Web Science Assessed Exercise – Level H

Erin Christie – 2317517c

Source code - https://github.com/ErinChristie/2317517cWebScienceAE

Data - https://github.com/ErinChristie/2317517cWebScienceAE/blob/master/WSAE.csv

Section 1 – Introduction

To gather the Twitter data used in this project I first had to register as a Twitter Developer. Next, I created an App so I could generate an API key, an API secret key, an access token and an access token secret. These keys and access tokens provided authentication and access to the Twitter Account I created so I could use the App.

I created a MongoDB to store the data that would be collected from the Streaming API. A program was then written to use the Python library Tweepy and the Twitter Streaming API to collect the Twitter data. The code I used to do this was influenced by the code written by Sam Delgado (https://github.com/SamDelgado/twitter-to-mongo/blob/master/twitter-to-mongo.py). I chose to filter the streamed data with the keywords "megxit", "meghan" and "harry". The program was left to run for an hour which resulted in over 4000 Tweets being collected. I collected the data early afternoon on the 2nd March 2020.

Section 2 – Data crawl

The APIs used were the Twitter Steaming API and the REST API to collect and filter the Tweets respectively. The code first established a connection to localhost so that the MongoDB database and collections could be referenced. A list of the keywords was made, and the language was set to English to ensure only English Tweets were collected. To stream the Tweets, I used the StreamListener from Tweepy. By using this method, I had to ensure that error handling was done correctly which is good software development practice. Also, data from the stream will be outputted immediately as it creates a persistent connection. This is done with little overhead costs. The StreamListener method on_data_ extracts the kind of Twitter data sent. The stream was filtered using the REST API and the keywords (see section 1). The Tweets were parsed using json.loads() which adds the Tweet to a Python dictionary. The Tweet's id, username of the poster, the number of followers the user has, the contents of the Tweet (the text), any hashtags used, the time it was created and the language used were stored in the database, provided an exception was not thrown. In the case an error occurred, the error status was printed to the command line.

The REST API uses the Tweepy Cursor to find the followers of each user identified in the data stream. The follower list for a specific user (@DailyMirror) was also collected this way.

Section 3 – Grouping of Tweets

In order to group the Tweets, I wanted to cluster the usernames, hashtags and text into groups. For each of the categories, 8 clusters were made. The code I wrote to gather the clusters was influenced by the code found at https://stackoverflow.com/questions/27889873/clustering-text-documents-using-scikit-learn-kmeans-in-python?fbclid=IwAR13agTGUdH3e7Xdpt2x6ee6R8vrzjWCuguWgCgTklOcmcYBwVdO6ak8c3k.

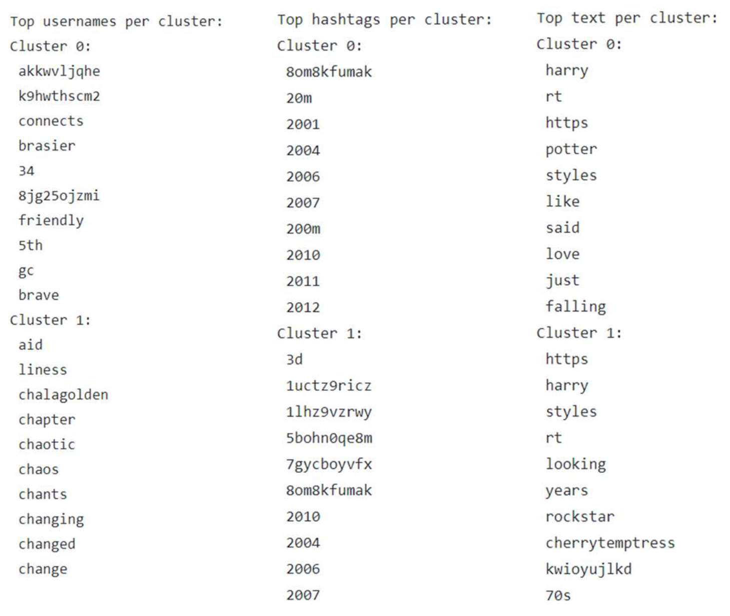
First, the data stored in the database had to be put into a CSV file so the data could be read in by the clustering program that was written. The data was vectorised using TfidfVectorizer so that each of the usernames, hashtags and texts could be fitted to the dataset and then transformed using fit_transform().

I chose to cluster my data using KMeans clustering for a couple of reasons. KMeans clustering is easy to implement and as I had a large dataset, it tends to be computationally faster compared to other clustering methods such as hierarchical clustering. I chose to group my data into 8 clusters to give a good representation of the data collected. I did this by setting the k value to 8. A for loop was then used to find the top 10 usernames for each of the 8 clusters. This step was repeated for the hashtag and text data. As the data was processed, the data was written to a text file (see Clusters.txt on GitHub).

To identify and extract the usernames, hashtags and text I used the same table headings as in the database and csv file which are then used later in the code to get the clusters. The strings need to be vectorised first as the clustering only works on numerical values.

The groups are formed by clustering the data into the specified number of groups that show a natural pattern. Therefore, some of the groups include very similar data whereas others may not make much sense at all.

Examples of the clusters formed:



```
Top usernames per cluster:
Cluster 0:
 akkwvljqhe
 k9hwthscm2
 connects
 brasier
 34
 8jg25ojzmi
 friendly
 5th
 gc
 brave
Cluster 1:
 aid
 liness
 chalagolden
 chapter
 chaotic
 chaos
 chants
 changing
 changed
 change
```

```
Top hashtags per cluster:
Cluster 0:
 8om8kfumak
 20m
 2001
 2004
 2006
 2007
 200m
 2010
 2011
 2012
Cluster 1:
 3d
 1uctz9ricz
 1lhz9vzrwy
 5bohn0qe8m
 7gycboyvfx
 8om8kfumak
 2010
 2004
 2006
 2007
```

```
Top text per cluster:
Cluster 0:
 harry
 rt
 https
 potter
 styles
 like
 said
 love
 just
 falling
Cluster 1:
 https
 harry
 styles
 rt
 looking
 years
 rockstar
 cherrytemptress
 kwioyujlkd
 70s
```

<u>Section 4 - Capturing & Organising User and hashtag information through a user interaction graph</u>

In order to develop a method to capture user mention information, the csv file containing all the Tweets' data had to be filtered. A user mention involves the @ symbol being used followed by a username and so the data was filtered by this condition. No Retweets were included in this. I used software provided by Netlytic to generate the user interaction graphs. The user interaction graph (figure 1) show that the power users of my data set are

@piersmorgan with 11 interactions (figure 2), @gmb with 10 (figure 3) and @susannareid100 with 9 (figure 4). These are the users that occur together. Piers Morgan and Susanna Reid both currently present Good Morning Britain (GMB) so it would make sense that these users occurred together in the user interaction graph.

To show the Retweet network of the data, only Tweets that had “RT” at the start of the text field were included. A user interaction graph was then created with this data (figure 5). As expected, the number of Retweet interactions was much higher than that of user mentions. This could be because users are more likely to Retweet something that they agree with so they can share it with all their followers. It takes less effort for a user to Retweet a certain Tweet than it is for them to construct their own Tweet that includes a user mention. Another reason for this could be that only a sample was taken meaning the resulting data collected may not be completely representative of the entire population of Tweets containing the keywords as mentioned in section 1. The graphs (figures 5 and 6) indicate that @hsdaily had the most Retweets (70) and @theharrynews had the second most with 49 Retweets.

A user interaction graph was also created for the general Tweet data that was originally collected (figure 7). The same process was followed as above but with the original csv file. The users with the most interactions were @hsdaily (figure 8), @theharrynews (figure 9) and @youtube (figure 10). These results are in line with the results from the Retweet network graphs which was anticipated. If a user has a high number of Retweets, they will also have a high overall interaction with users.

The graph in figure 11 shows hashtag co-occurring information. This was produced using the software Cortext and filtering the original csv file to only include the hashtags used by users. This information arranged itself into 6 distinct categories. The bigger the circles, the more Tweets containing those hashtags there are. The two largest groups centre around #HarryandMeghan (figure 12) and #BestLyrics (figure 13). The former was likely since two of the keywords used to filter the Tweets were Harry and Meghan. The latter is less obvious. #BestLyrics refers to Harry Styles rather than the Harry of Harry and Meghan. The hashtags #LoveYouToLoveMe (one of the songs Harry Styles’ songs) and #iHeartAwards were also included in this group. #iHeartAwards may have be included as Harry Styles has been nominated for an award in the upcoming event taking place on 30th March 2020. Other hashtags included in the #HarryandMeghan group were #RoyalFamily, #LetThemEatCake, #Megxit and #MarieAntoinette which are in light of recent events to do with Meghan and Harry leaving the Royal Family. There is another group with the hashtags #Harry and #MeghanMarkle (figure 14). On further inspection of the original Twitter data, the other hashtag groups relate to books that have just been announced by @JudyandKeith (figure 15) and @ENicolson1 (figure 16) each with characters called Harry, or Harry Potter merchandise (figure 17).

Appendix

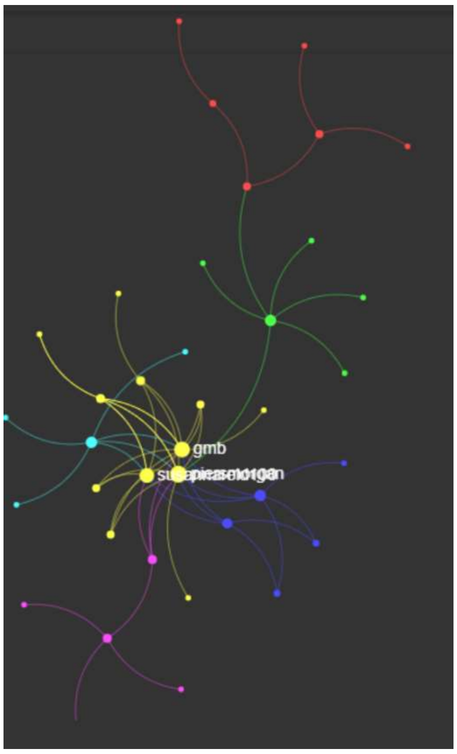


Figure 1 – User interaction graph for user mentions.

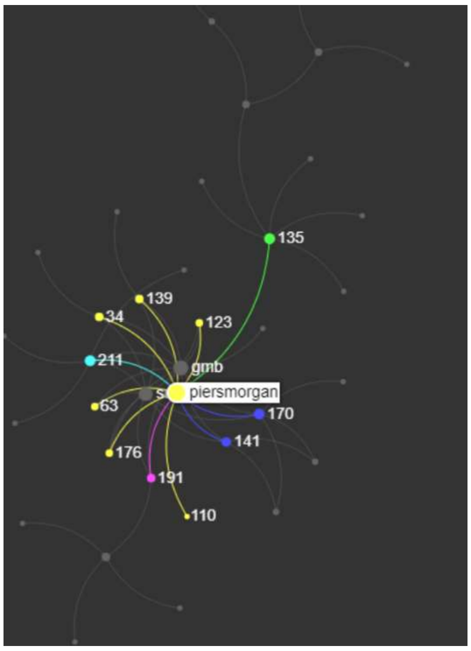


Figure 2 – User interaction graph for user mentions highlighting network with @piersmorgan.

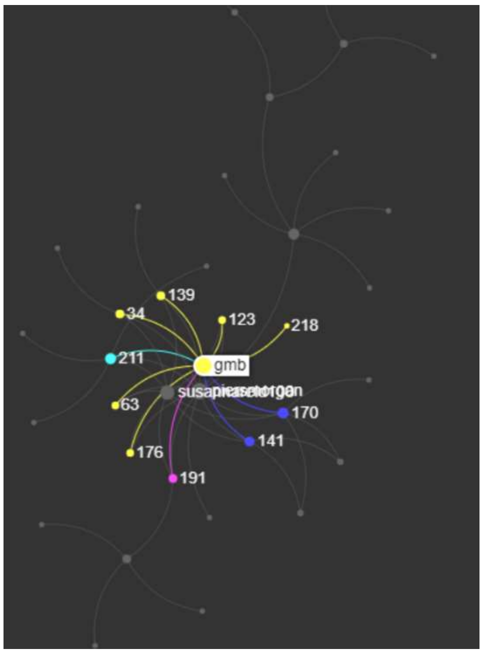


Figure 3 – User interaction graph for user mentions highlighting network with @gmb.

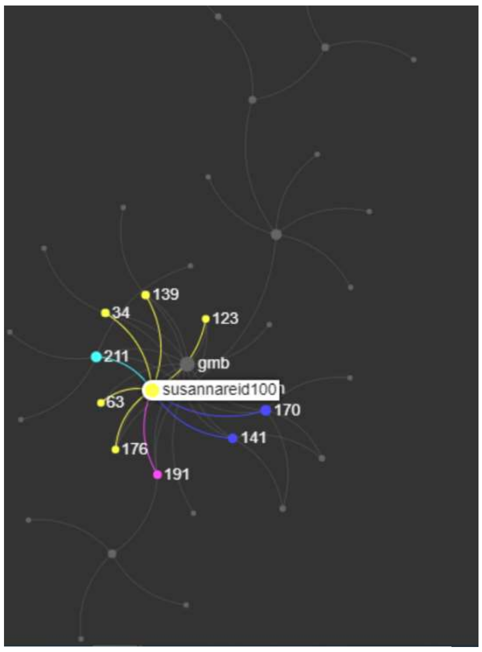


Figure 4 – User interaction graph for user mentions highlighting network with @susannareid100.

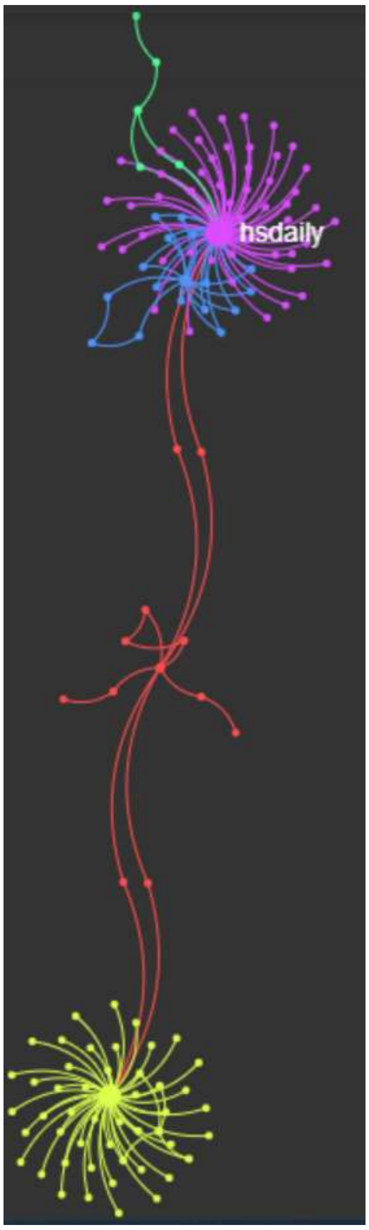


Figure 5 – User interaction graph for Retweet network.

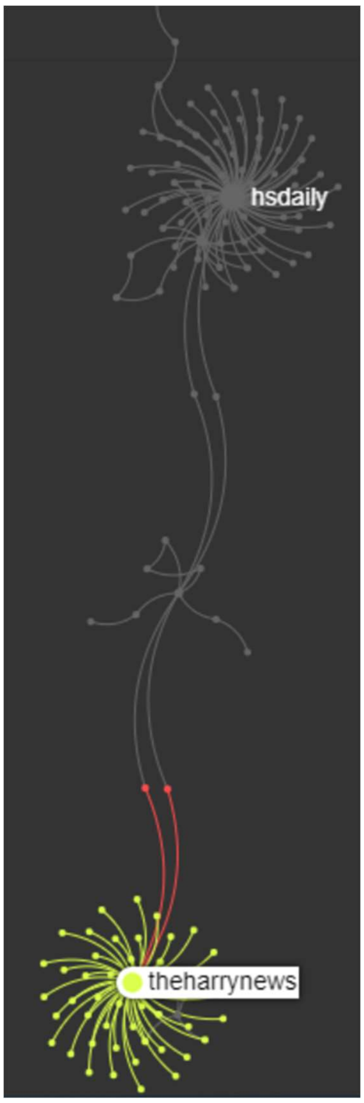


Figure 6 – User interaction graph for Retweet network highlighting @theharrynews.

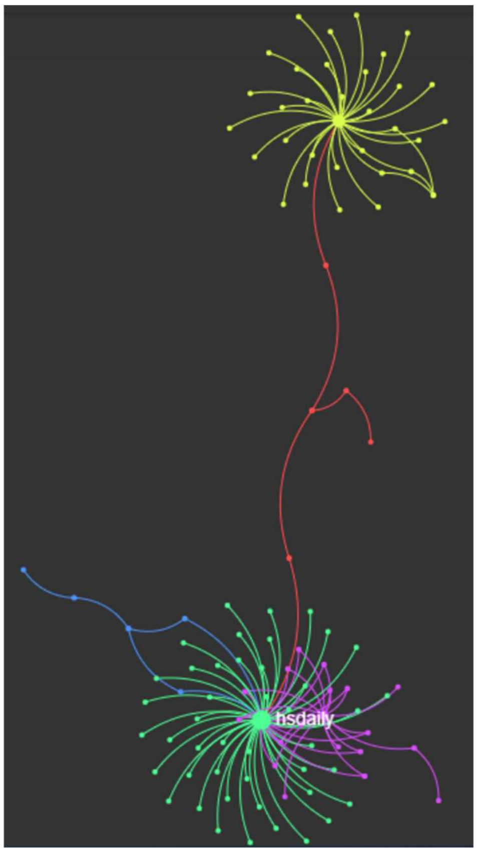


Figure 7 – User interaction graph for general Twitter data.

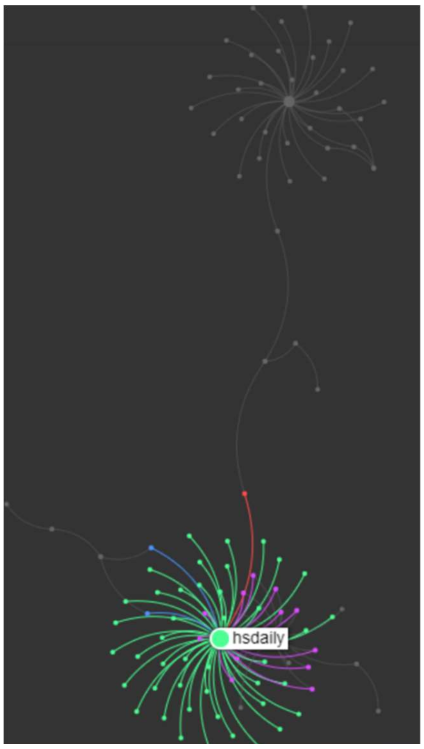


Figure 8 – User interaction graph for general Twitter data highlighting @hsdaily.

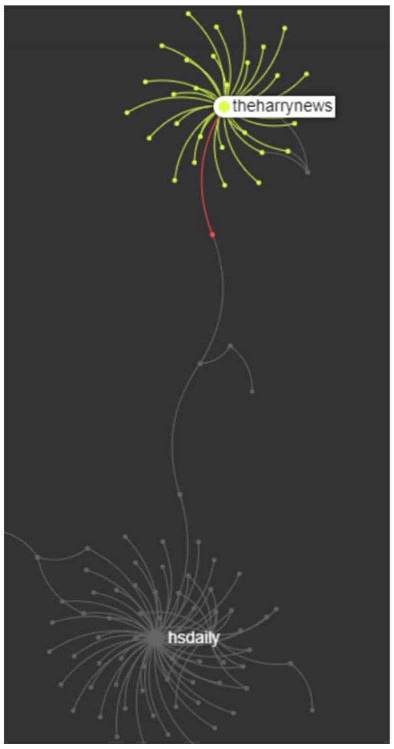


Figure 9 – User interaction graph for general Twitter data highlighting @theharrynews.

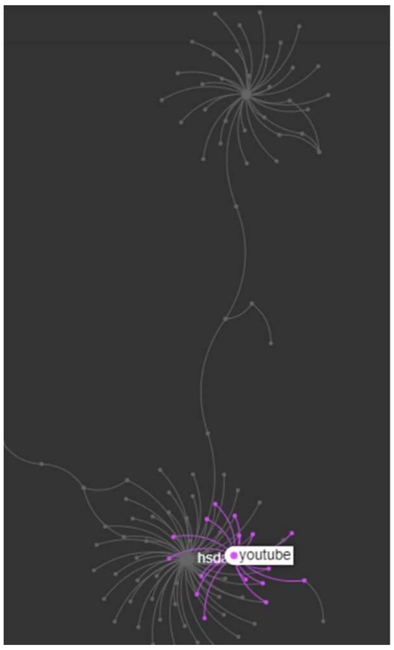


Figure 10 – User interaction graph for general Twitter data highlighting @youtube.

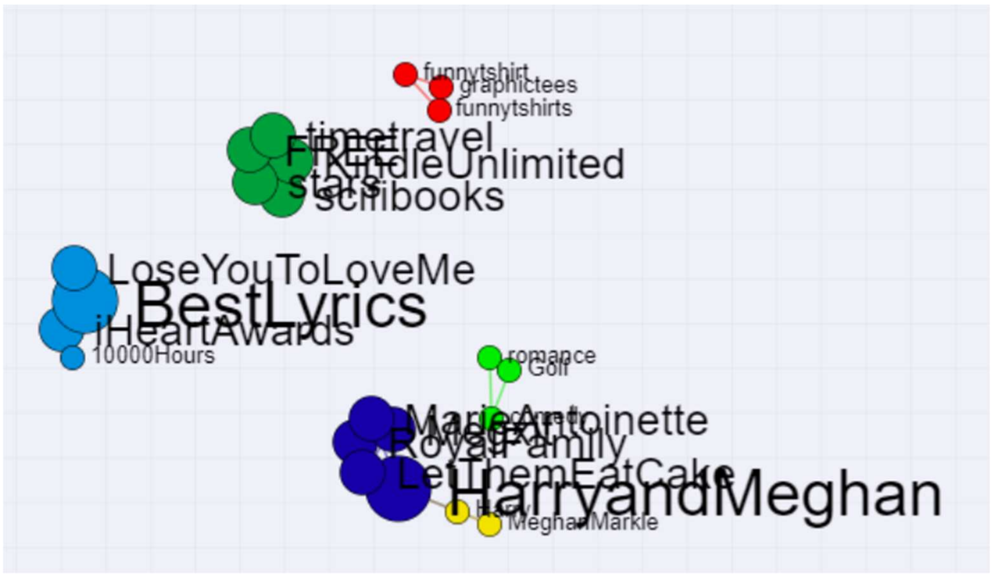


Figure 11 – Hashtag co-occurrence graph.

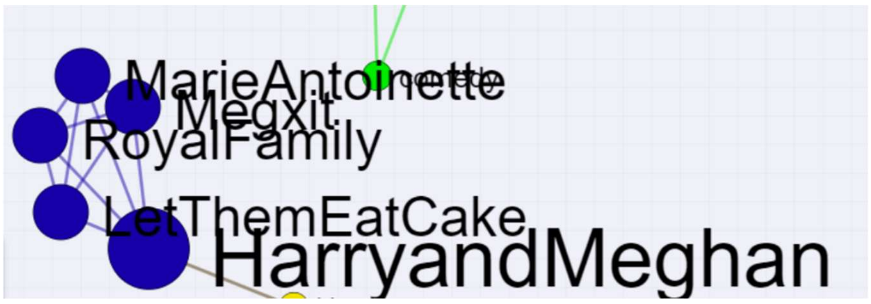


Figure 12 – Hashtag co-occurrence graph, #HarryandMeghan group.

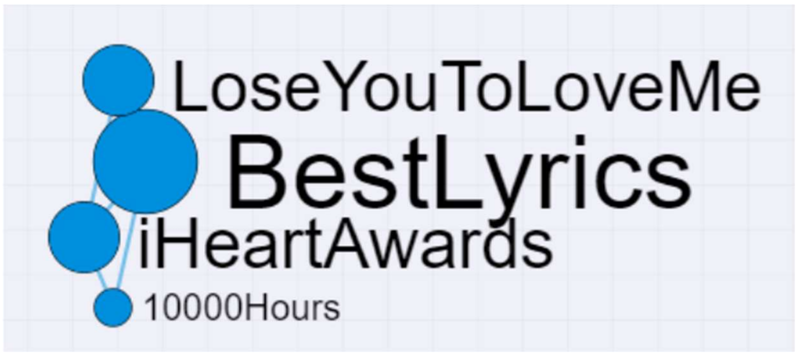


Figure 13 – Hashtag co-occurrence graph, #BestLyrics group.

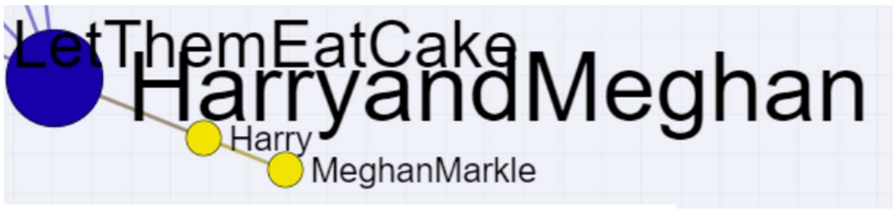


Figure 14 – Hashtag co-occurrence graph, #Harry group.

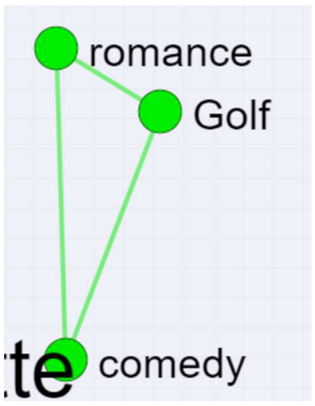


Figure 15 – Hashtag co-occurrence graph, #comedy group.

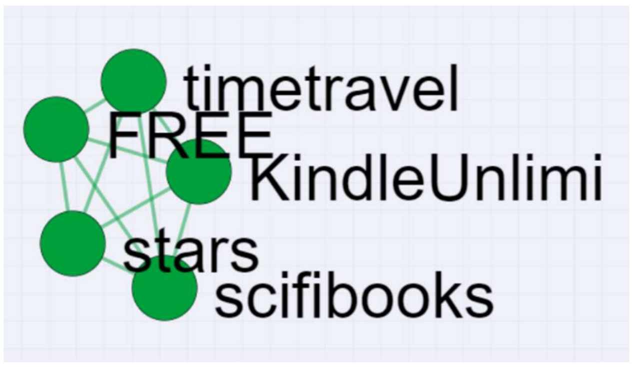


Figure 16 – Hashtag co-occurrence graph, #timetravel group.

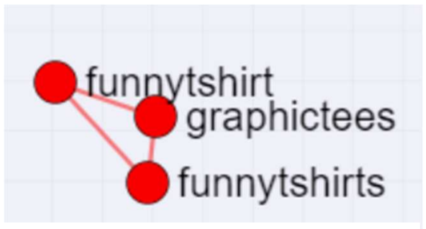


Figure 17 – Hashtag co-occurrence graph, #funnytshirt group.