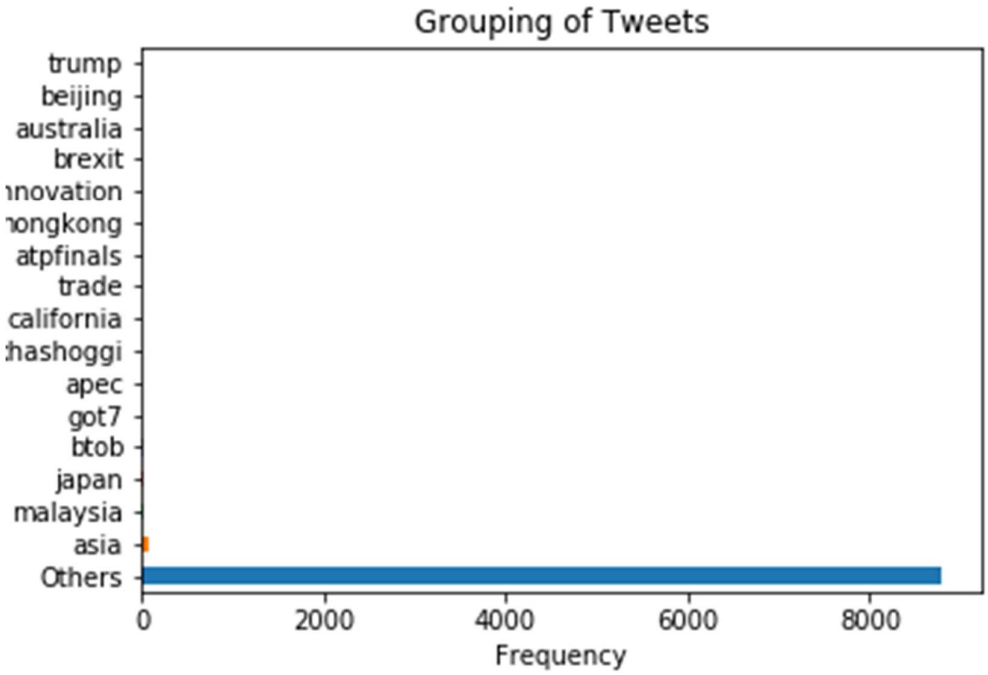
i. **Show a histogram of 10 minutes periods (x –axis duration of 10 minutes – y-axis count) for both collected data, re-tweets, quotes**



4. **Enhance the Geo-tagged data**

a) **Grouping of tweets**

```
test = data.Trend_group.value_counts().plot(kind='barh')
plt.title(label="Grouping of Tweets")
plt.xlabel("Frequency")
plt.ylabel("Topics")

# store to file
plt.savefig('Grouping of Tweets.png')
```




```
In [10]: data.Trend_group.value_counts()

Out[10]: Others        8784
         asia            84
         malaysia        35
         japan           29
         btob            22
         got7             9
         apec             5
         khashoggi        4
         california       4
         trade            4
         atpfinals        3
         hongkong         2
         innovation       2
         brexit           2
         australia        2
         beijing          1
         trump            1
         Name: Trend_group, dtype: int64
```

The reason for the high frequency of the classification 'Others' is due to the way the Twitter REST API Search Operator works. The screenshot below shows the parameters of the search operator:

```
Cursor(api.search, q=trendname, count=100, geocode=location, include_entities=True, lang='en', tweet_mode='extended')
```

Geocode will be the first filter for the search operator, followed by the q (query) parameter which is optional. Tweets must be within the geocode specified, else they will be ignored even if they match any of the keywords specified in the q parameter. The following is a list of possible combinations and their acceptance by the search operator:

Geocode	Q (Query)	Accept?
Yes	Yes	Yes
Yes	No	Yes
No	Yes	No
No	No	No

Thus, it is possible for the crawler script to retrieve a tweet that matches the geocode but not the list of keywords we throw into the q parameter.

b) Geo-location assignment

A possible method will be to find the majority geo-location of a specific topic and assign the location to any tweets that belongs to that topic (e.g. majority of the tweets in the topic group "asia" have a geo-location of Singapore. Thus, there is a high probability that any tweets that contains the word "asia" is from Singapore).

However, a keyword tokenization process is required for the category "Others" to further classify the sub-topics under "Others" before we begin to calculate the majority geo-location of each topic.

c) Conduct an evaluation of the method

A possible evaluation is to take 100% geo-enabled tweets and use the above geo-location assignment method to give them another geo-location based on the topic group they belong to.

We will then compare if the actual location and assigned location matches, taking the number of correct matches and divide it by the total amount of geo-enabled tweets which will give us the accuracy of the geo-location assignment method.

$$\frac{Correct\ Matches}{Total\ Amount\ of\ geo-enabled\ tweets} \times 100 = Accuracy\ (\%)$$

5. Crawling data from another social media

a) Develop a crawler for any of the following social media sites (facebook, Instagram, Google Plus, Tumblr, Flickr, any other);

Similarly, the software for Part 5 also consist of a Crawler Script and an Analytics Script. All codes were developed from the documentation of their respective packages.

Part 5	
Crawler (reddit_crawler.py)	Analytics (reddit_analytics.ipynb)
• praws • pandas • datetime • pymongo	• wordcloud • pymongo • pandas • re • os • PIL • numpy • matplotlib • seaborn

i. Describe access restrictions and the approach developed

When accessing sub-reddits and comments, there is a limit parameter which specifies the number of sub-reddits and comments we can iterate through.

To ignore the limit, we can simply set the parameter limit=None

```
for submission in subreddit.top('month', limit=None):
    submission.comments.replace_more(limit=None)
```

b) Conduct data analysis similar to Twitter data analysis

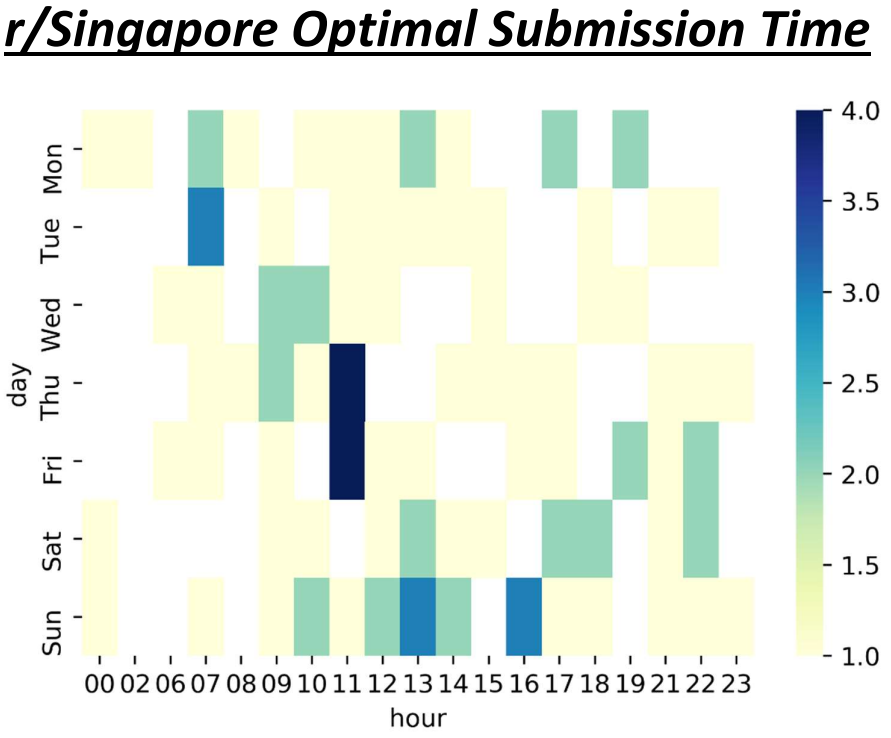
i. Word Cloud

A word cloud will be generated to get the top trending words for past month for the sub-reddit r/Singapore. Stop-words like conjunctions and common profanities were removed for better theme analysis

ii. Heatmap of submissions with a score of more than 200

Having a knowledge of "peak period" to post a submission will be effective for marketers to promote their content.

In this analysis, we will generate a heatmap to discover what is the optimal time interval of the week to create a submission on the sub-reddit r/Singapore if we want a higher probability of obtaining more than 200 points.



This analysis shows that the optimal period to create a submission on r/Singapore is on Thursday and Friday 11pm GMT +8 Singapore Time.