Short Programming Project (WBCS015-05) Longitudinal Analysis and Visualization of Network Data

Bora Yilmaz (s3903125)
Supervisor: Mirela Riveni

January 2023

Contents

1 Introduction

In this programming project, the objective is to perform longitudinal analysis and visualization on the PHEME Twitter Rumour dataset in order to gain insights on how network data behaves over time.

1.1 PHEME Dataset

The PHEME dataset, namely "PHEME rumour dataset: support, certainty and evidentiality", is a dataset collected from Twitter and annotated later on for rumour and misinformation. Quoting the publisher, this dataset includes "rumour tweets, collected and annotated within the journalism use case of the project" [1]. The mentioned Twitter conversations are initiated by a rumourous tweet; and the reactions to this tweet form the threads which we iterate over as we analyze the dataset. These tweets have annotation on whether they are misinformation and rumour or not and also if they changed such states over time.

1.2 Network Data

Network data comprises of relationships between entities and their connections. In this context, the PHEME dataset is data organized and annotated from Twitter threads on breaking news. Thus this is a kind of social network data where for example entitites can follow each other and reactions occur throughout time.

1.3 Longitudinal Analysis

Longitudinal analysis is a method used to study changes in variables over time. It involves collecting data within the same context at multiple points in time, and then analyzing the data to identify trends and relationships between variables. The collected data in our project is the PHEME dataset. Analyzing will be done through visualizations and key values, as seen in the upcoming sections. Since the PHEME dataset does not include and provide labels on all activity on Twitter related to the events, we can not reach concrete conclusions on the events themselves through this analysis as it would be (very) incomplete with respect to the data at hand. Thus, the objective of this project leans more towards this: to construct a longitudinal analysis approach into network data, especially those like the PHEME dataset which is data collected and labelled from a social media network. This analysis thus provides visualizations and values which are key to formulating hypothesis, from this limited yet insightful data.

2 Implementation

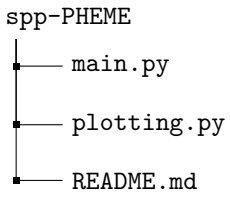
The project consists of two files. `main.py` is for the whole process of reading in, preparing and then working with the data while using the functions from the other file, `plotting.py`.

2.1 Technologies

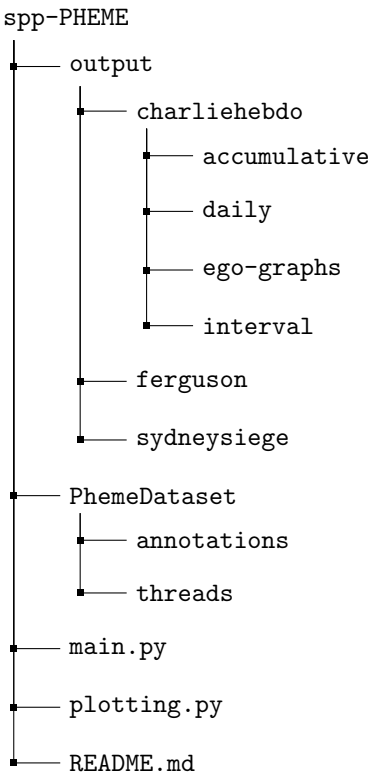
As the project only uses the Python language, we use the most popular modules. `json` is used to load JSON files. `matplotlib` is used to draw all of the visualizations, including the graphs made with `networkx`. The `os` module is used to structure and organize the output folders and files.)

2.2 File Structure

The file structure of the project is organized clearly in order for it to make the navigation of the output, scripts and dataset easy. Before `main.py` is run, there is no `output` folder. So we have the structure as below.



After running `main.py`, the dataset is downloaded and extracted as `PhemeDataset`.`output`folder is created and within this folder, each event has it's own respective folder. Below we see an exaxmple of this where the events on Charlie Hebdo, Ferguson and Sydney Siege were included in the main script.



2.3 Choosing events

In order to choose which events to process, the variable `event_names`, which is a list of the names as strings, must be changed. All possible names are stored in a constant `ALL_EVENT_NAMES`.

```
# For all names, see above of this file (ALL_EVENT_NAMES)
# Decide on the event names to be used
event_names = ['charliehebdo', 'ferguson', 'sydneysiege']
```

2.4 Loading and reading data

For full detail on the process, please see the code, as the all logic is commented on. The (not so detailed) general idea of this part is to iterate all the selected event names by

`for event_name in event_names:`
Then using `os.path.join()`, paths to the events threads are constructed and iterated. `json.load()` is used to read in the files. For each thread that is processed, we do the following to combine them all into one list.

```
# Form the dictionary for the current thread being processed
thread_dictionary = {
            'thread_id': thread_id,  # same as source_tweet id
            'source_tweet': source_tweet,
            'no_of_reactions': len(reactions),
            'reactions': reactions,
            'annotation': annotation,
            'user_follow_dictionary': user_follow_dictionary,
        }
# Add this threads dictionary to the list collection of all threads
thread_dictionaries.append(thread_dictionary)
```

2.5 Organizing data

Now that we have all threads for an event organized in a single list `thread_dictionaries`, it is really easy to extract groups from this data. By iterating it and / or sorting, we create all of the desired groups. We get the groups such as; all reactions, rumour and non-rumour reactions, top thread dictionaries based on amount of reactions or number of people following source tweeter.

One example of this process is below, where we get the threads sorted by number of reactions.

```
    # w.r.t. nr of total reactions (highest to lowest)
    thread_dictionaries_by_reactions = sorted(
        thread_dictionaries, key=lambda d: d['no_of_reactions'],
                                           reverse=True)
```

This is the simplest example that is first seen in the code. The other formulations include more work such as iterating all threads, checking fields like `misinformation` and `users`.

2.6 Visualizing and extracting key values

Now that we have the data structured as mentioned above, we define and call plotting functions on these lists. All of these functions are seen in the file `plotting.py`. The imports are seen at the top of `main.py` as `from plotting import plot_reactions_accumulative, plot_reactions, plot_reactions_daily , plot_ego_graph`
For plots, the main idea is to generate the x-axis and y-axis values as needed.

- interval and daily plots: the x-axis timestamps are generated between the initial timestamp and the latest tweets timestamps, using a selected time-interval. The time-interval can be modified within the function by the variable `min_interval`. The y-axis values are the amount of reactions respective to the timestamp. These values are the amount of reactions within the time window which is from the current time stamp minus the time interval until the current time stamp). The y-axis does not accumulative. This means that the value is reset to 0 for each time interval, so the amount of reactions alternating between 0 and higher values, showing reaction activity occurs over time, with specified time window.

- accumulative plots: the x-axis timestamps are generated between the initial timestamp and the latest tweets timestamps, using a selected time-interval. The time-interval can be modified within the function by the variable `min_interval`. The y-axis values are the amount of reactions respective to the timestamp, but in an accumulative order. This means that the value is not reset for each time interval, so the amount of reactions keeps increasing, showing how it accumulates over time.

- Key values: In some plots, you will see also a `.txt` file in the respective output folders. This file includes key values extracted from the respective plot's context. An example of such file is below, from the interval plot for the thread with most reactees following the source, within Charlie Hebdo event.

  ```
  Time interval used: 5 minutes
  Peak time: 2015-01-07 13:19:33
  Peak value: 12
  Last activity time: 2015-01-07 13:39:33
  Time from source to peak: 0:45:00
  ```

- ego graphs: Ego graphs are created using `networkx` module (written as `nx`). The who-follows-whom.dat file from the dataset is read in using `read_edgelist()` to create a graph. Then functions provided are used to make the graph look nicer and as we want. The ego graph is then saved into the corresponding folder as a `.png`.

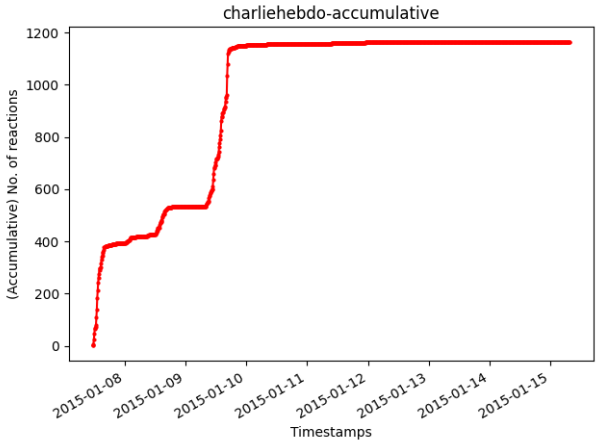
For full details with comments, please take a loot at them in `plotting.py`.

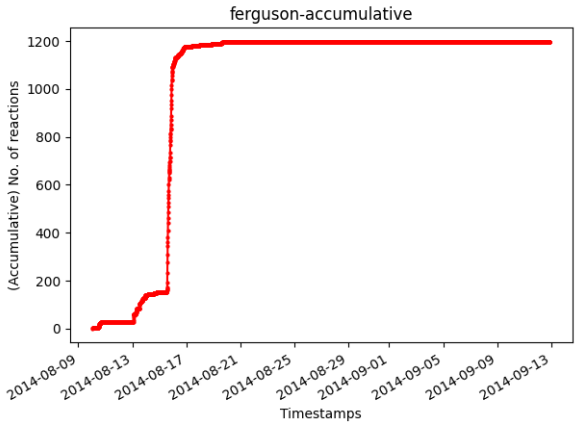
3 Analysis

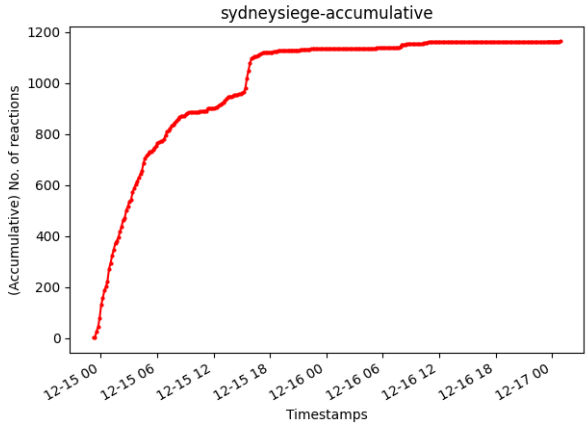
There are many outputs to consider when performing the analysis through multiple events. There is no appendix in this report so in order to replicate and look at such output along with all output, simply running the code as it is without changing the parameters will give the corresponding full output.
We focus on the `charliehebdo`, `ferguson` and `sydneysiege` events for this analysis.

3.1 Accumulative plot on all reactions

In these plots, we see how the accumulative amount of reactions rise and reach the peak before eventually stopping. In these plots, all threads of the given event are combined, so we are basically looking at all threads as a whole over time.



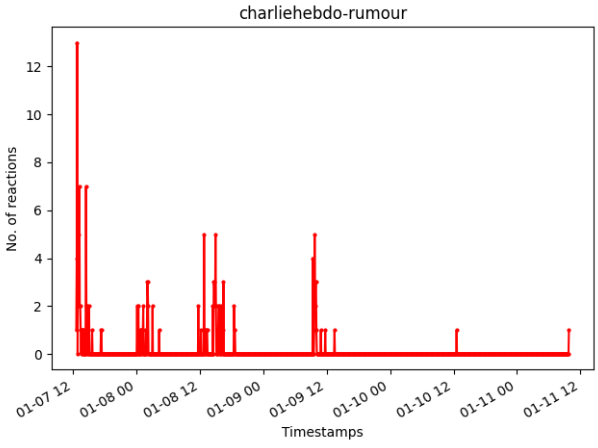




From all of the plots, we see that all events have a similar pattern in which the peak is reached in a very sharp rise, meaning that there was a surge of reactions over all threads in the same time window. Looking at the `sydneysiege` event, we see that the pattern is not clearly seen, yet this is because the threads last only two days so it looks as if the increase is not steep.

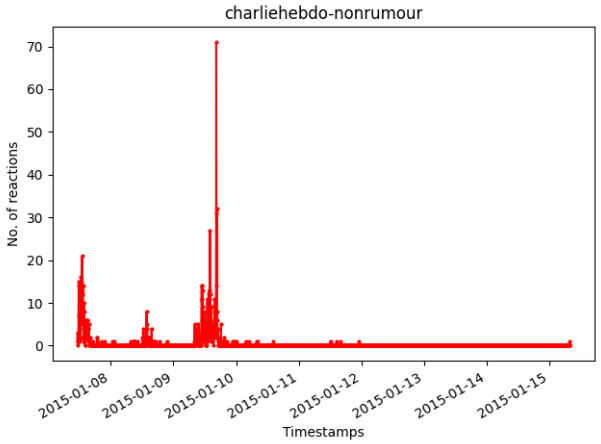
3.2 Interval plots

With these plots, we see how many reactions occur in given time windows, based on the time interval selected. Note that rumour plots reactions does not mean that the reactions themselves are rumours, but the thread that the reaction belongs to has been labeled as misinformation. The set of plots we look at below are over all reactions across all threads.

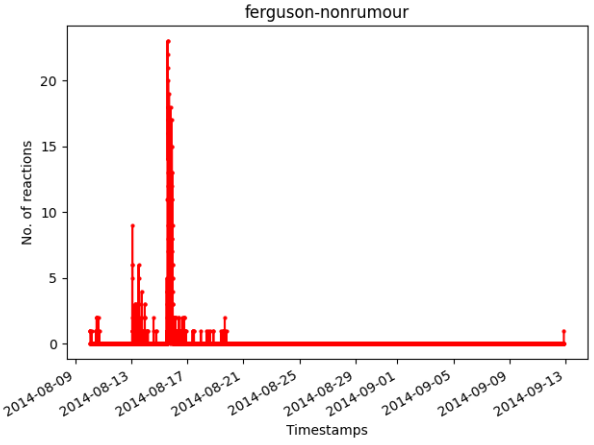



```
Source time: 2015-01-07 12:39:02
```

```
Time interval used: 5 minutes
Peak time: 2015-01-07 12:49:02
Peak value: 13
Last activity time: 2015-01-11 09:49:02
Time from source to peak: 0:10:00
```

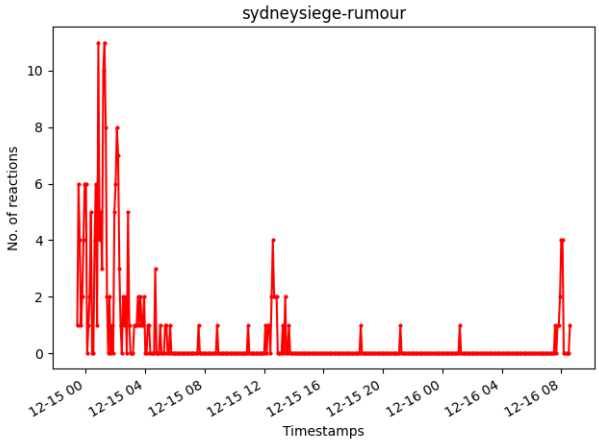



```
Source time: 2015-01-07 11:14:38
Time interval used: 5 minutes
Peak time: 2015-01-09 16:24:38
Peak value: 71
Last activity time: 2015-01-15 07:44:38
Time from source to peak: 2 days, 5:10:00
```

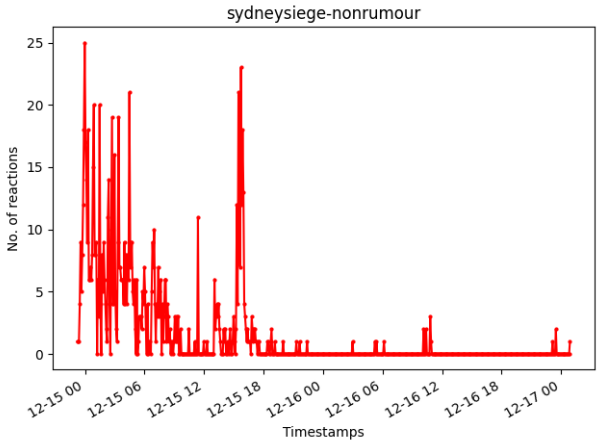



```
Source time: 2014-08-10 01:30:14
Time interval used: 5 minutes
Peak time: 2014-08-15 14:15:14
```

Peak value: 23
Last activity time: 2014-09-12 21:05:14
Time from source to peak: 5 days, 12:45:00



Source time: 2014-12-14 23:25:23
Time interval used: 5 minutes
Peak time: 2014-12-15 00:50:23
Peak value: 11
Last activity time: 2014-12-16 08:35:23
Time from source to peak: 1:25:00



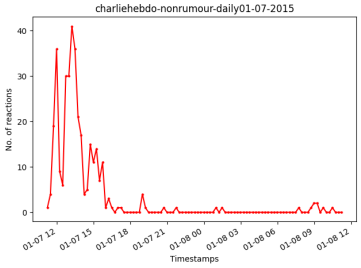
Source time: 2014-12-14 23:13:10
Time interval used: 5 minutes
Peak time: 2014-12-14 23:58:10
Peak value: 25
Last activity time: 2014-12-17 00:53:10

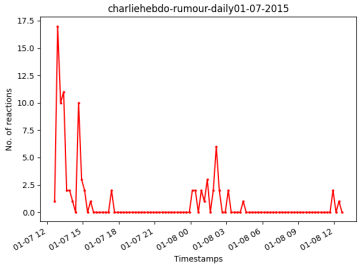
`Time from source to peak: 0:45:00`

In all events, we see a sharp peak in the non-rumour plots, and in rumour plots we see multiple smaller activity spikes. This suggests that threads that were non-rumours had high activity in a short time interval resulting in a clear peak, compared to the rumour threads which show multiple but lesser peaks.Insights that can be researched further such as come to mind; rumour threads occur multiple times throughout the course of an event and produce a fair amount of activity in a spacious time window, while non-rumour threads tend to have much higher activity but are centralized in a much denser time window.
When it comes to the the time from first reaction to the peak, we look at the text information below the plots. The rumour plots take a much shorter time to reach their peak in comparison to non-rumour plots, as they reach their peak in a span of days. This leads to an insight which can be confirmed with further analysis; rumour sources and reactions are seen most intensely in the start of an event across the whole network and produce a fair amount of activity. This can be tied into how debunks and credible sources eventually appear, not too long into the event, which leads to a central source, which is seen as the most credible and then this source turns out to be proven clearly as the ground truth.

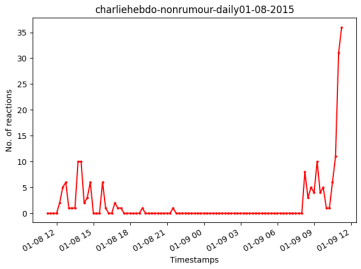
3.3 Daily Rumour and Non-rumour Reactions

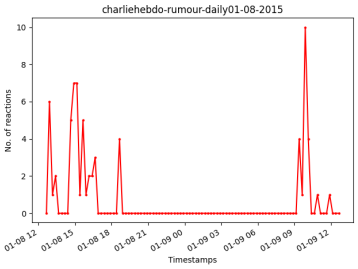
In these plots, all reactions are plotted in an interval but for each day. In total, there are too many plots generated to be able to include in the span of this report. rather than display and compare all plots achieved by the code, we focus only on the initial plots from `charliehebdo` event and show a general approach to how daily plots can prove insights in our analysis.

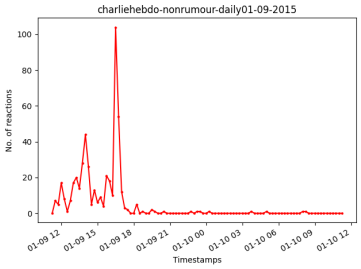


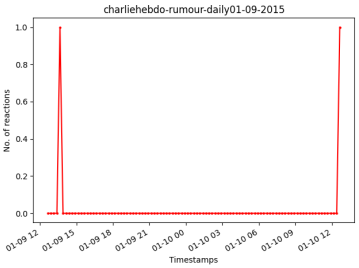


We should consider these daily plots as a longitudinal view which has been narrowed down into days, so we are looking for trends based on the hour of the 24 hour windows. From these initial 3 days of rumour and non-rumour plots, we see how the activity is mostly around 12:00 to 15:00. This suggests that across all threads, regardless of misinformation, activity took place in the afternoon in the Charlie Hebdo event. On 01-08-2015, we see how there is an overlap in activity spikes around 9:00 in the morning, possibly due to a node with high centrality activity or a spread of misinformation or truth.









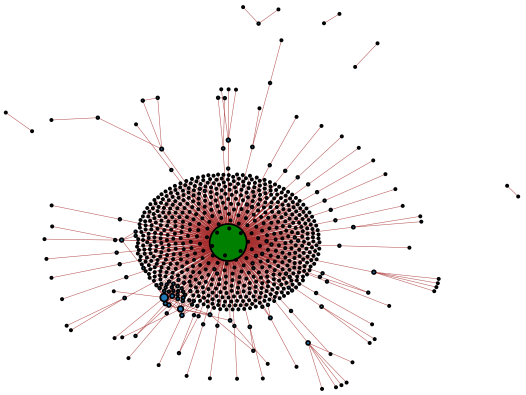
In general, the `charliehebdo` events activity took place in the afternoon around 12:00 to 15:00 in majority, local to the events time. These plots can be used in exploring what time most users contributing to misinformation are active, but further exploration is needed, such as being able to further plot and see who contributed to these reaction activities in the given hours of day.
Ego graphs in this section are modified such that the sizes of the nodes are directly mapped to their centrality. Thus, we will see nodes with high centrality as bigger and more visible. Also, the source nodes are colored green.

3.4 Top 3 Threads by Following

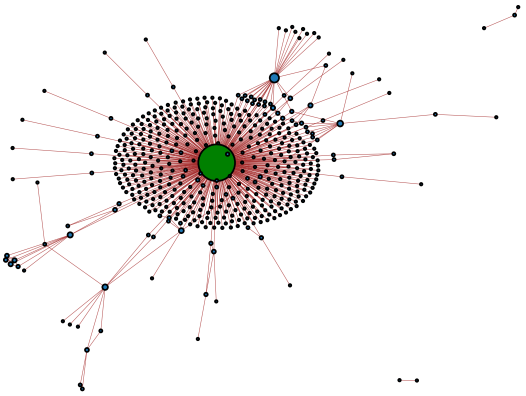
In this section, we take a look at the top 3 threads of the `charliehebdo` event based on the amount of reactions within the thread who follow the source user. In other words, for each thread, the amount of reactions within that thread who's user who actually follows that threads source tweet's user. Then, we select the top 3 from this sorted threads.

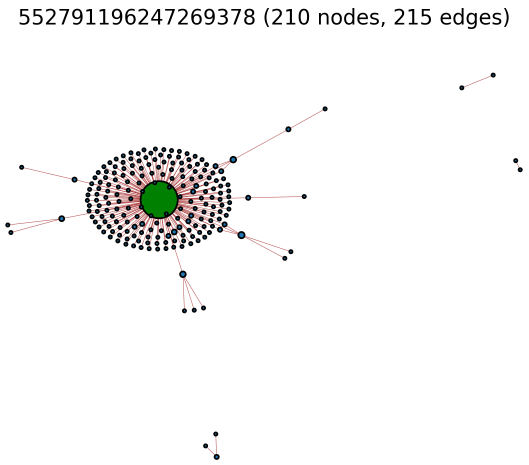
3.4.1 Ego graphs

552805488631758849 (761 nodes, 824 edges)



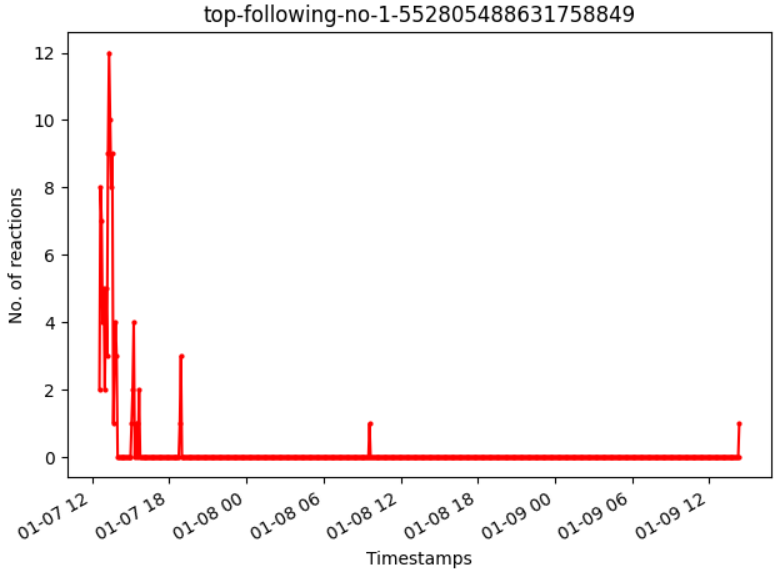
553506608203169792 (496 nodes, 555 edges)

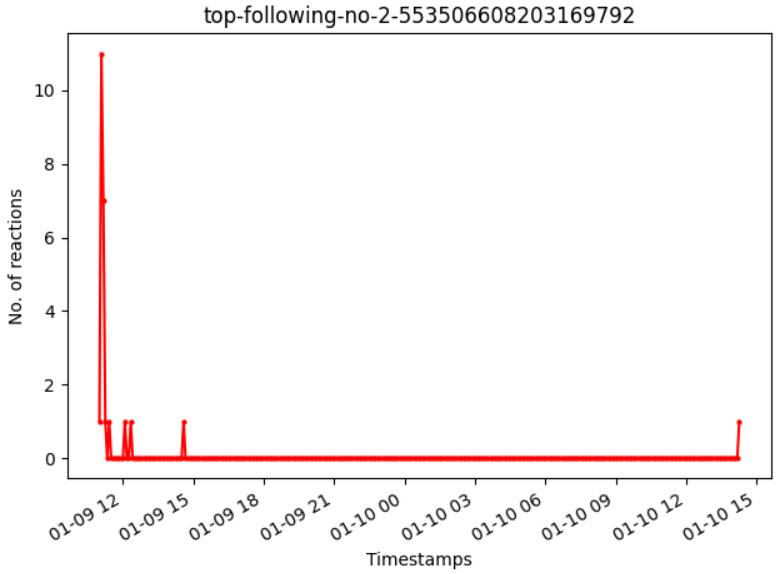


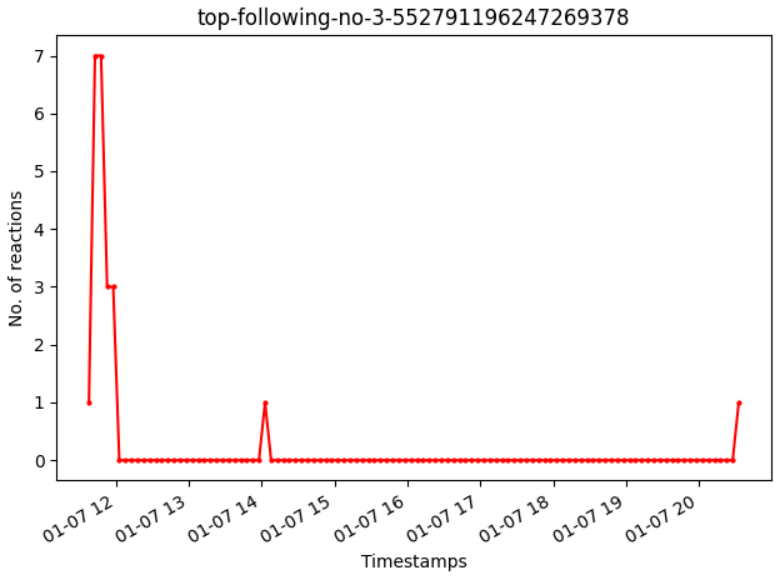


We see how all of the ego graphs have clearly defined clusters and there is not much intertwining of edges or groups. As expected, we see how the source node is the biggest and central node in all graphs. Something that was not expected certainly is how there seems to be no non-source nodes with high centrality. We can thus argue that for threads where the reactions tend to follow the source more, we expect the source to have the one and only high centrality.

3.4.2 Interval plots





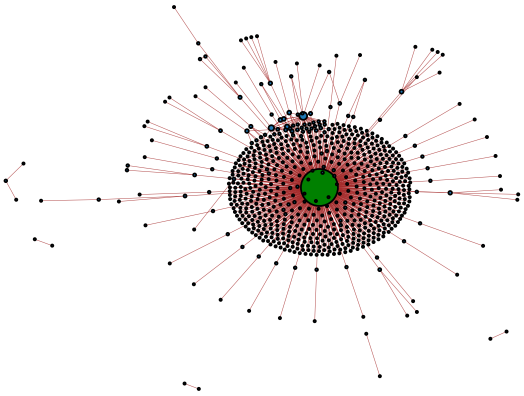


3.5 Top 3 Threads by Reaction count

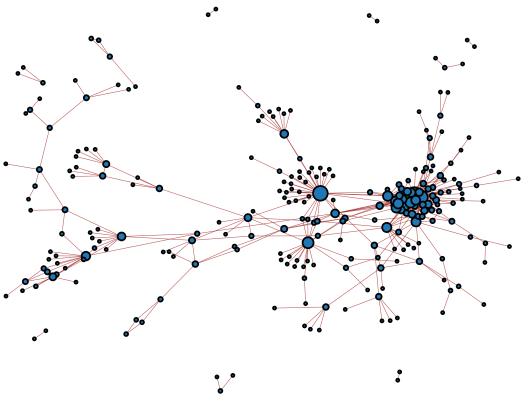
In this section, we take a look at the top 3 threads of the `charliehebdo` event based on the amount of reactions, without taking into consideration whether or not the reacting user follows the source user. In other words, for each thread, we rank based on the total amount of reactions. Then, we select the top 3 from these sorted threads.

3.5.1 Ego graphs

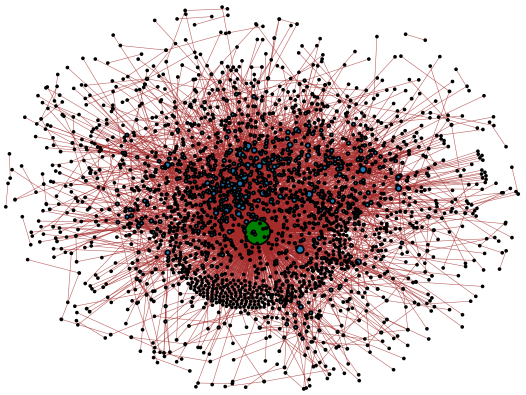
552805488631758849 (761 nodes, 824 edges)



552806309540528128 (258 nodes, 451 edges)

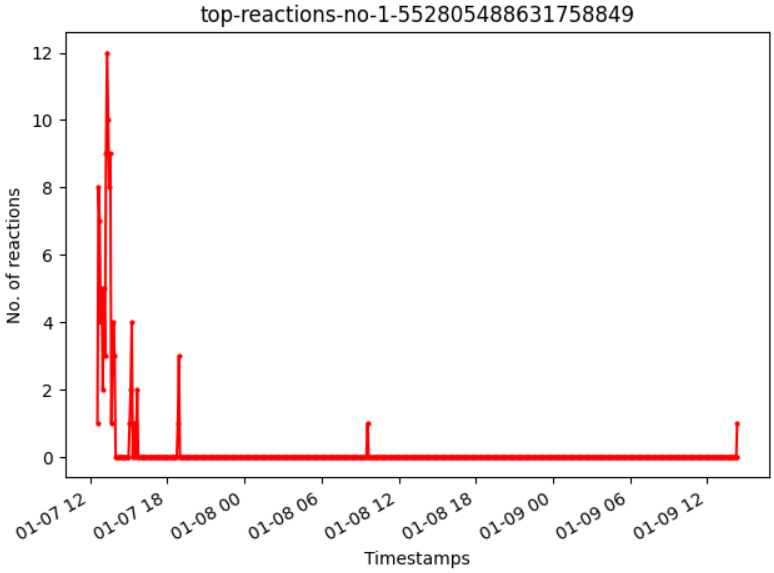


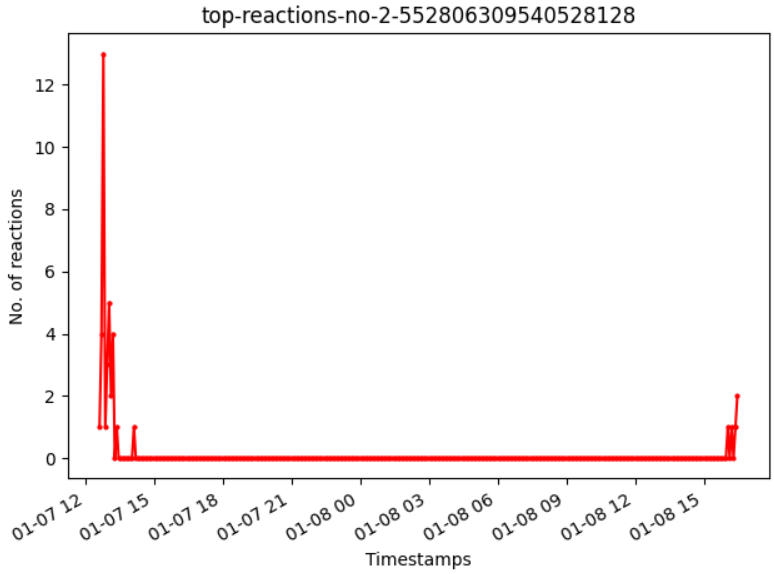
553587013409325058 (2322 nodes, 4806 edges)

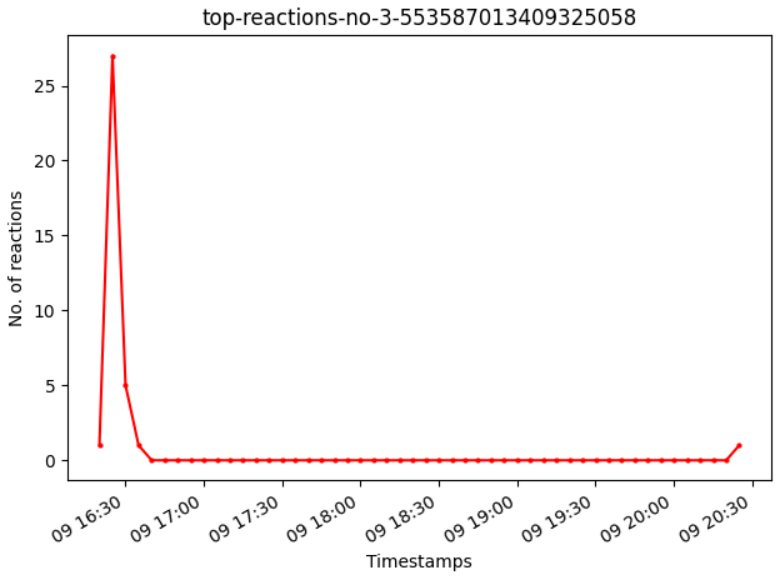


From the ego graphs, we see that the top ranking thread is the same top ranking thread based on the reactions that follow the source. The second thread in this ranking, sows multiple nodes other than the source with high centrality. The third thread is very chaotic and hard to interpret due to the amount of nodes and edges. There seems to be a lot of nodes with a little higher centrality compared to the majority of the non-source nodes. the source node is clearly seen with the highest centrality in the middle of the graph.

3.5.2 Interval plots







4 Conclusion

In this project, using Python we have achieved the visualizations and values mentioned throughout this report. By making use of different kinds of visualizations with the help of `matplotlib` and `networkx`, we have constructed an approach which allows a perspective into network data and extract insights, which overall makes it easier to formulate hypothesis on the data.
We only considered three events from the PHEME dataset, and at times only took a look at `charliehebdo` event's data to be able to fit the analysis into this report's span. Through accumulative and non-accumulative interval plots, we saw how reactions in events spread or peak throughout time. Accumulative plots show a steep rise in reactions right before the peak is reached, rather than a steady increase towards it. Interval plots showed that threads with rumour (misinformation) sources tended to take a much shorter time to reach their highest peak in reaction activity, and how non-rumour threads tended to have less peaks overall until the thread eventually halted all activity. Daily plots, which show the reaction activity for an events threads, showed how there is a pattern of activity occurring mostly around 12:00 to 15:00 for `charliehebdo`. For other events, looking and comparing all days would lead to the same insights. To finalize, we look into the top threads, sorted by the amount of reactions and by the amount of reactees who follow the source user on the network. The ego graphs showed that, as expected, when the reactions tend to follow the source node, the ego graph neatly displays the group around the source node. There also seems to be no non-source nodes with higher centrality. As for the other top threads, which is ranked on the total amount of reactions only, we see dif-

ferent patterns in all graphs. One common insight was that compared to the other ranking, we see how there are many more non-source nodes with a high centrality.
All in all, even though there are drawbacks in this analysis, such as the output format not being feasible to interpret due to size and the dataset focusing more on labelling rather than coverage of more data on Twitter, we can say that we have achieved some plausible insights into the dataset and the network within it. The code works for all events and generates extensive visualizations and text files with values, and is aimed to work for similarly organized datasets which are in the context of networks and activity over time. This project can be further improved by testing the approach on other datasets, comparing all of the output on all events. The reading in and preparing of the threads is clearly documented and works without any problems. This could prove beneficial in case someone also wants to work on the dataset in a similar way using Python.

5 Acknowledgements

I would like to thank my supervisor Mirela Riveni, for her support, guidance and knowledge throughout the project.

6 References

[1] University of Warwick. PHEME rumour dataset: support, certainty and evidentiality. June 2016. https://www.pheme.eu/2016/06/13/pheme-rumour-dataset-support-certainty-and-evidentiality/