Clustering Tweets, Users and Their Following Lists in A Network Graph Based on a Specific Hashtag

Brigitta Jesica Kartono [03692769][1,✉], **Karahan Yilmazer [03717524]**[1,✉]
[1]Department of Electrical and Computer Engineering, Technical University of Munich (TUM), Arcisstr. 21, 80333 Munich, Germany
✉ jesica.kartono@tum.de, karahan.yilmazer@tum.de
March 21, 2021

1 Introduction

Twitter is one of the most popular social media platforms and hence a strategic source for data analysis. In this project the authors aim to research whether there is a relation between the cluster a user belongs to in a network graph and the content of this user's tweet based on a hashtag.

To reach a conclusion, 1000 tweets with a particular hashtag, #cryptocurrency in this case, the usernames of the users that posted these tweets, and these users' following lists were streamed.

For visualization purposes a network graph of every user's following list was created. This graph allowed to see if the studied users followed mutual accounts and whether they followed each other as a community that shares common interests. The 1000 tweets were also clustered using k-means clustering and these clusters were overlayed on the network graph to observe if there is any correlation between the user connections and their sentiment.

2 Theory

I. NETWORK GRAPH

A network is a set of objects (called nodes or vertices) that are connected together. The connections between the nodes are called edges or links. [1] A network can be directed or undirected (bidirectional). What determines this property is whether the edges have a preferred direction. In this case the nodes are the Twitter users and the edges are their following relations amongst each other. The network is directed since a user following another doesn't necessarily mean that the followed user also follows its follower.

Trivially, the interconnections of the nodes are based on the users' followings and it is logical to assume that some prominent users would have more connections than the average. The technical term to define these prominent users are "hubs" which are highly connected nodes. [2] These hubs should somehow correspond to the hashtag #cryptocurrency, as that is a determining factor for this project.

II. K-MEANS CLUSTERING

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. [3]

K-means algorithm partitions n observations into k clusters where each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster. [3]

There are various methods of clustering but for this project k-means clustering was picked, since it is one of the simplest unsupervised learning algorithms. Moreover it was important to use an unsupervised learning algorithm, because the data labels were not known beforehand.

3 Experimental Setup

I. DATA MINING AND DATA SORTING

The first step of data mining was to collect the tweets using the Python library Tweepy [4] to access Twitter Stream API, which requires a Twitter Developer Account. Twitter API allows a developer to stream Tweets in real time. [5] For this project, authors utilized a standard API, which is free of charge, but has limitations for the amount of data obtained: it allowed only 300 followers to be collected every 15 minutes.

First, 1000 tweets which contained the hashtag #cryptocurrency were streamed and saved locally as a TXT file. The information obtained from this stream contained a lot of unnecessary details, which were filtered out in the next step. At this point the tweets were saved as a JSON object to make data extraction easier. JSON objects are written in key/-value pairs, hence the values in a JSON object can be accessed relatively easily. [6]

After 1000 Tweets were obtained, the full text of each tweet, the username and the user ID of the

user that posted the tweet were extracted to be stored separately in a CSV file. This was done by parsing over JSON objects by their keys to obtain the values needed. Saving the data as a CSV file was preferred for easier processing with Pandas, a data manipulation and analysis software library. [7]

The Twitter API was then used again to obtain each user's following list. Each list was stored locally as a CSV file under the name "followinglistx.csv", where x denotes a number between 0-999. The following list of the user who tweeted the first tweet would have the file name "followinglist0.csv". To avoid unnecessary repetitions, a conditional statement was implemented: if the following list was already obtained for a particular username, then the list was not fetched a second time thus leaving it empty. This was done because it does not make a difference for the network graph in either case, as the graph also prevents repetitions to maintain objectivity.

II. PREPROCESSING AND CLUSTERING

Once the data had been fetched and stored, it was ready to be processed, making the next step the clustering of the tweets and thus the users that posted those tweets using the k-means clustering method. Before starting with the actual clustering, the tweets had to be preprocessed for better performance.

The preprocessing mainly consisted of cleaning up the tweets. All letters were converted to lowercase so that capital letters didn't play a role in the clustering. Line breaks, multiple spaces, mentions, hashtags, retweets, URLs, emojis and special characters were removed.

After this basic step, stopwords were discarded. Stopwords are commonly used words whose presence in a sentence has less weight compared to other words. They include words like 'and', 'or', 'has' etc. [8] Then the tweets were tokenized, meaning that the strings were split into a list of tokens (mostly words in this case), and lemmatized, meaning that the words were reduced to their root form. For these steps the Python library NLTK was used. The Natural Language Toolkit (NLTK) is a Python package for natural language processing. [9]

All the above mentioned preprocessing steps are fundamental for efficient language processing, since they greatly improve the performance of the clustering algorithm.

The preprocessed data was now ready for further processing. To get a general idea of the data, a word cloud was generated out of the tokenized tweets using the WordCloud library [10]. Since every tweet had the hashtag #cryptocurrency, it was filtered out of the word cloud. The result of this step can be seen in Figure 1.

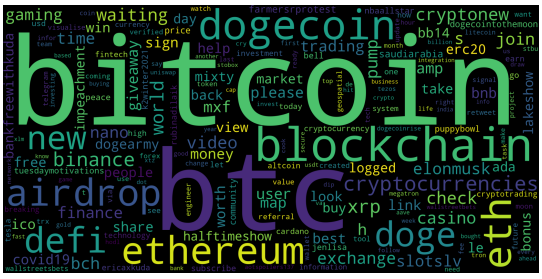

Figure 1 Word cloud out of the fetched tweets

The next step was to gather keywords for the topic. For this purpose an online glossary [11] was used and the keywords taken from the glossary were stored in a string variable. To have a measure of relevance to the topic, the Jaccard similarity was used. The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used in understanding the similarities between sample sets. [12] The compared sample sets in this case were the tweets and the string involving all the keywords.

Further features for clustering were obtained by using the TextBlob library, a Python library for processing textual data. It provides a simple API to dive into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification and more. [13] For simple sentiment analysis purposes two main features were extracted from each tweet: subjectivity and polarity. Both of these properties provide scores to explain how subjective/objective or negative/positive the given text is. The subjectivity score ranges from 0 to 1 with 0 being the objective and 1 being the subjective end. For an easier understanding the polarity scores were further categorized into three groups: negative (polarity < 0), neutral (polarity = 0) and positive (polarity > 0).

To find an optimal k-value for the k-means clustering the elbow method was used. In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method plots the explained variation as a function of the number of clusters, and picks the elbow of the curve as the number of clusters to use. [14]

As can be seen in Figure 2, the number of clusters, so the k-value, is on the x-axis whereas the Within-Cluster Sums of Squares (WCSS) is on the y-axis, where WCSS serves as the cost function. Examining the graph, a good pick for the k-value was 3, meaning that the tweets were going to be separated into three clusters.

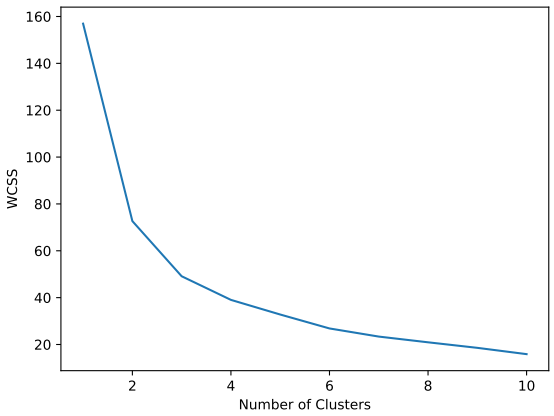

Figure 2 Elbow Method to Determine The Optimal k-Value

At last, the tweets could be clustered based on the three extracted features: Jaccard similarity index, subjectivity and objectivity. To fit the data with k-means clustering, Scikit-Learn library [15] was used. The results of the clustering can be seen in Figure 3. The computed cluster numbers were stored in the CSV file containing the fetched data to be later used in the network graph.

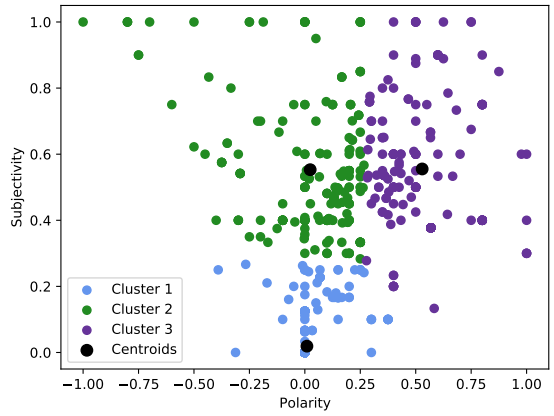

Figure 3 k-Means Clustering on the Fetched Tweets

II. CREATING THE NETWORK GRAPH

Creating the network graph required the Python library NetworkX, a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. [16]. For each following list, first a node was created for the following list's owner, and then an edge was created for each of the users in the following list. The created network was a directed network with the direction being from the follower to the followed. Some conditional statements were added to make sure that when a username was encountered again, the old node was expanded instead of making a new node.

Each time a new graph was created from a following list, it was "merged" with the graph composed of previous lists. Hence the complete graph was built iteration after iteration by adding each new graph to the rest.

As the graph was very massive in size, Gephi, a visualization and network analysis software [17], was used. NetworkX itself has a draw function which draws graph directly into a matplotlib window, as well as functions to do graph clustering or to set node sizes. However, for big scale graphs this approach is very time consuming. It is also not possible to analyze and interact with the graph on the created file post-creation through the NetworkX draw function. Due to all these reasons, Gephi was used, as mentioned before.

For analysis and double checking reasons, a function to check the prominence of users was also implemented. This function counted the number of times a user was followed throughout the 1000 following lists, and also returned a list with the most followed users among the network. Again, repetitions were discouraged as the function checked if two following lists belong to one username, and thus went through it only once instead of multiple times.

III. NETWORK ANALYSIS IN GEPHI

At first sight, the imported graph to Gephi was completely messy. The first step to improve the visibility was to apply the Force Atlas 2 layout. This layout is geared for large networks. It allows a rigorous interpretation of the graph with the fewest biases possible by focusing on quality, and offers a good readability. [18] The scaling was set to 4.0, which determines distance between nodes.

The next step was to categorize the clusters by colors. This was done by running one of the built-in filters from Gephi, namely modularity. Modularity is a quality index for clustering. [19] This algorithm looks for nodes that are more densely connected to each other than to the rest of the network. Modularity resolution was set to 0.5 since values lower than

1.0 get more communities. The computed modularity scores were then used color the clusters.

To better visualize the importance of a node, the node size was set according to how prominent that user was in the network. This was accomplished by running the eigenvector centrality filter from Gephi. This algorithm measures the importance of a node by the level of influence it has in the network. The calculated eigenvector centrality scores ranging from a minimum size of 10 to a maximum size of 150 were used for the ranking size of the nodes. Simply put, this process makes the prominent nodes larger than the rest.

Lastly, another built-in function to prevent overlap of the nodes was used, which allowed to decide how much of the overlapping to handle without disturbing the graph's visual cohesion.

After all these steps, the graph had nodes whose sizes were proportional to their prominence, and nodes were categorized by colors into different clusters, as it can be seen from Figure 4.

Figure 4 Network Graph with Modularity Clustering

IV. OVERLAYING THE TWEET CLUSTERING WITH THE NETWORK GRAPH

As the last step, clusters acquired with k-means clustering on the tweets could be imported to the network data sheet in Gephi. This allowed coloring the nodes based on these three clusters and an extra "cluster" for the initial 1000 users the tweets were fetched from. The results of this step can be seen in Figure 5.

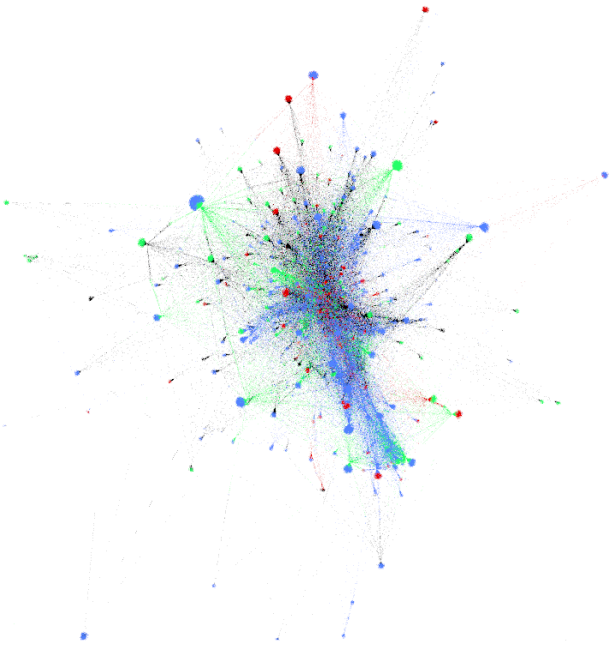

Figure 5 Network Graph with Tweet Clustering

Clustering the tweets only clustered the 1000 users that posted those tweets. This was not enough for visualization purposes since the rest of the users in the network graph, from whom no tweets were fetched, weren't assigned to any clusters. To assign these users to one of the three clusters a straightforward solution was implemented: assigning the followed users to the clusters of their respective follower. This approach helped with the visualization of the network graph to achieve the look one might see in Figure 5.

4 Results

As mentioned in the introduction, this project aimed to see if there is a relation between the cluster a user belongs to and the content of this user's tweet based on a hashtag. Since the chosen hashtag was #cryptocurrency, the created word cloud had to reflect the topic. As it can be seen in Figure 1, even though only a very small subset of all the tweets with the #cryptocurrency hashtag were analyzed, the word cloud gives a great overview of the topic. Being the first cryptocurrency created, bitcoin and its abbreviation "btc" are the most commonly used words, showing that it still protects its popularity even after the rise of different cryptocurrencies. Having a very central meaning for cryptocurrency, "blockchain" came second. Other cryptocurrencies such as "ethereum" and "dogecoin" with their recent

popularity can also be seen to be of importance for the chosen hashtag.

As it can be seen in Figure 3, the tweets in cluster 3 are strictly positive with moderate subjectivity. The more objective tweets belong to cluster 1 with no apparent extremes when it comes to polarity. The tweets in cluster 2 include rather subjective tweets with less positive sentiment. One has to bear in mind that the Jaccard similarity index was also a feature and even though it doesn't have a dedicated axis in the 2-dimensional plot, it is embedded in the distribution and clustering of the data points.

With 212927 nodes and 296044 edges, the network graph gives important insights to the intrinsic dynamics of the users. Although it was decided before starting with the project that the main clustering method for the network graph to be the language processing of the tweets, modularity of the nodes also served as a great way of determining local hubs and clusters. Furthermore the eigenvector centrality ranking and the written function to find the most followed users showed that the most prominent users were Elon Musk, Coinbase, Binance and CoinDesk.

Coming to the tweet clustering, the results were surprising since the authors expected more global localization of the clusters than the seen results. Looking at Figure 5, it can be said that the biggest cluster, being the blue cluster which correspond to cluster 1 from Figure 3, is rather focused on the center and not spatially distributed whereas the other clusters (red being cluster 2 and green being 3 from Figure 3) are on the peripheries and mostly separated from each other. Furthermore the results demonstrate local hubs and clusters to be existent and that the network is highly interconnected, as expected.

5 Conclusions

This project demonstrated the relation between the user and tweet clusters. Although the results were rather surprising, two possible explanations for the rather weak localization of the clusters in the global scale could be that either the data wasn't large enough to reach reasonable conclusions or that the performance of the clustering algorithm was low. It would be very interesting to see the results of this project being reproduced with bigger data and more accurate language processing done for different topics and social networks.

References

[1] D. Q. Nykamp, "Network definition," https://mathinsight.org/definition/network, accessed: 2020-03-17.

[2] A.-L. Barabási, *Network Science*, 2016.

[3] S. Priy, "Clustering in machine learning," https://www.geeksforgeeks.org/clustering-in-machine-learning/#:~:text=Clustering%20is%20the%20task%20of,data%20points%20in%20other%20groups., 2020.

[4] J. Roesslein, "Tweepy: Twitter for python!" *https://github.com/tweepy/tweepy*, 2020.

[5] The Twitter API developer team, "Twitter api," https://developer.twitter.com/en/docs/twitter-api, accessed: 2020-03-07.

[6] C. Manu, "A script to download all of a user's tweets into json," https://gist.github.com/manuchandel/bc8a6ca4b1527b7594945e5091013905, 2016, accessed: 2020-03-07.

[7] The pandas development team, "Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[8] A. Kibet, "Tweets classification and clustering in python." https://medium.com/swlh/tweets-classification-and-clustering-in-python-b107be1ba7c7, 2020, accessed: 2020-03-17.

[9] E. K. Steven Bird, Edward Loper, *Natural Language Processing with Python*, 2009.

[10] A. Mueller, "Wordcloud for python documentation," http://amueller.github.io/word_cloud/, accessed: 2020-03-17.

[11] C. Stead, "Crypto a to z: Cryptocurrency glossary," https://www.finder.com/cryptocurrency-glossary, accessed: 2020-03-17.

[12] R. Real and J. M. Vargas, "The probabilistic basis of jaccard's index of similarity," *Systematic Biology*, vol. 45, no. 3, pp. 380–385, 1996. [Online]. Available: http://www.jstor.org/stable/2413572

[13] S. Loria, "Textblob: Simplified text processing," https://textblob.readthedocs.io/en/dev/, accessed: 2020-03-17.

[14] Wikipedia, "Elbow method (clustering)," https://en.wikipedia.org/wiki/Elbow_method_(clustering)#:~:text=In%20cluster%20analysis%2C%20the%20elbow,number%20of%20clusters%20to%20use., accessed: 2020-03-17.

[15] T. S.-L. developer team, "sklearn.cluster.kmeans," https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, accessed: 2020-03-18.

[16] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, 2008, pp. 11 – 15.

[17] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154

[18] The Gephi development team, "Gephi tutorial layouts," https://gephi.org/tutorials/gephi-tutorial-layouts.pdf, 2011.

[19] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, pp. 172–188, 2008.