Mining on a Social Graph

Kiran Noolvi
CS
UT Dallas
Texas, United States
kxn180017@utdallas.edu

Saisuhas Kodakandla
CS
UT Dallas
Texas, United States
sxk1800114@utdallas.edu

Divya Gummadapu
CS
UT Dallas
Texas, United States
dxg170018@utdallas.edu

Aishwarya Madabushi Gadavarthi
CS
UT Dallas
Texas, United States
axm180011@utdallas.edu

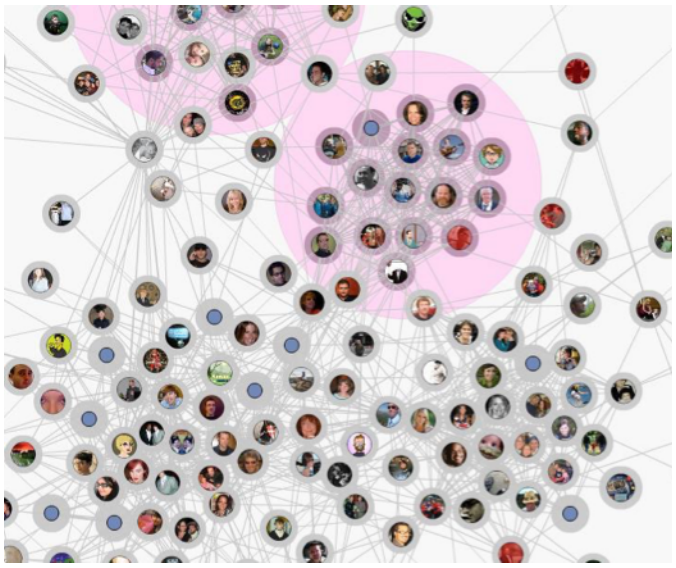
***Abstract*—This paper is the report for our final project on Mining of a social graph. BFS is used to find the degrees of separation between any two nodes in the graph. We performed the analytics of the result with running time and presented the results**

I. Introduction

For our project, we've used the concept that two individuals will be strangers if they have six or more degrees of separation. For example, if handshakes are a reason for transmission of a disease (like covid19 in the present scenario) then, a target can be affected by a source if they are at most six handshakes apart. We can find the number of degrees of separation between a source node and a target node by breadth-first search. Moreover, we can get the top famous things from the data by using GraphX. This is the main idea behind this project. The implementation of these algorithms are discussed in detail in separate sections

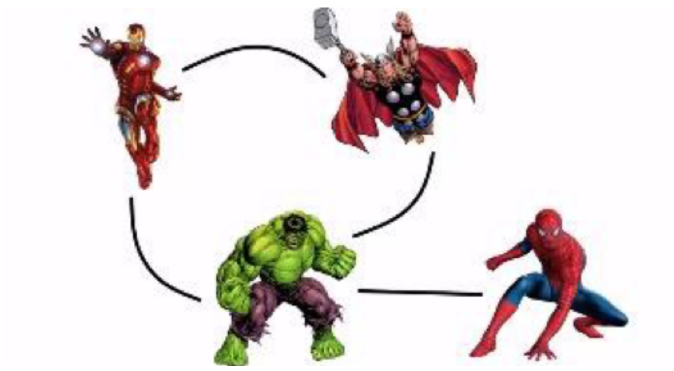
II. THEORETICAL STUDY

The main concept of implementing the concept of degrees of freedom is that we have to find an efficient way in which vertices can be found with minimum distance to find out the degrees of separation between two nodes. An algorithm which keeps minimum distances/ shortest paths between two vertices is required. One such efficient way is Breadth-first search on this graph. With breadth-first search, we can calculate the result between two vertices. And with GraphX we can compute the most frequently occurring nodes in a file what we provided as the dataset.

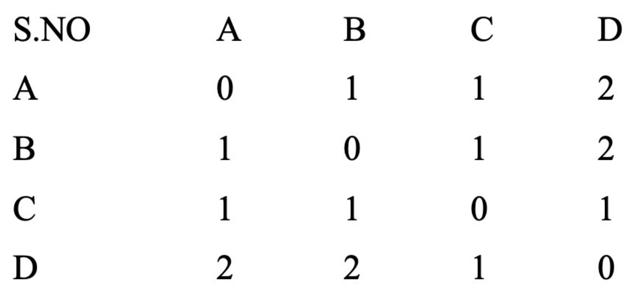


A. Algorithm to find degrees of separation by Breadth-first search in a graph

The details presented in this section are about Spark's iterative breadth first algorithm. A few things can be inferred from the picture presented below



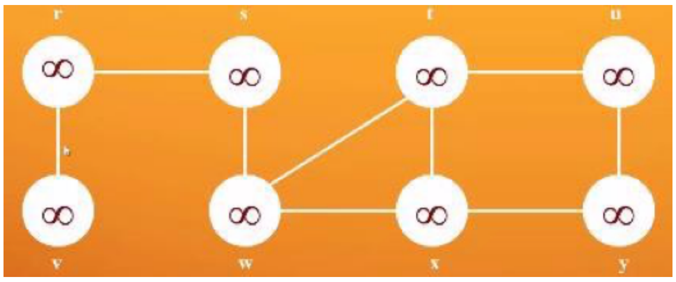
The above picture constitutes of four characters and for convenience let the names be A,B,C and D. The distance from D to both A and B is of degree two. The distance from D to C is of degree one. Similarly, from C the distance to all other characters is one. B is a at a distance of degree one from A and C and of two from D. Finally, A has a degree one distance from B and C and distance of degree two from D. The below table represents the distance between the characters



S.NO	A	B	C	D
A	0	1	1	2
B	1	0	1	2
C	1	1	0	1
D	2	2	1	0

Identify applicable funding agency here. If none, delete this.

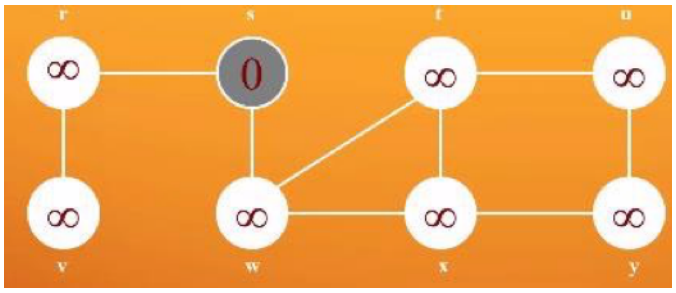
There can be more than one way to move from one character to another. For example, to traverse from A to D, we can either choose from the available three paths. One is A-B-C D and the other is A-B-D and the final one is A-C-D. The degree's of these paths are three, two and two respectively. Ultimately, we need to choose a path that provides us with the lowest degree that can be considered as the degree of separation between the vertices. So, the path chosen to traverse from A to D would be the one with degree two in this case. In case of graphs with large number of vertices we use an algorithm called breadth first search to find the shortest path and smallest degree. To understand the concept breadth first search easily let us move from the picture provided above to a regular graph shown below.



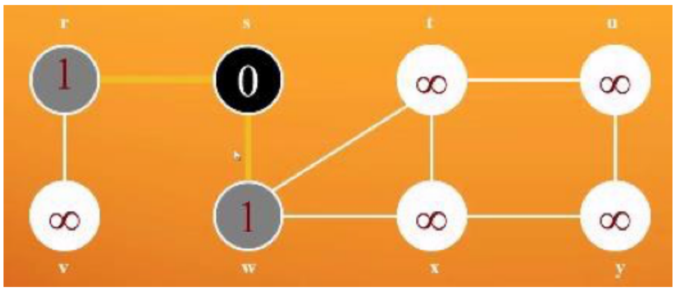
The picture shown below is the initial graph before any operation has been performed. Initially all the vertices hold a value of positive infinity. Consider the vertices mentioned in the above graph to be the characters from our dataset and the whole graph to be our data. The numbers on the vertices which represent the distance between the characters and will be revised with each step of the algorithm. Here, the relations between the characters is represented by the edges. Each graph has a root, and the distance from root to any node is represented in the vertex corresponding to that particular node. From the algorithm, the vertices are bound to hold a value which represents a relation with the lowest value. If only one path exists from source to destination, then the vertex is updated with that value or as per the working of the algorithm the minimum of the possible paths is considered. We use a set of colours to represent different states of vertices for better understanding. The colours used here are white, black and grey. Each colour has its own significance in the algorithm. Initially when the node is not yet processed, the colour white is used to represent it. When the node is in the process of being processed, it will be grey in colour. Once, the node is completely processed, it will be represented in the colour black.

Now let us go through the steps involved in the algorithm based on this graph. Let the root be the node s. The distance from any node to itself is zero which is represented in its vertex. Its colour changes from white to grey because it is in the process of being explored. It can be changed to colour black after all it's neighbours are done being processed. All this is represented in figure 4

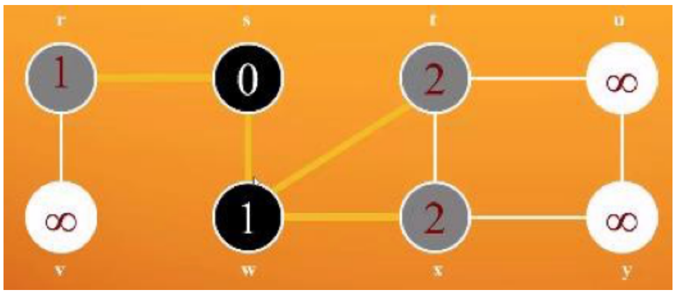
In the next step, we move from root s to it's neighbours

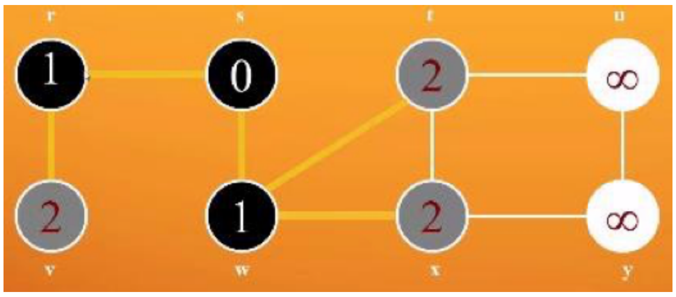


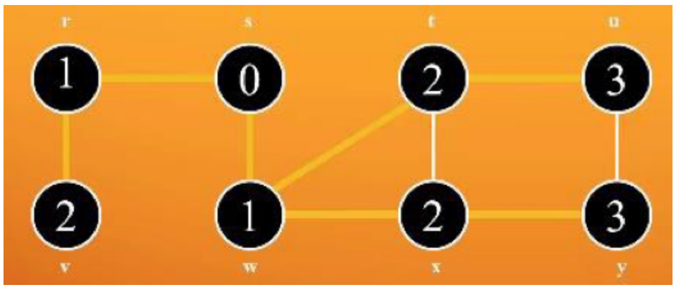
(w,f). Since all the neighbors of s are being processed and itself being explored the color of s can be changed to black and it has been completely explored. Now the colors of w and f can be changed to grey from white as they in in the process of being explored. They can be changed to black once their neighbors have been processed. All this is represented in figure 5



In the same way, the various others steps of the algorithm are represented in the following figures and the numbers represent the distances of particular node from root node. Once all the nodes are marked black and there is no other node to be explored the algorithm stops and we get the final distances.







Once the algorithm halts, we have the shortest distance from root to every other node. We can see that the algorithm checks every possible path and gives the shortest path. For example, from s to x we have many paths. One such path is from s-w-t-x with degree 3, other one being s-w-t-u-y-x with degree 5 and finally s-w-x with degree 2. Ideally, we choose the one with lowest degree of separation. So, in this case it is s-w-x with degree 2. The algorithm also gives the degree from s to x as two verifying it correctness.

B. To compute the most frequently occurred nodes using GraphX

As a part of the mining, we'll compute the most frequently occurring nodes in the graph by using GraphX library. In this, we'll parse through the file that contains characters and their connections and we'll find the most famous characters. By most famous characters, we mean the nodes that are connected to maximum number of nodes in the graph. This is accomplished both by using GraphX and without using GraphX and the results are presented.

III. RESULTS AND ANALYSIS

Given a source node and destination node, which represent the characters of our data, breadth first algorithm gives the shortest degree of separation between the nodes. The table below represents the output of breadth first search. The output includes source and destination nodes and the degrees of separation and the run time.

IV. FUTURE SCOPE AND CONCLUSION

Breadth first search is found to be a effective algorithm for finding shortest path between any given nodes for a large data set. We can find if there are transitive connections between the characters as an extension to this project. Further expanding this, we can provide source and destination nodes at run time and make the algorithm such that it calculates the degrees of separation between them. Moreover, we can keep a check for whether the input nodes given belong to the universe of these characters i.e., our dataset.

REFERENCES

[1] Shuhan Cheng https://ieeexplore.ieee.org/abstract/document/7516026.
[2] Luis Remis https://ieeexplore.ieee.org/document/7789331.
[3] Data set https://aws.amazon.com/datasets/marvel-universe-social-graph/
[4] Ravikant Dindokar https://ieeexplore.ieee.org/document/8291860
[5] Mayank Daga https://ieeexplore.ieee.org/document/7004254

S.NO	Source	Destination	Time(milliseconds)	Degree of seperation
1)	CAPTAIN AMERICA	THING/BENJAMIN J. GR	313	1
2)	CAPTAIN AMERICA	SPIDER-MAN/PETER PAR	268	1
3)	CAPTAIN AMERICA	IRON MAN/TONY STARK	243	1
4)	CAPTAIN AMERICA	THING/BENJAMIN J. GR	343	1
5)	CAPTAIN AMERICA	WOLVERINE/LOGAN	372	1
6)	SPIDER-MAN/PETER PAR	TECHNOCRAT/RANDY	695	2
7)	SPIDER-MAN/PETER PAR	WOLVERINE/LOGAN	307	1

Fig. 1. Analytics for few source and destination nodes

S.NO	ID	NAME
1)	859	CAPTAIN AMERICA
2)	5306	SPIDER-MAN/PETER PAR
3)	2664	IRON MAN/TONY STARK
4)	5716	THING/BENJAMIN J. GR
5)	6306	WOLVERINE/LOGAN

Fig. 2. Top five famous characters