Social Media Community Identification and Analysis: An Overview

Franklin Louis
Indiana University
fblouis@iu.edu

Hithesh Busetty
Indiana University
bhithesh@iu.edu

Jongwon Lee
Indiana University
leejojo@iu.edu

Joy Zayatz
Indiana University
jzayatz@iu.edu

Tanukrishna Chetia
Indiana University
tchetia@iu.edu

Abstract

The proposed article discusses the topic of community detection and analysis in the context of Social Media. We discovered diverse libraries and tools that are available for solving a network science problem. The most efficient way to get started was to use the Twitter API's search function to collect Twitter data, cleanse/transform with pandas, then use NetworkX to create a graph and compute graph analytics. Gephi was the easiest way to stream Twitter data and visualize a graph exported from NetworkX. We did not explore R packages as much as Python ones, but similar libraries are available in both. Neo4j was more suited for large datasets, but a toy dataset should be used to learn how to write Cypher queries. While Neo4j does not have as many graph algorithms in its Graph Data Science Library as are available elsewhere, if those properties are written into the graph data, Neo4j is a good way to isolate and visualize smaller parts of the network or execute complex queries about the network. Cytoscape was very difficult to use and it takes a lot of time to render the Network.

Keywords— Network Science, Natural Language Processing, Social Media, Twitter, Community Detection, Telecritical Care, Wildfire Mitigation, Louvain Modularity

1 Introduction

The widespread adoption of social media has led to a boom in user-generated content. Users in various capacities: professionals, enthusiasts, bystanders, policy-makers, etc. can share experiences, knowledge, and observations and communicate on platforms like Twitter. Social media is increasingly becoming a crucial communication platform for scientists to communicate with peers and general audiences in near real-time (Walter, 2019). Community detection has no universal definition, which makes it challenging to evaluate the performance of an algorithm or a model and, at the same time, allows diverse approaches to problems (Fortunato, 2016). The association of entities such as online users, visual content, and metadata leads to the formation of Social Media Networks. Analyzing the structure of such networks can uncover the structure and boundaries of communities within a Social Media Network (Papadopoulos, 2012). In this project, we provide an overview of publicly available network science analytics and visualization tools. We focus on discovering luminaries, who are the most important and influential individuals in a network and detecting communities within the network. Given a collection of Twitter data, the purpose of this project is to 1) identify the most influential individuals in the network (luminaries), 2) model and visualize the network's communities as a graph, and 3) aggregate important tweets (text, hashtags, etc.) in that network.

2 Objective

The scope of the project as a whole is to be able to identify luminaries, visualize network structures, and aggregate important tweets for a collection of Twitter data. The end product would include an interface for users to import data, identify and visualize then run and visualize their network of interest. This is the first semester of this project. We establish a foundation of background knowledge, explore methods and tools, and create a prototype that is a proof of concept for data ingestion, cleansing, analysis, and visualization. Moreover, we explore tools/methods and identify the pros and cons of each for use in future work. Also, we suggest repeatable/reusable methods that can be used by different interest groups.

- Provide an overview document to help future students understand the concepts used in this project, including network science, graph

databases, a survey of analytical tools, etc.

- Create ELT pipeline to collect and store Twitter data, and transform/cleanse into the format required for analytical tools
- Implement model(s) for community detection and other network analytics
- Visualize networks by their communities
- Build a suggested roadmap for the future groups to continue the project

3 Dataset and Methods

We aim to build a Twitter Luminaries Model to examine the tele-critical care domain, use network analytics to identify influential members of a network, and apply community detection algorithms. To do so, we collect a set of Tweets containing words about tele-critical care from the Twitter API V2.

3.1 Dataset

We use Tweepy and Jupyter notebook to collect Twitter data. We store this data in a MongoDB Atlas cluster. The data stored in MongoDB are stored in three collections

- luminary_tweets: 677,399 tweets from accounts listed in the luminaries list. This dataset is the payload from the API without modification.
- tweets_telecritical: 5,914 tweets that talk about tele-critical care. This dataset includes the API payload plus the user ids that retweeted them, and any relevant tweets the retweeter then tweeted.
- users_telecritical: 4,313 users who authored the tweets in the tweets_telecritical collection. This dataset includes the API payload plus the user ids of the accounts the users are followed by and are following.

3.2 Methods

We use the initial set of tweets to build the next layer of the network by connecting the original set of tweets and users with various types of relationships, e.g. followers, retweets, and mentions, then iterate to create each successive layer of the network. Once the network is constructed, we implement network analytics to calculate metrics such as centrality and modularity (measures the strength of division of a network into modules) on the constructed network. Lastly, we visualize the network by calculated analytics. The list of libraries and technologies used in this project are as follows:

- Twitter API V2: Elevated Twitter Developer account, Tweepy Data ETL, cleaning: pandas
- Network Analysis: NetworkX, igraph, Gephi, Neo4j Graph Data Science Library, Cytoscape
- Graph database: Neo4j Desktop, Py2neo
- Graph visualization: Matplotlib, Gephi, Neo4j Bloom, jgraph, neo4jupyter, d3.js, Cytoscape

4 Observations

4.1 Data collection and storage - Tweepy and MongoDB

We discovered using Tweepy to collect Twitter data and storing it in MongoDB was well suited for our project. Leveraging both methods in a jupyter notebook allowed us to collect various types of data. During this process, using sleep timers prevented us from being caught by Twitter's API rate limits. MongoDB's semi-structured data schema quickly ingests the json format that comes from the Twitter API. The fetched data in csv or json format is cleaned/transformed using pandas. We used Gephi's Twitter plugin to stream Twitter data. Gephi has a statistics feature that generates a visualization to check if the query used to filter the Twitter stream is generating the type of data you are interested in.

4.2 Network Construction and Analysis - NetworkX, Neo4j

Network construction from pandas was a good way to make sure the API queries are returning relevant data. The values selected for max_limit returned by each API query (100 was the largest number we used, and we also used 50 in some cases) were too small to distinguish nodes from each other. Community detection and centrality algorithms provide more useful information when the network nodes are too similar. For analysis, NetworkX had the most complete library (or compatible with other libraries) of graph analytics. NetworkX's methods for creating networks from pandas dataframes are also a convenient way to build a network quickly.

We did not use any large datasets in this project, but it seemed that NetworkX started taking longer

times to compute network statistics even in networks with greater than 5000 nodes. Neo4j is advertised to be scalable to large datasets. We found the learning curve to begin using Neo4j is steeper than with NetworkX (may take up to 3 weeks to learn how Neo4j works and Neo4j's Cypher query language), but Neo4j is specialized to work with graph data and has much more capability to store and query attributes for nodes and edges. Neo4j's Graph Data Science Library is a powerful tool and is advertised to be able to be scaled for large data sets, but the algorithms available are not as numerous as NetworkX.

4.3 Graph Visualization - Neo4j, Neo4j Bloom, igraph, Gephi, Cytoscape

NetworkX has built-in methods to visualize graphs with Matplotlib. This quickly shows a general picture of what we are working on with NetworkX in a jupyter notebook. However, we found that exporting the NetworkX data to Gephi was an easier way to visualize the graph data by network statistics with various layout options because you can do everything within the graphic user interface rather than creating it in lines of Python. Neo4j is useful because it can return query results as either tables or graphs, so if you want to "zoom" in on a certain aspect of the graph, you can query for it and see the graphical depiction without an extra step. Neo4j's Bloom application in the desktop client is also fairly capable and easy to start using. Bloom allows you to filter and query for specific nodes and relationships and change the appearance of notes and relationships by analytics already written in the graph. A limited number of graph algorithms can be run in Bloom, but it's not many, and the data has to be in a Neo4j sandbox, rather than a local graph DBMS. Bloom does not have the range of layout options that Gephi does.

5 Instructions

5.1 Tweepy and MongoDB

- Twitter API V2: The data collected from Twitter requires an Elevated Twitter Developer account. (An Academic Research account would provide a higher monthly quota and more access to Twitter data)
- MongoDB: We have set up a MongoDB Atlas cluster to store twitter data and added fields, e.g., a list of follower user ids, so the network can be recreated using data stored in MongoDB, rather than querying the Twitter API again.

5.2 NetworkX

We used Tweepy to collect the tweets and user data from the 7-day lookback search. We started with a group of tweets containing relevant text and created 2 types of networks. In both types of networks, the "source" and "target" nodes are combined in a Pandas Dataframe that can be used by NetworkX to create a graph. We also performed network analysis with NetworkX, including community detection with Louvain and Leiden algorithms.

- Network of followers: Nodes are users, and edges are followers. Nodes representing followers have edges directed back to the user they are following. Because the Twitter API has the capability to find a user's followers and who the user is following, the network can be built in both directions from each user node. Start with a set of tweets, find the tweet that has the highest public metric score, and select the author of that tweet at the luminary. Find 100 users that follow the luminary. Find 100 accounts that the luminary is following. For each follower, find 50 accounts that they follow. Repeat twice. Compute network analytics and visualize with NetworkX/matplotlib and Gephi

  Figure 1 is created with Gephi's Circle pack layout and groups the network by modularity. Clusters with the largest number of nodes are colored with colors other than gray. Figure 1 shows that the network has many small clusters and a few large clusters. This indicates that nodes in this network do not have many edges connecting to other nodes.
- Network of Retweets: nodes are users and edges are retweets. Nodes representing retweeters have a directed edge point back to the original author. The relevant tweets of the retweeters are used to build the next level of the network. Start with a set of tweets, find tweets that have the highest public metric score, get retweeters for those tweets, and find retweeter's relevant tweets. Use the retweeter's relevant tweets to build the next layers of the network. Repeat twice. Compute network analytics and visualize with NetworkX/matplotlib and Gephi

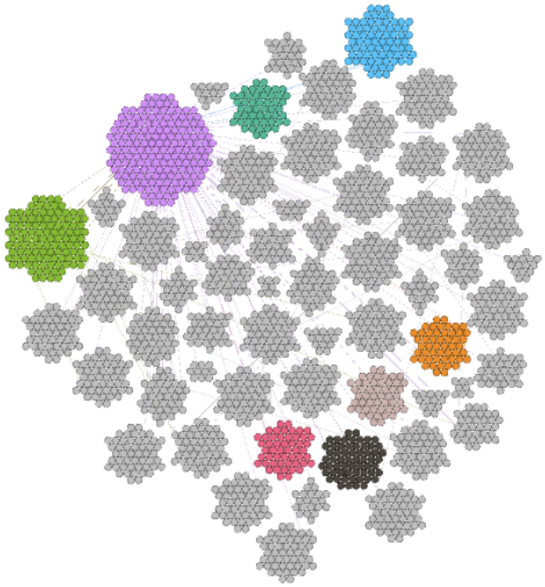

Figure 1: Follower network grouped by modularity

Using the same method to visualize the retweet network, in Figure 2 we can see there are fewer clusters in the retweet network with one large cluster.

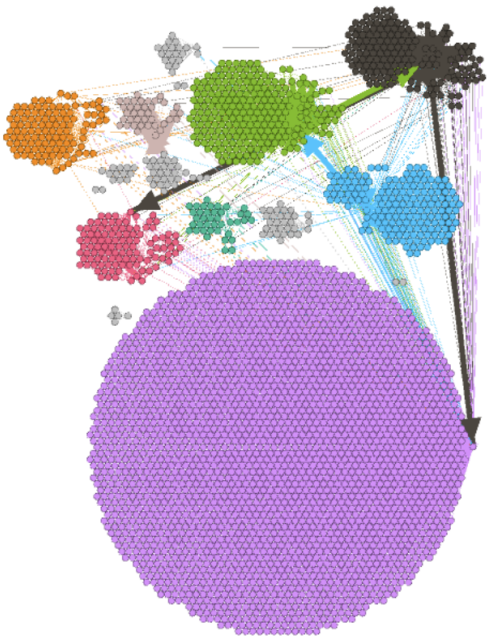

Figure 2: Retweet Network grouped by modularity

- Network of Mentions (Using Cytoscape & Gephi): Nodes are tweet authors and persons mentioned in the tweets and an edge is mentioned between the tweet author and the person who is mentioned. This network will help identify the luminaries who are most mentioned or interactions between the different luminaries.

  Start with a set of tweets from a list of luminaries, the data must contain data of tweet text and author data of tweets. Perform NLP on tweet text to extract the names of persons mentioned in the tweet. Load this dataset into Cytoscape or Gephi and perform analysis. Perform statistics with Gephi and do analysis on degree, In degree, out-degree, modularity, and Statistical inference to identify communities.

Figure 3: Mentions Network grouped By Statistical Inference

5.3 Neo4j

- Figure 4: After building the Followers and Retweet network, we construct a network with tweets and users data. Next, load nodes and relationships from csv. Then, Compute network analytics with Graph Data Science Library, visualize in Neo4j Bloom, Neo4jupyter, and jgraph.
- Figure 5: The nodes are colored by Louvain clusters. There is a large number of clusters with few connections. There is one cluster in the middle of the left edge that consists of nodes that do not have a better fit than any other cluster.

5.4 igraph

We created a bigram network with R to analyze text contained in tweet data. The network con-

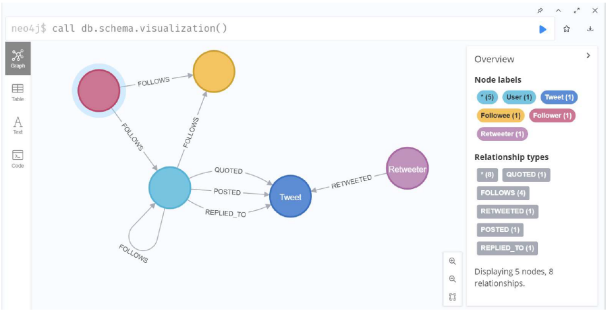

Figure 4: Neo4j schema for the combined network

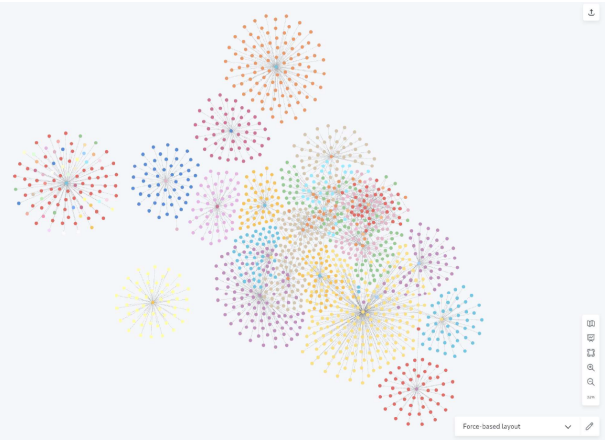

Figure 5: Combined network visualized in Neo4j colored by Louvain clusters

nects pairwise occurrences of words that appear in a single tweet.

5.5 Gephi

We used Gephi with the Twitter API v2 plug-in as a self-contained environment to collect streaming data, analyze the network, and visualize the results. We also found Gephi to be a useful tool for visualizing and analyzing data exported from NetworkX.

5.6 Cytoscape

We created a mentions network in Network. We felt Cytoscape is very difficult to use and it takes a lot of time to render the Network. We had to reset and restart the work sessions several times.

6 Conclusion and Discussion

This project provides data collection methods, data sets, network science analytics, and visualization methods. We were able to cover popular network libraries for Python, R, and standalone software such as Gephi, Cytoscape, and Neo4j. We could ingest, analyze and visualize various networks with different methods. The visualization can be presented either on a web page or packaged, and sent out as email newsletters, or even, as a curated account back to Twitter. We regret not using the first three weeks of the semester to learn more about network science concepts and tools. Having a general understanding of what questions are asked and answered in network science and how NetworkX, Gephi, and Neo4j work would have helped us conceptualize the project and focus on what we wanted to accomplish for the semester more quickly. We believe the Intro to Network Science materials, datasets that are collected and cleaned, and code from this semester will provide any following team with a better environment to improve the project. As of the writing of this report (December 2022), the state of Twitter as a company is somewhat unstable. Up to this point, Twitter's data has been open source and available for developers. However, that might not always be the case. Theoretically, the analysis of this project can be done with any data as long as you can assign nodes and edges. We are using Twitter not only because the data is available but also because Twitter is the only public platform that is real-time, used by professionals and a wide level of practitioners.

In the future, we recommend the following possible extensions:

- Build a large, dense, well-connected network: Pick one approach and build a single network that contains as many types of relationships as possible e.g., retweets, follows, mentions, quotes, replies, etc. This will take time to allow users to post/respond to tweets, follow new accounts, etc., and accommodate API rate limits and monthly quotas.
- Identify luminaries, and detect sub-communities: From this network, use centrality measures to identify a group of luminaries, and apply community detection to find sub-communities.
- Create an interest graph: Incorporate Tweet text into the network described above. From tweet texts, Identify interests unique to sub-subcommunity and link sub-communities by common interests, and collect the text of important tweets.
- Consider how to handle large datasets: A larger, more connected network might be bet-

ter suited to be analyzed in Neo4j rather than NetworkX. It might also be useful to explore ways to connect Neo4j, Gephi, and NetworkX. Other adaptations include using a more powerful cloud computing platform, e.g., Google Colab/Cloud Platform, or IU's High-Performance Computing clusters.

- Visualize for the web: Find ways to create a graph visualization with a web-specific library like d3.js from NetworkX and Neo4j.
- Build a web app: Create an interface to allow users to upload/point to/collect data, run centrality and community detection analytics, and see a visualization. Doyle Groves, one of the project sponsors, has an app www.chattersum.com that achieves a similar goal for a larger Twitter data set.

References

Hric D. Fortunato, S. 2016. Community detection in networks: A user guide. *Physics Reports*, 659:1–44.

Kompatsiaris I. Vakali A. Spyridonos P. Papadopoulos, S. 2012. Community detection in social media. *Data Min. Knowl. Discov*, 24:515–554.

Lörcher I. Brüggemann M. Walter, S. 2019. Scientific networks on twitter: Analyzing scientists' interactions in the climate change debate. *Public Understanding of Science*, 28(6):696–712.