**Nick Mankowski**
**Writeup 2**
**Due 22 November 2023**

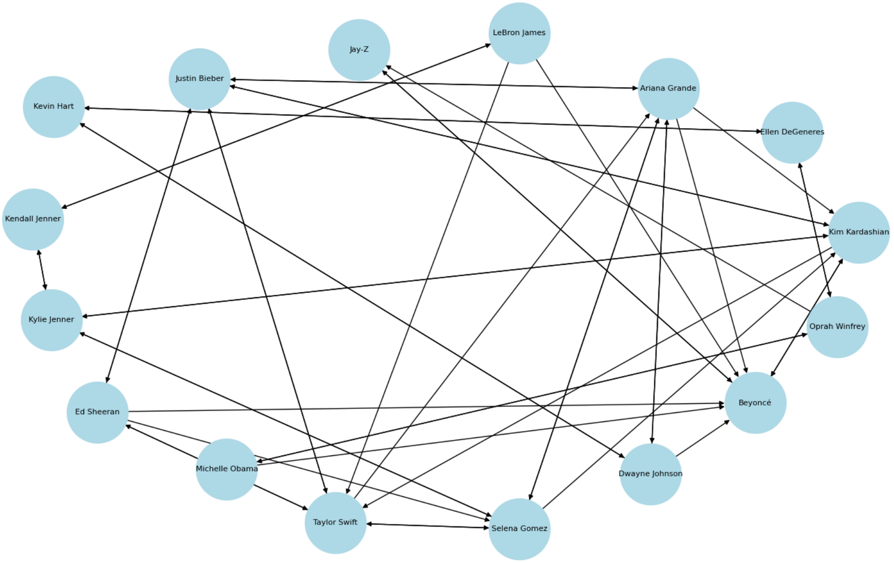
# Using PageRank to Model Phone Tag

## 1 Introduction

Ed Sheeran has been feeling awfully lonely in his day-to-day life of writing and performing songs. He feels as though he needs the child-like wonder that comes only from a game of tag in order to cure this loneliness. However, Ed has an awfully busy group of friends, so he cannot have them take time out of their busy schedules to meet him in order for a game of tag. As a form of compromise, Ed has decided to schedule a game of phone tag. This also gives the opportunity for the game to grow bigger, as Ed's friends can invite people he does not know.

(*) We can represent the players in this game of phone tag using the following directed graph. In the graph, an arrow directed towards an individual signifies that the person at the arrow's origin possesses the contact details of that individual.

Figure 1: Relations Between Celebrities



Ed Sheeran is an awfully competitive person, though, so he wants to be able to predict how low his odds are of being "it". In this paper, we will use the techinque of PageRank in order to analyze the results of a game of phone tag using the graph given above.

## 2 Description of Mathematics

### 2.1 Forming our Google Matrix

In order to perform our analysis on the graph displayed in 1, we must form an adjacency matrix indicating relationships between edges. We will assemble ~~this~~ one adjacency matrix with the following constraints:

- All celebrities are given a unique index value 1 through 16
- $A_{ij} = 1$ indicates that the celebrity with index $i$ has the contact information of the celebrity with index $j$.
- $A_{ij} = 0$ indicates that the celebrity with index $i$ does not have the contact information of the celebrity with index $j$.
- $A_{ii} = 0$ for all $i$.

If we have our data organized in a 2 dimensional vector called `connections`, we can form our adjacency matrix as follows:

```
% Initialize an empty adjacency matrix
n = 16;
adjMatrix = zeros(n, n);

% Populate the adjacency matrix
for i = 1:length(connections)
    adjMatrix(i, connections{i}) = 1;
end

```

**Note:** `connections(i)` is assumed to hold a list of celebrities whose contact information is available to celebrity $i$.

Furthermore, we can make the assumption that each celebrity has an equal likelihood to call each of their contacts in the game. Thus, we can modify our adjacency matrix so that each element $A_{ij}$ now represents the probability that the celebrity with index $i$ will call the celebrity with index $j$. We can perform this change as well as calculate the transpose of our matrix in order to produce the transition matrix for this particular graph. We can create our transtion matrix T as follows:

```
% Get our transition matrix
T = zeros(n, n);
for i = 1:length(adjMatrix)
    T(i, :) = adjMatrix(i, :) / sum(adjMatrix(i, :));
end
T = transpose(T);
```

Now that we have our transition matrix, we are able to form our Google matrix. Since all of our celebrities have assistants who have the contact information of everybody in the game, we must account that there is a small likelihood that they could call another celebrity in the game. This assumption has the effect of loosely coupling all elements of our graph. This allows us to ensure that there is no possibility of errors resulting in dead ends or disconnected components of our graph. We can create our Google matrix $M$ that accounts for this as follows:

mention this in intro

$$M = (1-p)T + pB, \quad \text{where } B = \frac{1}{n}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{1}$$

where ~~and~~ $p$ is some damping factor(typically around 0.15).
We are able to do this in MATLAB as follows:

Expand upon what damping factor means.

```
% Get the Google matrix
M = (1 - p) * T + (p / n) * ones(n, n);
```

## 2.2 Calculating PageRank

Now that we have our Google matrix $M$, we are able to calculate the PageRank and look at probabilistic outcomes of our game of phone tag. In order to do this, we must first produce a vector of initial possiblities. This vector, $\overrightarrow{v}$, will be a 1x$n$ matrix where each element is equal to $\frac{1}{n}$. That is,

$$\overrightarrow{v} = \frac{1}{n} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}.$$

This is representative of the fact that each celebrity has an equal probability of being the first person to be ”it” in the game of phone tag. We are able to build this $\overrightarrow{v}$ vector in code as follows:

```
% Get our initial vector
v = (1 / n) * ones(16, 1);
```

Now that we have our initial vector $\overrightarrow{v}$, we can observe that since $M$ is a regular, stochastic matrix, we can apply the Power Method Convergence Theorem to observe that the sequence $\{M\vec{v}, M^2\vec{v}, M^3\vec{v}, \dots\}$ will converge to a steady state vector $\vec{v}^\star$. Thus, we can computationally solve for our steady state vector in MATLAB as follows:

```
% Calculate PageRank
prev_ratings = M * v;
% Iterate until we have approached the steady state vector
for i = 1:999 % Cut off at 999 in case we don't converge
    % Get the next ratings
    ratings = M * prev_ratings;

    % Check if the ratings have converged to steady state vector
    is_equal = true;
    for j = 1:n
        if abs(ratings(j) - prev_ratings(j)) > 0.001
            is_equal = false;
        end
    end

    % If the ratings have converged, break out of the loop
    if is_equal
        break;
    end

    % Otherwise, continue
    prev_ratings = ratings;
end
```

Now that we have our steady state vector, we can sort and display the results of our PageRank. We can do this in MATLAB by using the `sort` function. We can do this as follows:

```
% Display the sorted results of PageRank
[sorted_ratings, sorted_indices] = sort(ratings, 'descend');
for i = 1:n
    fprintf('%s: %f\n', labels(sorted_indices(i)), sorted_ratings(i));
end
```

**Note:** This assumes you have declared a list of celebrity names called `labels`, where `labels(i)` is the name of the celebrity with index $i$.
Observing the output from the console, we can see the following results:

| Ranking | Celebrity | Rating |
|---|---|---|
| 1 | Beyoncé | 0.155709 |
| 2 | Kim Kardashian | 0.137929 |
| 3 | Taylor Swift | 0.085853 |
| 4 | Jay-Z | 0.084664 |
| 5 | Justin Bieber | 0.077819 |
| 6 | Ariana Grande | 0.068568 |
| 7 | Kylie Jenner | 0.068050 |
| 8 | Selena Gomez | 0.068018 |
| 9 | Ed Sheeran | 0.044216 |
| 10 | Kendall Jenner | 0.035700 |
| 11 | Dwayne Johnson | 0.035048 |
| 12 | Kevin Hart | 0.032981 |
| 13 | Ellen DeGeneres | 0.032136 |
| 14 | Oprah Winfrey | 0.030741 |
| 15 | LeBron James | 0.024466 |
| 16 | Michelle Obama | 0.018101 |

**Note:** This model was generated with a damping factor of $p = 0.15$.

# 3 Conclusion

mention what p explicitly means
(maybe in math section?)

## 3.1 Model Analysis

Based on the results generated in 2.2, we can come to the conclusion that Ed Sheeran has a low likelihood of being ”it” after a long time of playing phone tag. This is due to the fact that he has ranked 9th according to the results of our PageRank analysis, with a 4.4216% chance of being ”it” when we use a damping factor of $p = 0.15$. However, Ed Sheeran's rating and ranking could be changed if the damping factor was modified. This is because the way that our model is structured, an increase in damping factor will indicate that a celebrity's assistant is more likely to call a random celebrity. This has the effect of putting less emphasis on the relationships defined in our graph shown in 1. Thus, since Ed Sheeran is fairly well connected in our graph, he is likely to have a lower ranking if we were to put more emphasis on the random traversal implied by a high $p$ value. We can confirm this by regenerating our model with $p = 0.85$. If we do this, we can observe that Ed Sheeran now ranks 14th. However, the probability of him being ”it” at the end of the game is slightly higher, now at 5.7909%, with the larger $p$ value. This is because increasing the $p$ value has the effect of evening out the ratings generated by PageRank.

Conversely, we can observe that lowering the $p$ value will generally have the effect of descreasing the likelihood that Ed Sheeran is ”it” at the end of this game. This is because by lowering the $p$ value, we are putting less emphasis on random traversal. This means that the graph defined in 1 plays a more important role in determining who is ”it” at the end of the game.

## 3.2 Areas of Further Study

Spacing inconsistent (*)

While the PageRank model used to predict the outcomes of a game of phone tag has advantages in simplicity, it certainly could improve in a few key areas. For example, when creating our transition matrix in 2.1, we assumed a uniform probability that each celebrity was equally likely to call each celebrity that they have the contact details for. This is likely a massive oversimplification, as these probabilities could be dependent on a large number of variables. For example, it is likely that Beyoncé is more likely to call Jay-Z than Kim Kardashian, so the weight from Beyoncé to Jay-Z should likely be higher than the weight from Beyoncé to Kim Kardashian in our transition matrix.

Similarly, we assumed a uniform distribution of probabilities in our $B$ matrix used while creating the Google matrix in 2.1. If we had more information about the celebrity assistants, we could probably use better weights in our $B$ matrix and create a more accurate model. Another thing that we could put more emphasis on choosing our damping factor $p$. Rather than picking any $p$ value, we could put more consideration into the probability that an assistant would make the next call, or even use a function to get different $p$ values that are dependent on external factors.

Overall, by taking further consideration into these different techniques, we could likely greatly improve the efficacy of our model.

## 3.3 Concluding Remarks

Using PageRank to determine the likely results of a game of phone tag between celebrities does seem to be effective. Despite a few areas that could be improved upon, this techique has proved valuable in our analysis. This is because it gives us both a relative ranking and an objective probability that any one person is ”it” by the end of the game. The simplicity of this model allows low computational overhead, and further consideration taken during certain steps of computation could certainly improve the outcome generated by this model.