

Web Science Coursework Report (H)

Tawan Thampipattanakul – 2425142T

Both source code and data can obtained from this repository
github.com/maxoja/web-sci-coursework

Section 1 : Introduction

- (a) The software of this coursework was implemented into sequential Python scripts which should be run in a proper order separately except the crawler two scripts which should be run parallelly in hybrid crawler architecture manners. The software utilizes many third-party libraries such as NetworkX and Tweepy. Setup and execution instructions is provided in the repository's readme markdown as well as copied source code credits. Some low-level implementation about communication with Twitter's API and MongoDB was extracted into module files. You may not be interested in it and they are not mentioned in readme.
- (b) The Stream API and REST API crawler scripts were run parallelly for 3 hours on Saturday night of 29th February 2019

Section 2 : Data Crawl

- (a) Use Twitter Streaming API for collecting 1% data

Relevant source code is in repository <1_1_crawl_stream.py>

Basic filter Streaming API was used to crawl realtime data on specific topics ["corona virus", "Brexit", "sonic movie", "stock price"]. Initially I had chosen to get all realtime tweets without filtering any topics but then data grouping script in section 3 could not meaningfully cluster tweets with considerate level of square sum error. I believe that was because all tweets from everywhere were too sparse. That was why I later decided to scope down to only interesting topics. Those topics are what I thought topical at the time so I could gather more tweets compared to outdated topics. The retrieved data were stored in a Mongo DB collection using tweet IDs as unique indices so there would not be any duplicate documents in the document preserving storage usage and increasing analysis quality in later sections. There is no restriction for using Streaming API that the code needed to conform but it had limitation of 1% of data that is accessible.

- (b) Enhance the crawling using the architecture of Twitter Streaming & REST API

Relevant source code is in repository <1_2_crawl_rest_user.py>

The addition crawler loads tweet documents from Mongo DB collection and then analyses which users were most mentioned in the crawled tweets. Then, it uses the user_timeline REST API to gather more tweets from those users repetitively with newly crawled tweets taken into account. Since the has usage limit of 1500 requests per 15 minutes window, the script gives 0.67 seconds sleeping interval between each request guaranteeing that the limit of roughly 1350 request in a window. I intentionally left some quota buffer for unexpected reasons.

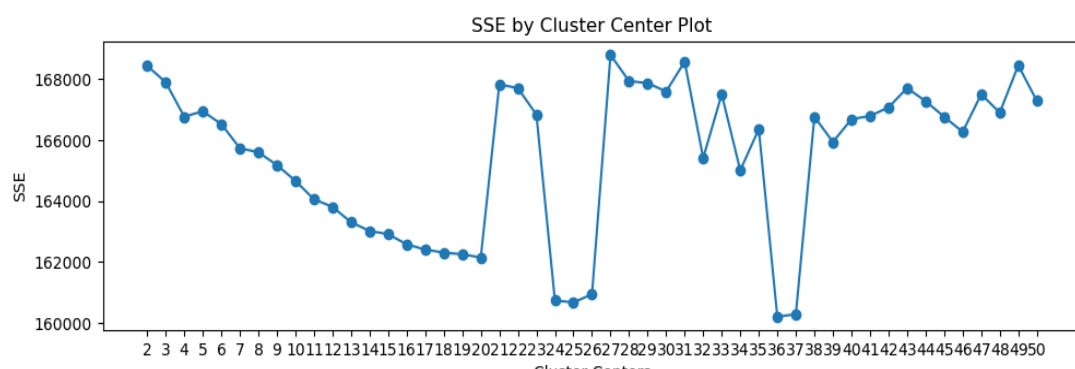
Section 3 : Grouping of tweets: Group the tweets based on content analysis, You can collect the data and then cluster them using any off-the-shelf software; or use any locality sensitive hashing software; or build a content index and group them

(a) Relevant source code is in repository <2_1_grouping_and_stats.py> <2_2_general_stats.py>

I have thought about several ways to cluster the crawled tweets including sentiment analysis, K-Mean of geolocation, K-Mean of vectorized tweets' content and N-bins hashing. First of all, randomly hashing would not create much semantic relation nor user relation among a group while sentiment analysis can only produce limited number of groups so I put away those two ideas. Clustering geolocations is interesting but very low proportion of tweets has geolocation enabled. Therefore, I finally decided to cluster texts of tweets using TFIDF vectorization and K-Mean clustering because it should yield decent number of clusters(groups) and it should be straight forward to look at texts of a same cluster and verify the result.

Please be noted that tweets of retweets was filtered out before the process started to reduce great amount of duplications which would alter the grouping result.

In brief details, TFIDF vectorization transforms texts into points in a vector space of |Vocab| dimensions. Then K-Mean algorithm tries to find points of similar semantic meaning. K-Mean starts with randomly generate K points in vector space and put vector points near to the same pivot as the same cluster. Average point of each cluster will be estimated and treated as a new pivot of the cluster. The process continues repetitively until the sum of distances from each vector point to its pivot (Square Sum Error or SSE) converged. At the final stage, each cluster represent a group with similar text meaning. The value K must be specified before the algorithm started and different K values will lead to different grouping result and SSE. To choose a proper K, elbow method was used to identify which amount of K best reduce SSE without splitting a meaningful group into multiple clusters. In the case of this coursework K=20 could be the ideal value but K=8 were chosen because it would be easier to analyse.



Overall statistics are as followed.

Num of Groups	Max Size	Min Size	Average Size
20	129056	971	21992

Numerical statistics for general group and clustered group (retweet excluded for clustered groups)

Group	Size	Unique Posters	Unique Hashtags	Unique Mentions
General	207269	112100	20648	75034
0	4125	3689	239	2169
1	129056	67594	11388	58891
2	2599	2341	839	1198
3	22627	17263	8921	9365
4	8532	7303	860	4441
5	3570	3241	443	2643
6	4455	4044	335	2521
7	971	908	162	155

- (b) After groups were identified, owning-users/hashtags/mentioned-users of tweets belonging to a group was extracted using the script by simply accessing [status->user->name], [status->entities->hashtags], and [status->entities->user_mentions] values and extract frequencies. Frequencies of them were analyzed and the result statistics of all tweets and each groups are as follow.

Group	Size	Important Posters	Important Hashtags	Important Mentions
General	207269	('Marvin R. Jeffcoat', 234) (',', 221) (Chris', 130) (quis custodiet', 117) (Julie.TrumpsGirl ususususus', 109)	('AEWRevolution', 894) (coronavirus', 249) (AEW', 219) (SCprimary2020', 200) (UFCNorfolk', 198)	('TySne B. Smith', 10141) (Donald J. Trump', 1830) (Cristiano', 1349) (perneeya', 1025) (CJ McCollum', 819)
0	4125	('My Style', 17) (Mawufesi', 11) (Andrew Wong', 10) (Daniel B. Orijuela', 10) (PeterSweden', 8)	(coronavirus', 30) (Corona', 28) (CoronaVirusUpdate', 19) (corona', 13) (COVID19', 11)	(Donald J. Trump', 280) (Nancy Pelosi', 52) (Dana Milbank', 31) (President Trump', 24) (YouTube', 20)
1	129056	(', 133) (quis custodiet', 115) (Chris', 83) (Ankhesenamun 🌐🇺🇸🇯🇵se', 74) (Portland Police log', 72)	('AEWRevolution', 660) (UFCNorfolk', 157) (AEW', 157) (SCprimary2020', 153) (SouthCarolinaPrimary', 138)	(Donald J. Trump', 894) (Joe Biden (Text Join to 30330)', 607) (Bernie Sanders', 386) (Elizabeth Warren', 258) (Tom Steyer', 141)
2	2599	('BoyBandStan', 20) (Dr.Bones', 18) (gallina fiedler', 7) (Terry Dinan', 7) (Mark Hansen', 7)	(ON', 22) (Sia', 20) (NMS', 18) (coronavirus', 12) (ctbb', 7)	(방탄소년단', 23) (Most Requested Live', 22) (Radio Carson PNW', 20) (Donald J. Trump', 14) (YouTube', 6)
3	22627	('brandon 🇺🇸🇯🇵', 61) (Tickeron', 46) (All Elite Wrestling', 26) (Mystery Solvent', 20) (Marvin R. Jeffcoat', 19)	('AEWRevolution', 129) (MyTwitterAnniversary', 87) (AEW', 35) (SCprimary2020', 32) (LeapDay', 29)	(YouTube', 144) (Bernie Sanders', 92) (Donald J. Trump', 72) (Joe Biden (Text Join to 30330)', 51) (Nashville SC's MLS Debut is TONIGHT', 33)
4	8532	('BrexitBot', 14) (', 11) (Taylor Rae', 9) (💎 Diamond Mami 💎', 8) (Gouthama Venkata Ramana Raju Chekuri', 8)	(Endomondo', 75) (endorphins', 69) (AEWRevolution', 40) (WoodenAward', 27) (SCprimary2020', 11)	(Joe Biden (Text Join to 30330)', 55) (Donald J. Trump', 53) (Wendy's", 28) (Bernie Sanders', 26) (Elizabeth Warren', 15)

5	3570	('surendra shetty', 14) (('Muhammad Farid', 11) (('Mike', 8) (('Traylonda', 8) (('Chris', 7)	('AEWRevolution', 25) (('UFCNorfolk', 8) (('SouthCarolinaPrimary', 4) (('photography', 3) (('TheInvisibleMan', 3)	('Joe Biden (Text Join to 30330)', 13) (('Donald J. Trump', 12) (('Nashville SC's MLS Debut is TONIGHT', 6) (('St. Louis BattleHawks', 5) (('Elizabeth Warren', 5)
6	4455	('Chelle Belle', 16) (('Pussy fairy', 5) (('Simon Clancy', 4) (('Simon Clancy', 4) (('Simon Clancy', 4)	('AEWRevolution', 19) (('loveisblindnetflix', 4) (('loveisblind', 4) (('SouthCarolinaPrimary', 3) (('AEW', 3)	('Donald J. Trump', 33) (('Joe Biden (Text Join to 30330)', 19) (('Elizabeth Warren', 12) (('Tom Steyer', 12) (('Mike Bloomberg', 9)
7	971	('River Levels UK', 27) (('Tim King', 6) (('Sassysherina', 4) (('Phil Strickland', 4) (('C', 3)	('March', 7) (('birthmonth', 3) (('BirthMonth', 2) (('minggupagi', 2) (('Matchmaking', 1)	('Jon Rothstein', 3) (('Vince Mathews', 2) (('Duke Men's Basketball', 2) (('LANY', 1) (('lauv', 1)

The categories are not clear. Many of them have the same popular hashtags and users. For examples, #AEWRevolution appears in 4 groups @Donald J. Trump appears in 5 groups. Fortunately, #coronavirus was strongly included in group 0 so it is quite clear that the content surrounds the topic. Each group has wide range of size from 971 to 129056 tweets.

It is noticeable that most topics used in crawling scope in section 1 are not reflected in the result of this grouping. I suspect that because I crawled the data on Saturday. Consequently, it could be hard to find real time tweets about stock market or Brexit. However, I cannot explain why there is no topics about Sonic movie.

Section 4 : Describe the metod for Capturing & Organising User and hashtag information. Objective is to build a user interactive graph

(a) Relevant source code is in repository <3to4_analysis.py>

For each group (including the general data group of all tweets) three interaction graphs (mentioning, retweeting, replying/quoting) were produced using Python's dictionary as the data structure behind on top of the following attributes.

[status->entities->user_mentions->id]

[status->retweeted_status->user->id]

[status->in_reply_to_user_id]

The initiative users' ids were used as keys and referencing to a list of targeted users' id with frequencies in form of tuples. The graphs are directional and can be bidirected. Please be noted that retweet graphs were constructed only for general tweets group because retweeting tweets were all filtered out during grouping step to improve clustering results.

(b) Relevant source code is in repository <3to4_analysis.py>

Hashtag information can be obtained by accessing the [status->entities->hashtags->text] attribute from a tweet. While looping over all tweet statuses in a group, the script uses Python's dictionary data structure to contain connections between hashtags that occurred in the same tweet. The dictionary serves the same functionality as adjacency list where we can use a key/index in form of a hashtag to reference a list of hashtags in the same tweet. Such data structure provides better search complexity but less memory efficiency. The graph is

undirected because the occurrence of 2 hashtags appearing together in a single status applies for both hashtags.

I'm not sure what kind of tabular data should be represented and contrasted here since there is not much to say without network analysis in the next section. The only major difference between different type of interaction graph that I know is the graph of hashtags occurring together can be undirected while the rest must be directional.

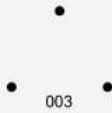

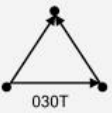
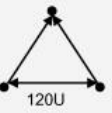

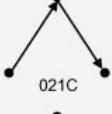

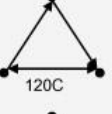
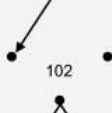
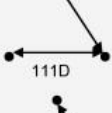
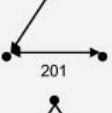
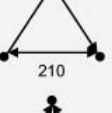
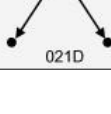
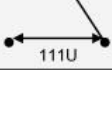
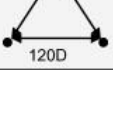
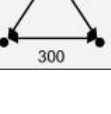
You can check out the interaction graphs visualizations by running the code. I did not include them here because I think they are overwork.

Section 5 : Network Analysis Information

(a) Relevant source code is in repository <3to4_analysis.py>

The script uses NetworkX's utility algorithm to count different types of triads. In prior to do so, NetworkX's Graph must be created based on interaction dictionary constructed in section 4. The script simply loops over keys. A key user M and interacted users [K..] will be processed by adding node M and edges M,K to the graph. Finally NetworkX's triadic_census(graph) is called to count how many triads of each type are there in the parsed graph. The algorithm took unknowingly large amount of time for large graph. Therefore, the script down-samples clustered and general groups to only 3000 tweets in the process of section 4 so that waiting time become feasible.

On the other hand, a new ties counting function was implemented on top of NetworkX's Graph object. Let's say A is the number of all directed edges and B is the number of all undirected edges of the same graph (2 edges of a same pair will be merged). The function then returns A and A-B as the numbers of first order and second order ties respectively.

Triad	Probability	Triad	Probability	Triad	Probability	Triad	Probability
	$N^{(3)}$		$\frac{3}{4} NA^{(2)}$		$\frac{3}{4} A^{(3)}$		$\frac{3}{4} MA^{(2)}$
	$3AN^{(2)}$		$\frac{3}{2} NA^{(2)}$		$\frac{1}{4} A^{(3)}$		$\frac{3}{2} MA^{(2)}$
	$3MN^{(2)}$		$3MAN$		$3NM^{(2)}$		$3AM^{(2)}$
	$\frac{1}{4} NA^{(2)}$		$3MAN$		$\frac{1}{4} MA^{(2)}$		$M^{(3)}$

Based on the network analysis results shown below, it is clear that mention interaction graphs always have significantly more triads in general compared to other kinds of interaction followed by reply and retweet graphs respectively. This reflects that more users are keen to mention others than reply nor retweet. It is significantly harder to find stronger

connected triads as 120D 120U 120C 210 and more triad types are never found in the sample. Because the difference in overall number of triads are very significant, it is not simple to compare proportion of each triad types and interpret further. Another highly potential cause of the problem could be that each group were down-sampled.

About ties, most reply graphs seem to have higher proportion of strong ties than mention graphs. I think because it is more often to see someone replied back than to see someone mention or retweet back to an initiative user.

Group General Tweets of size 207269 -> 3000

Mention Graph Triads

{'003': 7366396670, '012': 8451206, '102': 0, '021D': 1820, '021U': 12093, '021C': 145, '111D': 0, '111U': 0, '030T': 2, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 2419, strong 3)

Reply Graph Triads

{'003': 1023113418, '012': 1653283, '102': 0, '021D': 9, '021U': 43, '021C': 3, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 1001, strong 2)

Retweet Graph Triads

{'003': 35932451, '012': 236898, '102': 0, '021D': 5, '021U': 10822, '021C': 24, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Retweet Graph Ties

(weak 471, strong 0)

Group 0 of size 4125 -> 3000

Mention Graph Triads

{'003': 4430436967, '012': 6828767, '102': 8950, '021D': 1885, '021U': 16451, '021C': 7, '111D': 0, '111U': 2, '030T': 13, '030C': 0, '201': 0, '120D': 0, '120U': 3, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(2329, 3)

Reply Graph Triads

{'003': 2086209275, '012': 3052161, '102': 11609, '021D': 93, '021U': 5176, '021C': 9, '111D': 0, '111U': 1, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 1414, strong 5)

Group 1 of size 129056 -> 3000

Mention Graph Triads

{'003': 7137917058, '012': 7909659, '102': 3496, '021D': 1874, '021U': 625, '021C': 29, '111D': 0, '111U': 1, '030T': 6, '030C': 0, '201': 0, '120D': 1, '120U': 1, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 2325, strong 1)

Reply Graph Triads

{'003': 2566852285, '012': 3089894, '102': 0, '021D': 45, '021U': 38, '021C': 18, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 1324, strong 2)

Group 2 of size 2599

Mention Graph Triads

{'003': 1242919076, '012': 2416746, '102': 5845, '021D': 1014, '021U': 99, '021C': 21, '111D': 0, '111U': 2, '030T': 4, '030C': 0, '201': 0, '120D': 0, '120U': 12, '120C': 0, '210': 0, '300': 1}

Mention Graph Ties

(1263, 3)

Reply Graph Triads

{'003': 157840825, '012': 469322, '102': 0, '021D': 14, '021U': 18, '021C': 5, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 532, strong 0)

Group 3 of size 22627 -> 3000

Mention Graph Triads

{'003': 2578334524, '012': 4013769, '102': 2477, '021D': 1437, '021U': 173, '021C': 48, '111D': 2, '111U': 12, '030T': 1, '030C': 0, '201': 0, '120D': 0, '120U': 1, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 1558, strong 1)

Reply Graph Triads

{'003': 129803774, '012': 401012, '102': 0, '021D': 29, '021U': 22, '021C': 3, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 490, strong 0)

Group 4 of size 8532 -> 3000

Mention Graph Triads

{'003': 3533542051, '012': 4937247, '102': 5524, '021D': 1348, '021U': 416, '021C': 40, '111D': 0, '111U': 5, '030T': 2, '030C': 0, '201': 0, '120D': 0, '120U': 7, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 1805, strong 4)

Reply Graph Triads

{'003': 1174161321, '012': 1796034, '102': 7665, '021D': 49, '021U': 42, '021C': 5, '111D': 1, '111U': 2, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 1031, strong 1)

Group 5 of size 3570 -> 3000

Mention Graph Triads

{'003': 7702976532, '012': 8399970, '102': 0, '021D': 2516, '021U': 193, '021C': 22, '111D': 0, '111U': 0, '030T': 2, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 2353, strong 0)

Reply Graph Triads

{'003': 2496226834, '012': 3127657, '102': 0, '021D': 186, '021U': 22, '021C': 6, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 1286, strong 0)

Group 6 of size 4455 -> 3000

Mention Graph Triads

{'003': 3881386954, '012': 5290960, '102': 5712, '021D': 1578, '021U': 436, '021C': 12, '111D': 0, '111U': 0, '030T': 4, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 1871, strong 1)

Reply Graph Triads

{'003': 1341344122, '012': 2011772, '102': 2004, '021D': 78, '021U': 40, '021C': 4, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 1075, strong 1)

Group 7 of size 971

Mention Graph Triads

{'003': 2790018, '012': 38908, '102': 0, '021D': 129, '021U': 1, '021C': 0, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Mention Graph Ties

(weak 156, strong 0)

Reply Graph Triads

{'003': 519470, '012': 9924, '102': 0, '021D': 2, '021U': 0, '021C': 0, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 0, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Reply Graph Ties

(weak 82, strong 0)
