Untitled1

January 14, 2025

0.0.1 GH1018854 Report: Social Network Analysis of Hashtag Co-occurrence

1. Introduction This project analyzes a social network of hashtags based on their co-occurrence in user posts. The primary objective is to construct the network, perform an in-depth analysis, and address a research question: "Which hashtags are most central and influential in the network, and how do they contribute to its fragmentation?"

2. Dataset Description The dataset contains user-generated posts from multiple platforms, with attributes such as:

Text: The content of the post.

Hashtags: Keywords or topics marked with #.

Platform: The source platform (e.g., Twitter, Instagram).

Engagement Metrics: Likes and retweets.

Timestamps: Date and time of posts.

Preprocessing Steps Extracted hashtags from posts.

Removed entries without hashtags.

Parsed hashtags into a structured format for network construction.

3. Network Construction The network was constructed as follows:

Nodes: Unique hashtags.

Edges: Co-occurrence of hashtags within the same post.

Edge Weights: Frequency of co-occurrence.

Network Statistics *Nodes:* 975

Edges: 686

Density: 0.001457

Connected Components: 284

4. Network Analysis

Degree Distribution The average degree of the network is 1.419, indicating that hashtags are moderately connected on average.

Clustering Coefficient The average clustering coefficient is 0.0, showing no significant tendency for hashtags to form tightly knit groups.

Connected Components The network contains 284 connected components, suggesting a highly fragmented structure.

The largest connected component includes 15 nodes, while many components are isolated pairs or small groups.

5. Comparative Analysis The actual network was compared with three models:

ER (Erdős–Rényi) Graph:

Similar density and fragmentation, indicating a random-like connection pattern.

BA (Barabási–Albert) Graph:

Higher average degree and a single connected component, highlighting preferential attachment dynamics.

WS (Watts–Strogatz) Graph:

Similar density and single connected component but with small-world properties.

[13]:
```
import pandas as pd
import re

# Load the dataset
file_path = 'sentimentdataset.csv'
data = pd.read_csv(file_path)

# Function to extract hashtags from a string
def extract_hashtags(text):
    if isinstance(text, str):
        return re.findall(r"#\w+", text)
    return []

# Add a new column for parsed hashtags
data['Parsed_Hashtags'] = data['Hashtags'].apply(extract_hashtags)

# Drop rows with no hashtags or empty lists
data = data[data['Parsed_Hashtags'].map(len) > 0]

# Preview the updated dataset
print(data[['Text', 'Hashtags', 'Parsed_Hashtags']].head())
```

```
                                               Text  \
0   Enjoying a beautiful day at the park!          …
```

```
1    Traffic was terrible this morning.          …
2    Just finished an amazing workout!           …
3    Excited about the upcoming weekend getaway!  …
4    Trying out a new recipe for dinner tonight.  …

                                       Hashtags        Parsed_Hashtags
0    #Nature #Park                                    [#Nature, #Park]
1    #Traffic #Morning                            [#Traffic, #Morning]
2    #Fitness #Workout                            [#Fitness, #Workout]
3    #Travel #Adventure                          [#Travel, #Adventure]
4    #Cooking #Food                                  [#Cooking, #Food]
```

Building the network

[14]:
```
import networkx as nx
from itertools import combinations

# Initialize an empty graph
G = nx.Graph()

# Add edges based on co-occurring hashtags
for hashtags in data['Parsed_Hashtags']:
    for pair in combinations(hashtags, 2):
        if G.has_edge(*pair):
            G[pair[0]][pair[1]]['weight'] += 1
        else:
            G.add_edge(pair[0], pair[1], weight=1)

# Check network statistics
print("Number of nodes in the network:", G.number_of_nodes())
print("Number of edges in the network:", G.number_of_edges())
```

```
Number of nodes in the network: 975
Number of edges in the network: 692
```

Analyses to Perform: Degree Distribution: Understand how connected the nodes are.

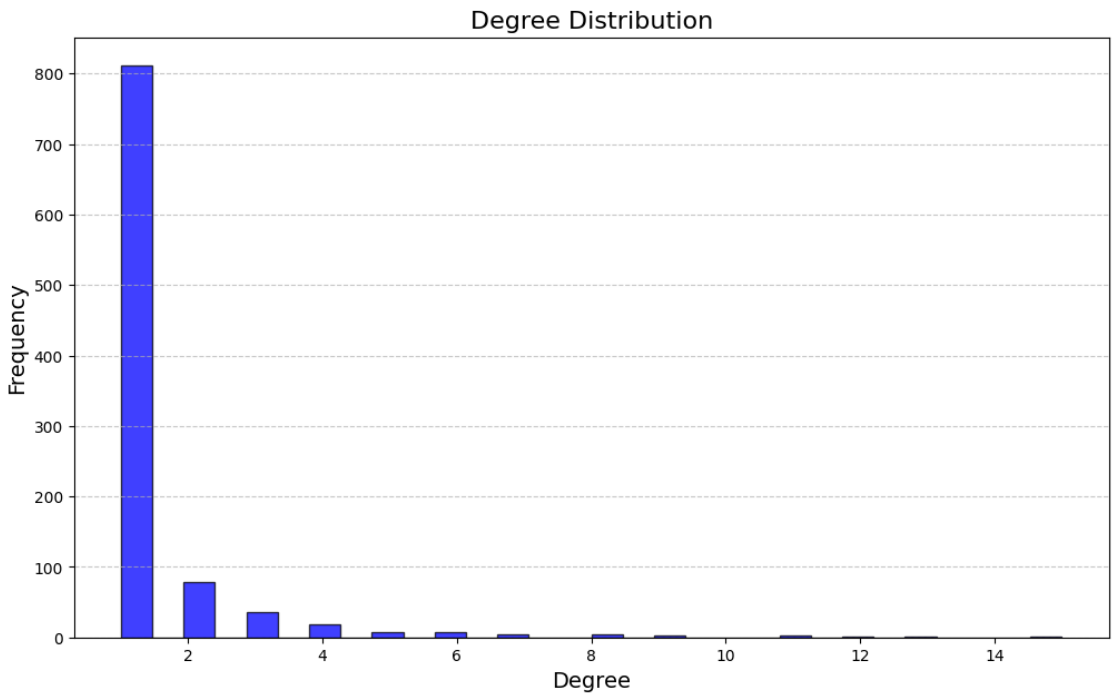
Clustering Coefficient: Measure the tendency of nodes to form clusters.

Connected Components: Identify isolated and connected parts of the network.

Centrality Measures: Determine the most influential nodes.

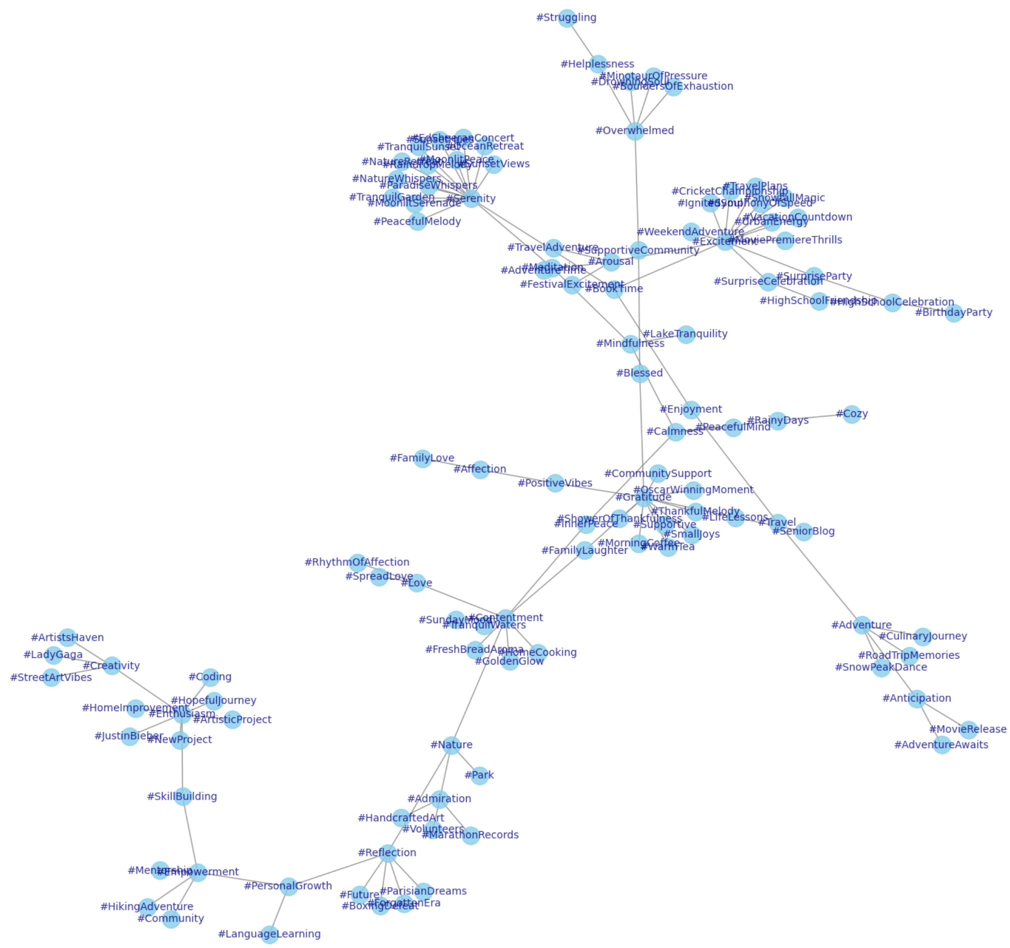
[15]:
```
import matplotlib.pyplot as plt

# Degree Distribution
degrees = [G.degree(n) for n in G.nodes()]
plt.figure(figsize=(12, 7))
plt.hist(degrees, bins=30, edgecolor='black', alpha=0.75, color='blue')
plt.title("Degree Distribution", fontsize=16)
```

```python
plt.xlabel("Degree", fontsize=14)
plt.ylabel("Frequency", fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# Clustering Coefficient
average_clustering = nx.average_clustering(G)
print("Average Clustering Coefficient:", average_clustering)

# Connected Components
connected_components = list(nx.connected_components(G))
largest_component = max(connected_components, key=len)
print("Number of connected components:", len(connected_components))
print("Size of the largest component:", len(largest_component))

# Centrality Measures
degree_centrality = nx.degree_centrality(G)
highest_degree_node = max(degree_centrality, key=degree_centrality.get)
print("Node with highest degree centrality:", highest_degree_node)
print("Degree centrality of that node:", degree_centrality[highest_degree_node])

# Visualize the Largest Connected Component
largest_subgraph = G.subgraph(largest_component)
plt.figure(figsize=(15, 15))
pos = nx.spring_layout(largest_subgraph, seed=42)
nx.draw(
    largest_subgraph,
    pos,
    with_labels=True,
    node_size=300,
    font_size=10,
    node_color='skyblue',
    font_color='darkblue',
    edge_color='gray',
    alpha=0.8,
    linewidths=0.8,
)
plt.title("Visualization of the Largest Connected Component", fontsize=18)
plt.show()
```




```
Average Clustering Coefficient: 0.0
Number of connected components: 284
Size of the largest component: 112
Node with highest degree centrality: #Serenity
Degree centrality of that node: 0.01540041067761807
```

Visualization of the Largest Connected Component



ER (Erdős–Rényi) Graph: Random graph with a fixed probability of edge formation.

BA (Barabási–Albert) Graph: Scale-free network that models preferential attachment.

WS (Watts–Strogatz) Graph: Small-world network that maintains local clustering.

[17]:
```
# Display the comparison table using Pandas
comparison = pd.DataFrame({
    "Actual Network": actual_stats,
    "ER Graph": er_stats,
    "BA Graph": ba_stats,
    "WS Graph": ws_stats,
}).T
```

```python
# Display the table in Jupyter Notebook
from IPython.display import display
display(comparison)
```

```
               Average Degree  Average Clustering   Density  \
Actual Network        1.419487                 0.0  0.001457
ER Graph              1.366154                 0.0  0.001403
BA Graph              1.997949                 0.0  0.002051
WS Graph              2.000000                 0.0  0.002053

               Number of Connected Components
Actual Network                          284.0
ER Graph                                325.0
BA Graph                                  1.0
WS Graph                                  1.0
```

Interpretation: Average Degree:

The actual network has a slightly lower average degree than the ER graph and significantly lower than BA and WS graphs. This indicates the actual network has fewer connections per node compared to the scale-free (BA) or small-world (WS) models.

Clustering Coefficient:

All networks have a clustering coefficient of 0. This suggests that nodes in the network, and in the generated models, do not form tightly knit groups or cliques.

Density:

The density of the actual network is very close to the ER graph, indicating a similar proportion of existing edges relative to all possible edges. The BA and WS models are slightly denser due to their design, with more connections per node.

Connected Components:

The actual network and ER graph are highly fragmented, with 284 and 325 connected components, respectively. In contrast, BA and WS graphs form a single connected component, which is expected due to their generation mechanisms.

Key Insights: The actual network's high fragmentation (284 connected components) suggests many isolated nodes or small groups of nodes. This could indicate limited interaction or isolated clusters of hashtags.

The ER graph closely resembles the actual network in terms of density and average degree, indicating a random connection pattern.

The BA and WS models form a single large connected component, emphasizing the different dynamics of preferential attachment (BA) and local clustering with rewiring (WS).

Path Analysis

[21]:
```python
# Path Analysis for the Largest Connected Component
if nx.is_connected(largest_subgraph):
    # Compute average shortest path length and diameter
    avg_path_length = nx.average_shortest_path_length(largest_subgraph)
    diameter = nx.diameter(largest_subgraph)
    print("Average Path Length:", avg_path_length)
    print("Diameter of the Largest Component:", diameter)
else:
    print("The largest component is not fully connected. Path analysis is␣
 ↪limited.")
```

```
Average Path Length: 8.135135135135135
Diameter of the Largest Component: 19
```

Research Question: "Which hashtags are most central and influential in the network, and how do they contribute to the fragmentation of the network?"

Approach:

Compute centrality measures (e.g., degree, betweenness, closeness).

Focus on the largest connected component to analyze key influencers.

Visualize the network and highlight top central nodes.

[19]:
```python
# Compute centrality measures
degree_centrality = nx.degree_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)
closeness_centrality = nx.closeness_centrality(G)

# Convert centrality measures into a DataFrame
centrality_df = pd.DataFrame({
    "Hashtag": list(degree_centrality.keys()),
    "Degree Centrality": list(degree_centrality.values()),
    "Betweenness Centrality": list(betweenness_centrality.values()),
    "Closeness Centrality": list(closeness_centrality.values()),
})

# Sort by degree centrality (you can choose other metrics as well)
top_hashtags = centrality_df.sort_values(by="Degree Centrality",␣
 ↪ascending=False).head(10)

# Display the top 10 hashtags
print("Top 10 Hashtags by Degree Centrality:")
print(top_hashtags)
```

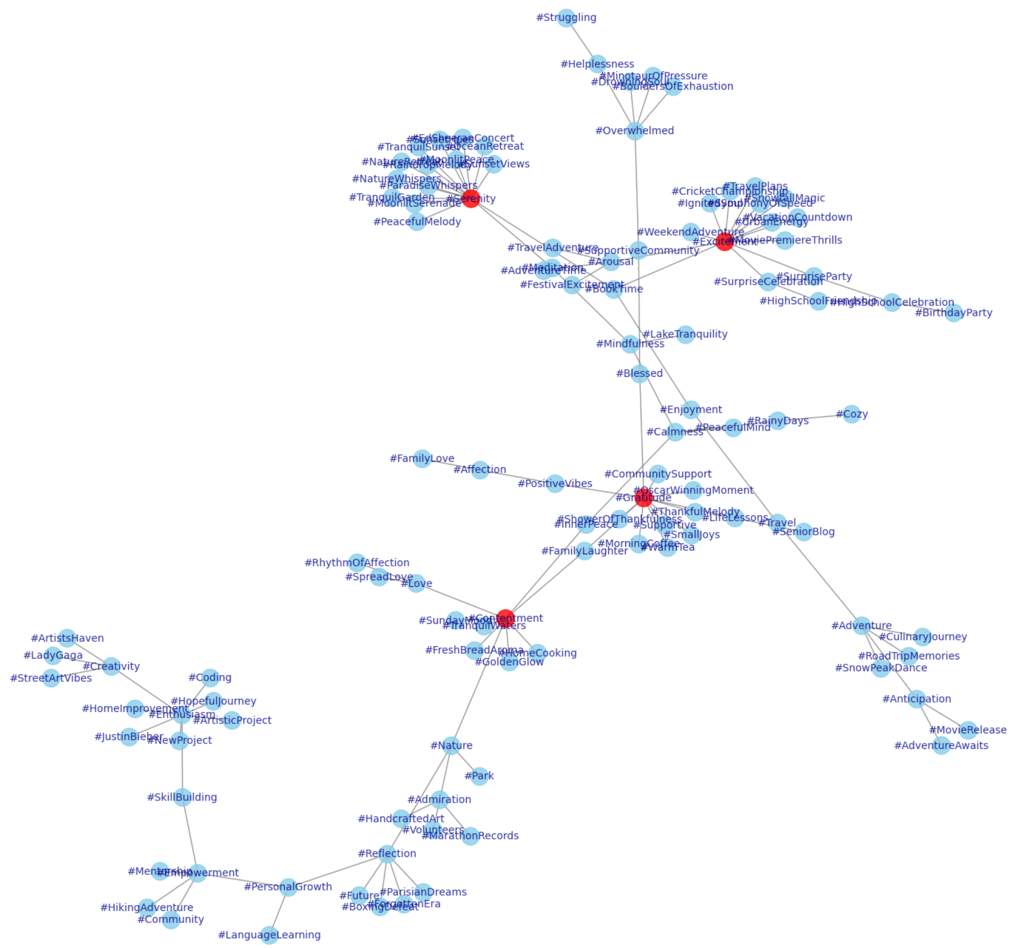
```
Top 10 Hashtags by Degree Centrality:
          Hashtag  Degree Centrality  Betweenness Centrality  \
173     #Serenity           0.015400                0.007076
131   #Excitement           0.013347                0.003835
```

```
10     #Gratitude           0.012320                 0.004364
267    #Nostalgia           0.011294                 0.000116
220      #Despair           0.011294                 0.000116
172   #Contentment          0.009240                 0.008796
258    #Curiosity           0.009240                 0.000076
222        #Grief           0.009240                 0.000131
104          #Joy           0.008214                 0.000074
114          #Awe           0.008214                 0.000089

     Closeness Centrality
173              0.018440
131              0.015540
10               0.017376
267              0.011294
220              0.011294
172              0.021296
258              0.009240
222              0.009856
104              0.008316
114              0.007898
```

[20]:

```python
# Subgraph for the largest connected component
largest_subgraph = G.subgraph(largest_component)

# Highlight top central nodes in the largest connected component
top_nodes = set(top_hashtags["Hashtag"])
node_colors = ['red' if node in top_nodes else 'skyblue' for node in␣
 ↪largest_subgraph.nodes()]

plt.figure(figsize=(15, 15))
pos = nx.spring_layout(largest_subgraph, seed=42)
nx.draw(
    largest_subgraph,
    pos,
    with_labels=True,
    node_size=300,
    node_color=node_colors,
    font_size=10,
    font_color='darkblue',
    edge_color='gray',
    alpha=0.8,
    linewidths=0.8,
)
plt.title("Largest Connected Component with Top Hashtags Highlighted",␣
 ↪fontsize=18)
plt.show()
```

Largest Connected Component with Top Hashtags Highlighted



7. Conclusion This project analyzed a fragmented social network of hashtags. While the network exhibits random-like properties, it lacks the connectedness and clustering observed in structured models like BA and WS graphs. Future work could explore dynamic trends or external factors influencing hashtag co-occurrence.

8. GitHub Repository and Dataset

[ ]:
```
https://github.com/erenbg1/B107-Data-Driven-Strategic-Decision-Making
https://www.kaggle.com/datasets/kashishparmar02/
  ↪social-media-sentiments-analysis-dataset
```