CSE 472 Project 1 Report

Myles Hobbs
CSE 472
October 14, 2024

Topic Description
A key characteristic of the turn of the century, particularly among Generation Z, has been a heightened focus on social and environmental awareness. This has given rise to significant social movements that have shaped and pushed for a more tolerant and just society. One such movement is #FreePalestine, which advocates for the liberation of the Palestinian people from Israeli occupation. The frequent bombing of major cities like Gaza has led to the deaths of thousands, and the displacement of Palestinian citizens has stripped them of critical resources needed for survival.

What we are witnessing has been described by some as the genocide of the Palestinian people, along with numerous human rights violations committed on Israel's behalf. These events are being broadcast globally through social media, amplifying the movement. However, this issue remains deeply controversial, as Israel claims that the land historically belongs to them based on religious texts and historical ties. Israel justifies its actions, including military violence, citing the threat posed by militant organizations like Hamas, which is considered a terrorist group by many countries.

Despite this, Palestinians have lived in the region for centuries, and the modern conflict intensified after the establishment of Israel following World War II. The #FreePalestine movement symbolizes the push for the recognition of Palestine as an independent nation and seeks to hold Israel accountable for what many see as unjust actions. This movement aims to bring an end to the violence and to secure the rights and sovereignty of the Palestinian people.

Data Crawling Methods
I experimented the most with this step to obtain the desired results. Initially, it seemed logical to integrate the search API into my Python file when obtaining my API key. I expected it to return results or posts based on any search query, similar to how the search function works in the Mastodon app. However, this was far from the case. Even after paginating through results, removing parameters to retrieve more posts, and adding additional hashtags, I was still unable to collect up to 600 posts.

My initial approach was to paginate through the results for each hashtag (I had six at the time) until there were no more posts for that hashtag, and then move on to the next one. However, this method only returned about 300+ posts in total across all six hashtags. That's when I realized I needed a more effective approach. I could have continued adding more hashtags related to my topic, but the results were too inconsistent to rely on this method. I knew that searching Mastodon directly for #freePalestine would return thousands of posts, but the search API wasn't pulling those results, indicating a discrepancy. This ultimately led me to change the API I was using.

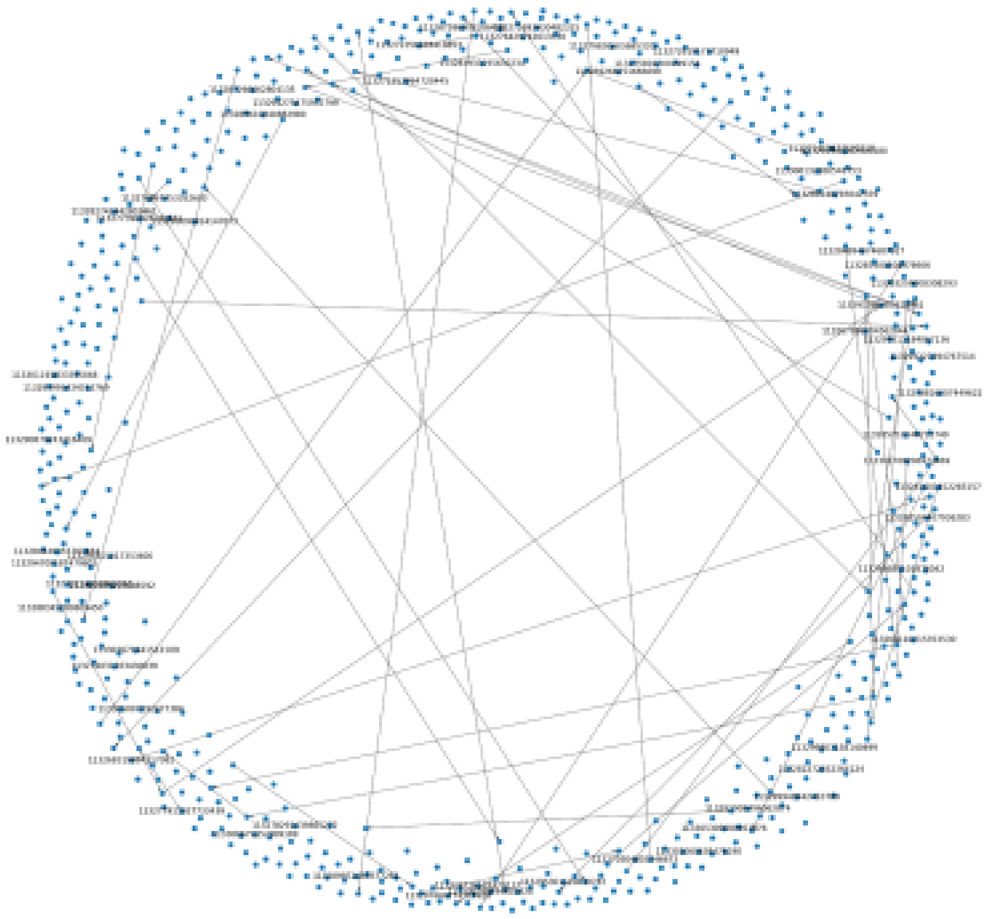
After researching other APIs, I concluded that the Timelines API was the best fit for my use case. By using the /tag query, I could fetch all posts with the #freePalestine tag. This API effectively returns a timeline of all posts containing the specified tag, which in my case, was more than sufficient. Compared to the search API, the Timelines API returned the data in a structured format, which was extremely helpful. Each post's attributes included the content, post ID, author, and much more. I was able to paginate through the results and collect 4,000 posts! As I retrieved each post, I quickly stored its attributes in a .json file, which became my main data source for the functions I defined in later steps. After some troubleshooting, implementing the Timelines API allowed me to move on to the next steps, where I could actually interpret and analyze the data.

Seed User Collection & Network Construction
After collecting all the necessary data, it was important to identify the correct seed users for analysis. I was initially unsure which criteria to use for selecting seed users. Should I focus on the user with the most followers, the user with the most liked post, the user who posted the most under the hashtag, or the user with the most interactions and replies? Ultimately, I decided to use a combination of these criteria. I selected the top 5 users who received the most replies, the top 5 users with the most likes, and the top 5 users who posted the most under the hashtag. I then combined these users into a set to eliminate any duplicates.

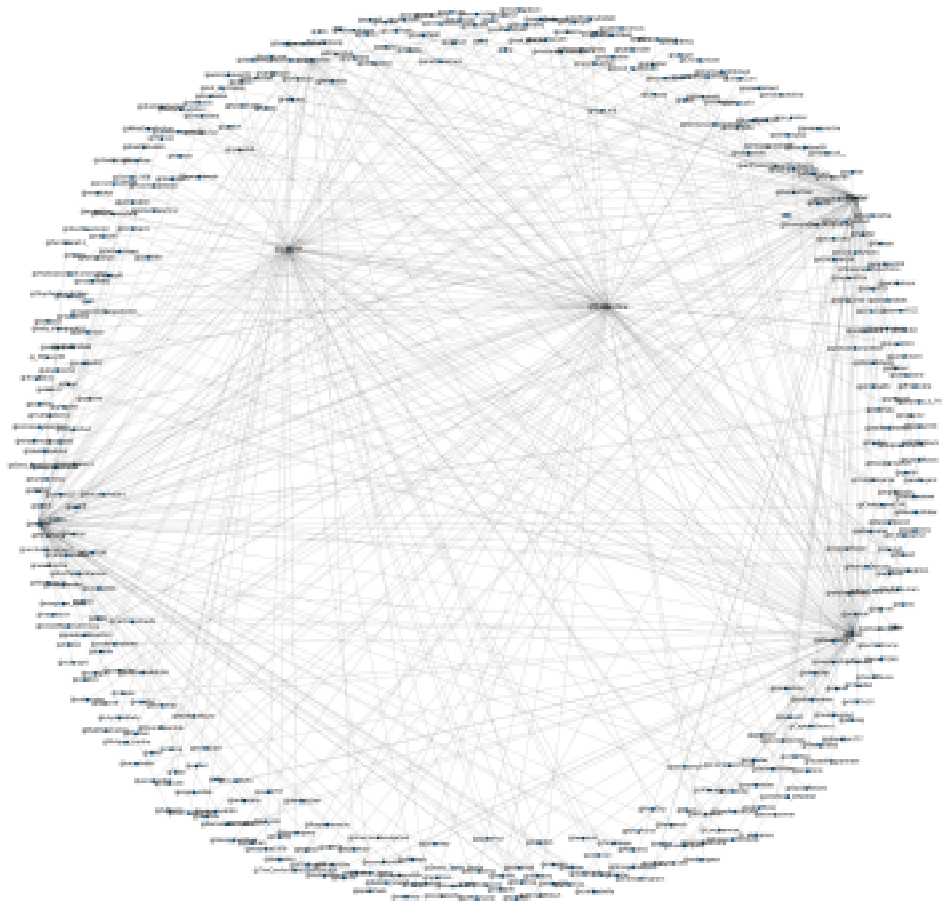
To achieve this, I parsed through each row in the .json file, counting the number of posts for each user, as well as tallying their reply and like counts. The next step was to determine how many followers each of my seed users had, which was crucial for constructing the Friendship Network. I used the Accounts API to loop through each seed user and retrieve their follower data, which I then attached to the mastodon_user dataset for use in creating the networks.

To construct the graphs, I used NetworkX. For the Information Diffusion Network, I went through each post in the mastodon_post.json file and added each post as a node. The caveat was that if a post was a reply to another post in the dataset, I created an edge between the original post and the reply. This approach led to one major issue: the graph became very cluttered and unreadable. My initial solution was to use Matplotlib to spread the graph out and improve readability. Some adjustments I made included decreasing the edge width, reducing node size, lightening the edges, labeling every 10th node, and decreasing the text size. While this helped a little, there were over 4,000 posts in the mastodon_post.json file, so I set a 600-post limit on the number of nodes displayed. This significantly reduced the clutter and made the graph more manageable.
Constructing the Friendship Network graph was much easier. I made each seed user a node and added their followers as additional nodes, connecting them with edges. Since this was an undirected graph, edge direction did not matter, but I wanted to emphasize the connections between these accounts. Pictures of both graphs are shown below.

Information Diffusion Network (Link attached for Zooming In Capabilities)



Friendship Network (Link attached for Zooming In Capabilities)



Classifying Toxic Nodes

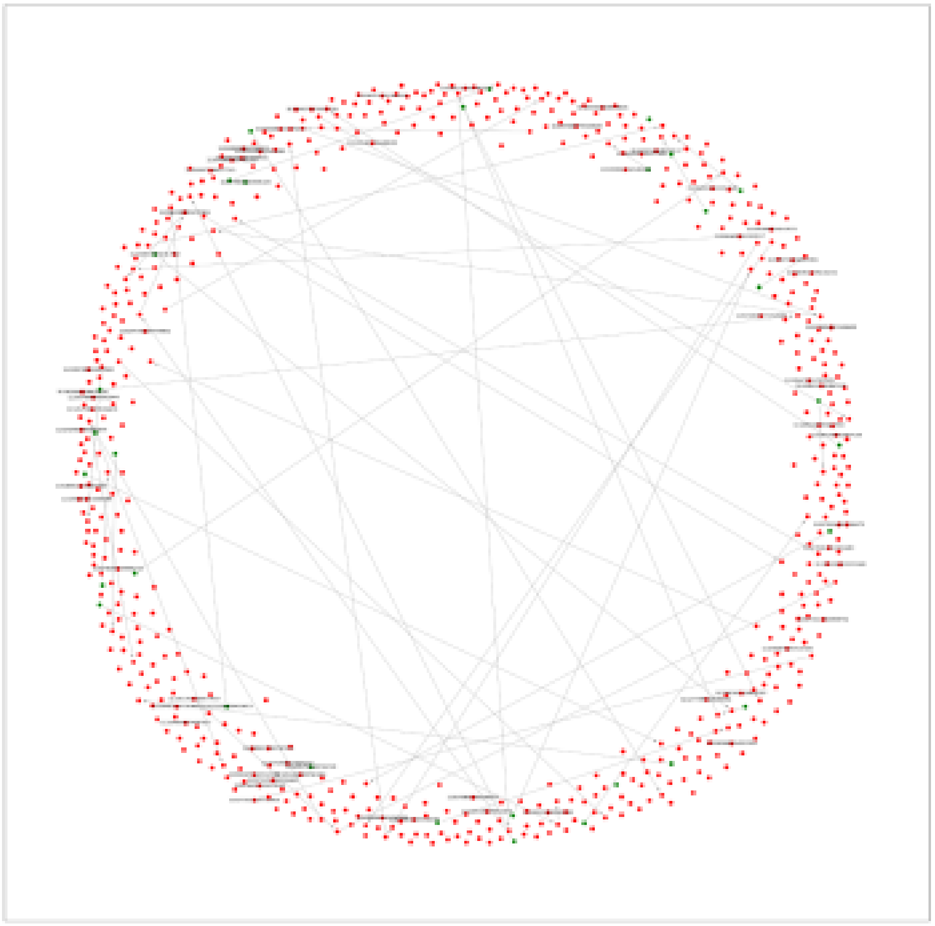
When I began this step, I encountered some initial confusion. I researched the use of Llama3-8B and Alpaca-7B models and tested them in a Colab notebook. They seemed like useful resources, given their instruction-input-response format. However, I was unsure how to integrate them into my solution due to the complexity of the installation process. Initially, I tried to append my own Python code to the Colab notebook, but this approach felt unorganized and inefficient.

I then conducted further research to find a more efficient and structured way to classify whether text (posts) are toxic or not. That's when I discovered the Hugging Face Transformers library, particularly the Toxic BERT model. This ultimately became the solution I integrated into my Python project.

The Transformers library provides access to various Natural Language Processing (NLP) models, including the Toxic BERT model. Some background on BERT: this model is fine-tuned to detect toxic language in text. It has been trained on data containing toxic, offensive, or harmful language, and its task is to classify whether a piece of text contains toxic content. When provided with text, the model returns a label of either "Toxic" or "Non-Toxic," along with a confidence score.

Now that some background on the BERT model has been given, I'll explain how I integrated it into my Python project. For each node in the Information Diffusion Network, I took the post_id and cross-referenced it with the mastodon_post.json file to extract the post text. I then passed the text into the BERT model, which assigned a label to each post based on its toxicity. If the post was labeled as toxic, I changed the node's color in the graph to red. Non-toxic posts were displayed as green nodes.

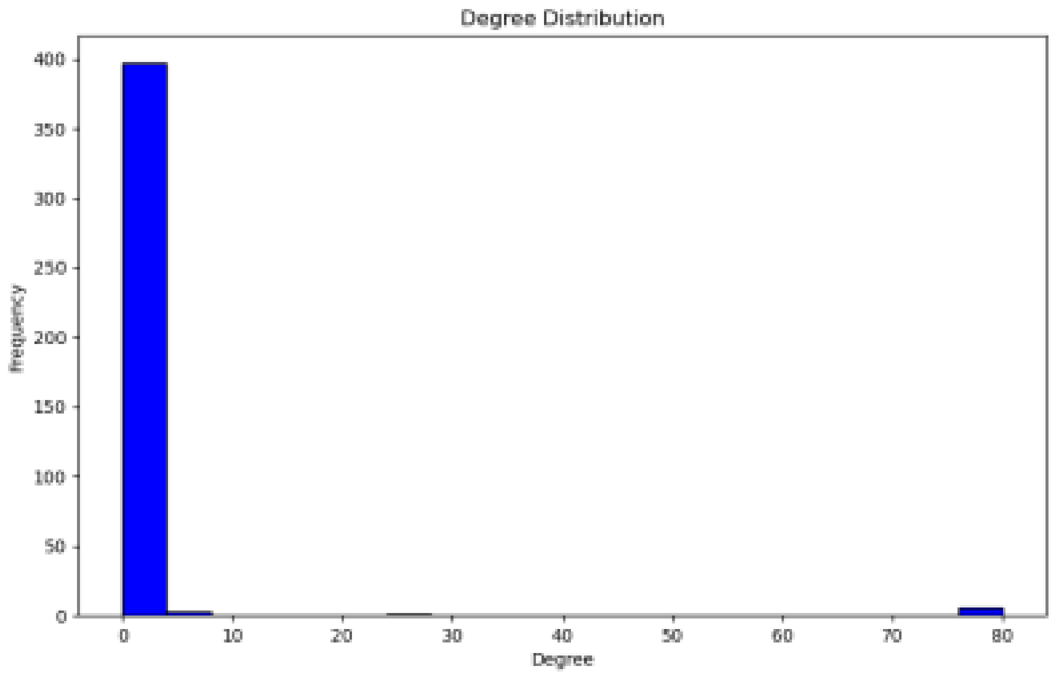
One criticism I have to be mindful of is potential bias in the BERT model, particularly since the topic of Free Palestine is highly controversial. It is possible that some posts were misclassified due to biases present in the training data. Nevertheless, I found the Hugging Face Transformers



library and the BERT model to be extremely useful in detecting whether or not these posts were toxic. A picture of the classified graph is shown below.

Classified Graph

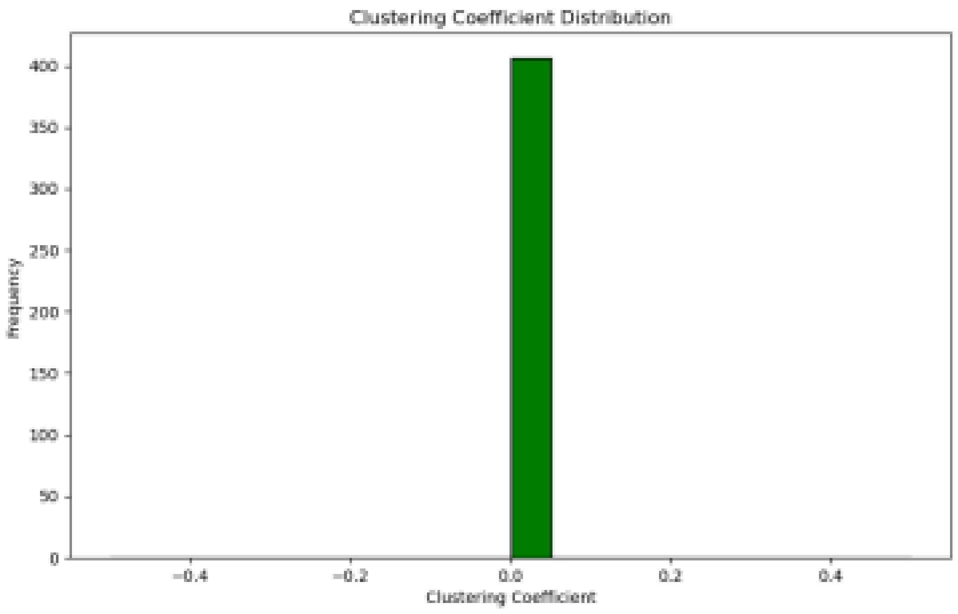
Degree Distribution Graph



Analysis

When constructing these graphs, I used the Friendship Network to analyze how many connections accounts have with each other. The graph shows that many users have very few connections, with the exception of some outliers who have 75+ connections. This subset likely represents the most influential users in our dataset, as the majority of users have little to no connections, indicating their lack of influence.

Clustering Coefficient Plot



Analysis

I also used the constructed Friendship Network to analyze clustering coefficients. This graph is intended to represent how interconnected users are within the Free Palestine community. Almost all nodes have a clustering coefficient close to 0, which means there is very little clustering or interconnectedness among the neighbors of each node. In contrast to the Degree Distribution graph and the Friendship Network graph, this suggests that there may be significant isolation between these user accounts.

References

- Search API: https://docs.joinmastodon.org/methods/search/
- Timeline API: https://docs.joinmastodon.org/methods/timelines/#tag
- BERT Model: https://huggingface.co/docs/transformers/en/model_doc/bert
    - Tutorial: https://www.youtube.com/watch?v=drdOS0QX2p4&ab_channel=AbhishekThakur
- Libraries Used
    - html2text
    - matplotlib
    - networkx
    - request
    - collections
    - csv
- Code was created with the help **Chat GBT** with the purpose of helping debug and correct errors throughout code