Identifying The Influential Users Of A Location In Twitter

Sara Aein

Iran University of Sience and Technology
Department of Computer Engineering

Farzaneh Mahmoudi

Iran University of Sience and Technology
Department of Computer Engineering

Maryam Sadat Hashemi

Iran University of Sience and Technology
Department of Computer Engineering

Social media usage has increased in the last decade and it is still continuing to grow. Companies, data scientists, and researchers are trying to infer meaningful information from this vast amount of data. One of the most important target applications is to find influential people in these networks. This information can serve many purposes such as; product recommendation, public opinion analysis, voting and etc. In this study, a network-based method on twitter is proposed to find influential users in a particular location including countries and cities. We present Degree centrality algorithm to find most popular people. But it is not enough and we look another algorithm called Betweenness Centrality.

1 Problem Statement

In this section, we explain definitions and problems.

Definition 1. (user) user is defined to person who is signed up in Twitter. For unified notations, let u_i be a user, and U be the set of all users.

Definition 2. (users of a particular location) U_l denotes the users of a particular city or country and it is a subset of U.

Definition 3. (graph of Twitter network) We correspond Twitter network to directed graph $G := (V,E)$. Each user $u_i \in V$ in U_l set is a vertex in graph $G(V,E)$. There is a directional edge $e_{ij} \in E$ from the u_i to u_j if user u_i follows user u_j.

Given a set $\boldsymbol{U}$ and its subset U_l and the directed graph $G(V,E)$, we aim at designing an Algorithm to find influential users in $G(V,E)$.

2 Methods

In this section, first we explain about how we generate an appropriate dataset. Next, we detail two simple but efficient algorithms: Degree Centrality(Naive) and Betweenness Centrality.

2.1 Dataset

We consider generating 50 graphs as a test case with the different number of vertex which is gradually increasing. The number of vertexes At least is 5 and at most is 1000. The vertices are numbered from 1 through n. Then we assign a random number D to each vertex v_i as the degree of vertex which illustrate the number of edges connect to v_i. With regard to the degree of each vertex, we choose randomly another ver-

tex v_j to be the end of the edge e_{ij}. Here, vertices correspond to Twitter users and edges correspond to following relationship between users.

2.2 Degreee Centrality Algorithm(Naive)

In graph theory and network analysis, indicators of centrality identify the most important vertices within a graph. [1]

Degree centrality is one of the easiest to calculate. The degree centrality of a node is simply its degree—the number of edges it has. The higher the degree, the more central the node is. This can be an effective measure, since many nodes with high degrees also have high centrality by other measures. [2] In the case of a directed network, we usually define two separate measures of degree centrality, namely indegree and outdegree. Accordingly, In-degree is the number of connections that point inward at a vertex. Out-degree is the number of connections that originate at a vertex and point outward to other vertices. So, in our problem, The higher In-degree, the more follower of the node and the more influential user is.

The degree centrality of a vertex v, for a given graph $G := (V,E)$ with $|V|$ vertices and $|E|$ edges, is defined as $C_D(v) = deg(v)$.

Algorithm 1 Degree Centrality

```
1:  function INFLUENTIALUSER(vertexCount, edges)
2:      g ← newGraph(vertexCount)
3:      g.BuildGraph(edges)
4:
5:          ▷ Store Degree of each vertex and initialize with 0
6:      InDegree ← new array(vertexCount, 0)
7:
8:      for each vertex v in V do
9:          for each vertex u in V do
10:             if there is a edge from u to v then
11:                 InDegree[v] ← InDegree[v] + 1
12: return Max(InDegree)
```

According to Algorithm 1, it iterates over all vertices. For each vertex v, it checks if there is vertex u; such that u is a neighbour of v. If such vertex u exist, then we increase in-degree of vertex v. finally we calculate max In-degree.

Calculating degree centrality for all the nodes in a graph takes $\Theta(V^2)$ in a dense adjacency matrix representation of the graph.

2.3 Betweenness Centrality Algorithm

betweenness centrality is a measure of centrality in a graph based on shortest paths. For every pair of vertices in a connected graph, there exists at least one shortest path between the vertices such that either the number of edges that the path passes through (for unweighted graphs) or the sum of the weights of the edges (for weighted graphs) is minimized. The betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex. [3]

we inspire from the idea of betweenness centrality to address this problem. next, it is enough to find a vertex with maximum betweenness centrality in the graph as an influential user.

Algorithm 2 Betweennes Centrality

```
1:  function INFLUENTIALUSER(vertexCount, edges)
2:      g ← newGraph(vertexCount)
3:      g.BuildGraph(edges)
4:
5:      for each vertex v in V do
6:          for each vertex u in V do
7:              if u! = v then
8:                  BFS(u,v)          ▷ Find shortest path
9:                  Update betweense of each vertex
10: return Max(betweenness)
```

Due to the Algorithm 2, it iterates over all vertices. For each vertex v, it checks if there is vertex u; such that u is different from v. If such vertex u exist, then we find the shortest path from v to u. Next, we update the betweenness of all the node which is passed through the shortest path betweeen v and u;

According to previous researches, the best algorithm to find the shortest path in terms of time in a directional graph without the weight is Breadth First Search(BFS). The time complexity can be expressed as $O(|V|+|E|)$, since every vertex and every edge will be explored in the worst case. Eventually, Calculating betweenness centrality for all the nodes in a graph takes $O((|V|+|E|)V^2)$ in a dense adjacency matrix representation of the graph.

3 Experiments

In this article, We implement both algorithms which is explained in the previous section with *C#* Programming Language. There is Source code in this GitHub link. We illustrate our experimental result in

Fig. 1. with help of Seaborn python library. The Fig. 1. Compare Algorithm 1 and 2 in terms of duration of time In milliseconds that each Algorithms run in our Generated Data set when the number of nodes in graph less than 500 and between 500 and 1000. Obviously, the Degree Centrality is faster and more efficient than Betweenness Centrality.

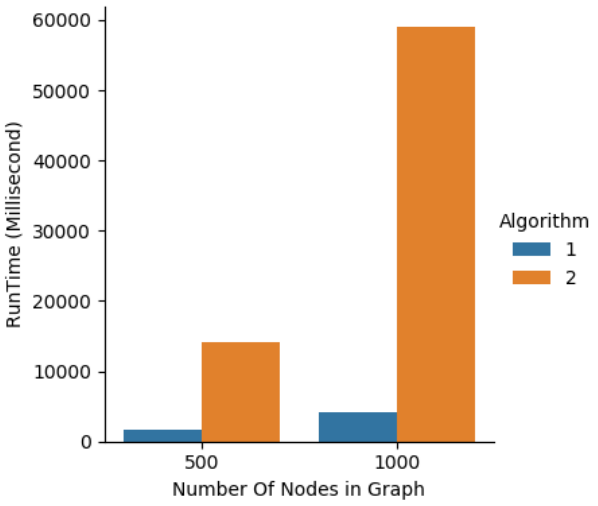

Fig. 1. Time running in milliseconds of Algorithm 1 and 2 when the number of nodes in graph less than 500 and between 500 and 1000.

4 Conclusions

In this article,we have presented Degree Centrality Algorithm and Betweenness Centrality Algorithm and we have also implemented these Algorithms and tested on our generated data set.

Although Degree Centrality is more efficient than Betweenness Centrality, we use Degree Centrality in order to find popular users with a large number of followers and Betweenness Centrality quantifies the number of times a user acts as a bridge along the shortest path between two other users. In reality, the users were obtained from the Betweenness Centrality is more important. So, we were able to increase the effectiveness With the help of Betweenness Centrality.

Finally, it is expected that more research or even a better understanding of the identifying influential users problem may lead to the emergence of more effective and efficient algorithms, as well as improvements in the Social Networks. [4]

References

[1] WikiPedia, 2019. Centrality. See also URL https://en.wikipedia.org/wiki/Centrality, May.

[2] sciencedirect, 2019. Degree Centrality. See also URL https://www.sciencedirect.com/topics/computer-science/degree-centrality, April.

[3] WikiPedia, 2019. Betweenness Centrality. See also URL https://en.wikipedia.org/wiki/Betweenness_centrality, April.

[4] Wei, H., and Pan, Z., 2018. "Identifying influential nodes based on network representation learning in complex networks". *Journal plos Name,* **1**(5), July, pp. 1–3. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0200091.