Celebrity Network Analysis using PageRank

Salim Dari

November 2023

1 Introduction

It is almost Christmas and a large celebrity party is about to be organized. To decide who will host the party, the celebrities decide to play a phone tag game. Each celebrity has an equal chance of calling someone whose number they have. There is also a chance (albeit small!) that a celebrity could use their assistants to randomly call anyone else in the celebrity group without interrupting the game. In this write-up, we aim to explore and understand the relationships between various celebrities using network analysis techniques. Specifically, we will set up a graph to visually represent these relationships, apply the PageRank algorithm to determine who would most likely be the host of the upcoming party, and discuss additional factors that could provide a more holistic view of the network's dynamics.

2 Description of mathematics

This project uses the PageRank algorithm to determine who is likely to host the party. PageRank is a way of measuring the importance of website pages (or, in our case, celebrities) based on how many links (or calls) point to them and the importance of those linking pages. Each web page (or celebrity) is initially assigned an importance value. This can be represented by a vector v.

The algorithm iteratively updates the importance of each page based on the importance of the pages linking to it. Mathematically, this is represented as $v_1 = Tv$, where T is a matrix representing the transition of the web (or the network of celebrity calls), and v is the importance vector. Subsequent iterations follow the pattern $v_2 = T^2v$, $v_3 = T^3v$, and so on.

The iterative process converges to an equilibrium value v^*, where v^* denotes the PageRank vector, representing the final importance or rank of each web page (or celebrity). The process continues until the importance values converge to a stable state. In this stable state, the importance of a page (or celebrity) reflects its probability of being visited or being "it". The key idea is that a page (or celebrity) is more important if it is pointed to by other important pages (or can call other important celebrities).

To help aid with visualization we can refer to the figure below that shows the connections between the 16 celebrities.

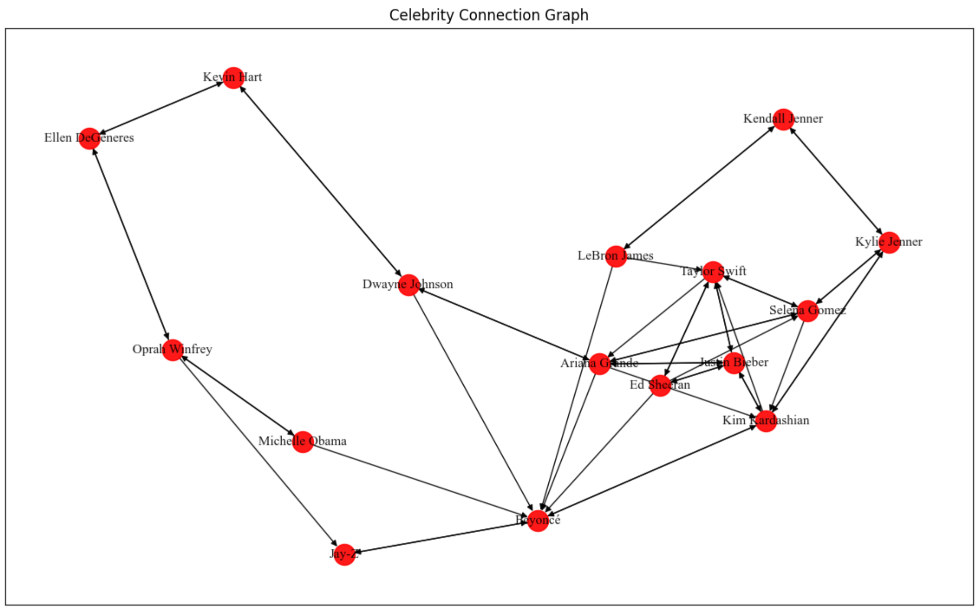

Figure 1: Directed Graph portraying the celebrities' connection

3 Results and Discussion

After we apply the PageRank algorithm to find the host we can find the following values that can help us predict the host of the party. We can refer to the table below for the PageRank values of each celebrity with different damping values. Let us take a deeper look at what each value in the table implies.

When p = 1, that is no calls are made by the assistants, Beyoncé and Kim Kardashian exhibit the highest PageRank values, with Kim K likely being the host. This indicates their central roles when the connections are solely considered. This suggests they have numerous direct connections, that is they can call a lot of people, a lot of people can call them, or both.

When p = 0, all celebrities have an equal PageRank value of 0.062. This scenario represents a completely random calling pattern, then all connections of who can call whom are irrelevant.

When p = 0.15, 0.25, 0.5, and 0.75, we notice that Taylor Swift, Selena Gomez, and Ariana Grande show relatively stable PageRank values. This indicates their consistent influence regardless of how much the network structure or randomness is emphasized. This likely indicates that they would be the host of the parties if there was a considerable amount of assistants making calls.

For this project however, we will assume that assistant participation is close to minimum and we will consider p = 0.15 as our damping factor. In this scenario, our likely host and the loser of the game is Kim K! This is because Kim has the most direct connections with celebrities

Celebrity	p = 0.15	p = 0	p = 1	p = 0.25	p = 0.75	p = 0.5
Taylor Swift	0.065	0.062	0.105	0.067	0.080	0.072
Ed Sheeran	0.058	0.062	0.050	0.055	0.044	0.048
Selena Gomez	0.063	0.062	0.080	0.063	0.065	0.063
Ariana Grande	0.063	0.062	0.077	0.064	0.067	0.065
Justin Bieber	0.062	0.062	0.097	0.062	0.072	0.064
Beyoncé	0.079	0.062	0.169	0.090	0.144	0.116
Kim Kardashian	0.069	0.062	0.170	0.074	0.122	0.093
Kylie Jenner	0.062	0.062	0.078	0.063	0.066	0.063
Dwayne Johnson	0.060	0.062	0.020	0.057	0.041	0.051
Jay-Z	0.062	0.062	0.086	0.063	0.080	0.069
Kevin Hart	0.061	0.062	0.010	0.059	0.041	0.053
Ellen DeGeneres	0.061	0.062	0.006	0.059	0.041	0.054
Oprah Winfrey	0.062	0.062	0.004	0.061	0.041	0.055
Michelle Obama	0.056	0.062	0.001	0.052	0.026	0.040
Kendall Jenner	0.059	0.062	0.031	0.057	0.040	0.049
LeBron James	0.058	0.062	0.016	0.054	0.030	0.044

Table 1: PageRank values for different values of p

4 Conclusion

In this study, we used the PageRank algorithm to analyze a network of celebrities engaged in a phone tag game to decide the host of a Christmas party. Our analysis revealed the likelihood of a celebrity being chosen to host the party as well as the effects of assistant intervention in the calling process. With none or minimal assistant intervention ($p = 0, 0.15$), Kim Kardashian stands out as the most likely host, reflecting her extensive direct connections within the celebrity network.

The project also examines scenarios with varying degrees of randomness in call patterns. In a completely random calling pattern ($p = 0$), all celebrities have equal chances of hosting, nullifying the impact of their specific connections. However, in situations with moderate to high assistant intervention ($p = 0.25, 0.5, 0.75$), Taylor Swift, Selena Gomez, and Ariana Grande exhibit relatively stable PageRank values. This suggests their consistent influence regardless of network structure or randomness, indicating their potential to host the party under these conditions.

Overall, this project demonstrates how network dynamics can be used to predict outcomes like the selection of a party host, depending on the level of external intervention or randomness in the network.

5 Code

Here is the code for creating the model, the code is straightforward and includes comments on implementations.

```python
import networkx as nx

# Create a directed graph
G = nx.DiGraph()

# Add edges (connections) to the graph (Edge,Edge)
connections = [
    ('Taylor Swift', 'Ed Sheeran'), ('Taylor Swift', 'Selena Gomez'),
    ('Taylor Swift', 'Ariana Grande'), ('Taylor Swift', 'Justin Bieber'),
    ('Ed Sheeran', 'Taylor Swift'), ('Ed Sheeran', 'Justin Bieber'),
    ('Ed Sheeran', 'Beyonc '), ('Ed Sheeran', 'Selena Gomez'),
    ('Selena Gomez', 'Taylor Swift'), ('Selena Gomez', 'Ariana Grande'),
    ('Selena Gomez', 'Kim Kardashian'), ('Selena Gomez', 'Kylie Jenner'),
    ('Ariana Grande', 'Selena Gomez'), ('Ariana Grande', 'Beyonc '),
    ('Ariana Grande', 'Justin Bieber'), ('Ariana Grande', 'Dwayne Johnson'),
    ('Ariana Grande', 'Kim Kardashian'), ('Justin Bieber', 'Ed Sheeran'),
    ('Justin Bieber', 'Ariana Grande'), ('Justin Bieber', 'Kim Kardashian'),
    ('Justin Bieber', 'Taylor Swift'), ('Beyonc ', 'Kim Kardashian'),
    ('Beyonc ', 'Jay-Z'), ('Jay-Z', 'Beyonc '),
    ('Kim Kardashian', 'Beyonc '), ('Kim Kardashian', 'Taylor Swift'),
    ('Kim Kardashian', 'Justin Bieber'), ('Kim Kardashian', 'Kylie Jenner'),
    ('Dwayne Johnson', 'Beyonc '), ('Dwayne Johnson', 'Ariana Grande'),
    ('Dwayne Johnson', 'Kevin Hart'), ('Kevin Hart', 'Dwayne Johnson'),
    ('Kevin Hart', 'Ellen DeGeneres'), ('Ellen DeGeneres', 'Kevin Hart'),
    ('Ellen DeGeneres', 'Oprah Winfrey'), ('Oprah Winfrey', 'Ellen DeGeneres'),
    ('Oprah Winfrey', 'Jay-Z'), ('Oprah Winfrey', 'Michelle Obama'),
    ('Michelle Obama', 'Oprah Winfrey'), ('Michelle Obama', 'Beyonc '),
    ('Kylie Jenner', 'Kim Kardashian'), ('Kylie Jenner', 'Kendall Jenner'),
    ('Kylie Jenner', 'Selena Gomez'), ('Kendall Jenner', 'Kylie Jenner'),
    ('Kendall Jenner', 'LeBron James'), ('LeBron James', 'Kendall Jenner'),
    ('LeBron James', 'Taylor Swift'), ('LeBron James', 'Beyonc ')
]

G.add_edges_from(connections) # G is the directed graph containing our celebrities

# Compute PageRank
pagerank = nx.pagerank(G, alpha=0)  # alpha is the damping factor p

# Sort the PageRank  in descending order
sorted_ranks = sorted(pagerank.items(), key=lambda item: item[1], reverse=True)

# Iterate over the sorted ranks and print them
for celebrity, rank in sorted_ranks:
    print(f"{celebrity}: {(rank):.3f}") # prints to 3 decimal places
```