Noname manuscript No.
(will be inserted by the editor)

UACD: A Local Approach for Identifying the Most Influential Spreaders in Twitter in a Distributed Environment

T. M. Tariq Adnan · Md. Saiful Islam · Md. Tarikul Islam Papon · Shourav Nath · Muhammad Abdullah Adnan

Received: date / Accepted: date

Abstract News and information spread faster than ever, thanks to social media. The impact can be twofold: (i) efficient spreading of important information (e.g., public awareness, important alerts) can be extremely beneficial while (ii) quick spreading of wrong content (e.g., fake news, violent content) is alarming. Finding social media users who are the most influential at spreading information can be helpful for both cases. Existing accurate methods for finding the most influential spreaders use the global view of the network and require a single machine to process the entire data. Nowadays, with the growth of gigantic social networks, these methods are neither efficient (takes a long time) nor feasible (limitation of memory). Few existing methods only require local information of the network, but lack accuracy. Today, the popularity of social media allows us to collect rich user-specific information that can guide us towards designing more effective methods. In this study, we propose `UACD`, a novel method of identifying the most influential spreaders on Twitter social network by incorporating the user-specific information

TMT Adnan
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
E-mail: tariqadnan@cse.buet.ac.bd

MS Islam
Department of Computer Science
University of Rochester
E-mail: mislam6@ur.rochester.edu

MTI Papon
Department Computer Science
Boston University
E-mail: papon@bu.edu

S Nath
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
E-mail: shourav9884@gmail.com

MA Adnan
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
E-mail: adnan@cse.buet.ac.bd

(extracted from his/her Twitter account) to the topological information. We provide a distributed implementation of our proposed algorithm on the `Amazon EC2` cluster and observe that the algorithm is scalable and can process a significantly large network. We compare our ranking result with that of the state-of-the-art methods using widely accepted metrics of ranking comparison and our experimental results indicate that our new method is on an average **12.5**% more accurate and can produce the result in **175**× less time.

Keywords Influential spreader · k-core decomposition · distributed implementation · centrality measures

1 Introduction

Social media has gained remarkable popularity in the past few decades [20, 22, 28, 31]. The users like to discuss a diverse range of topics in social media, and generate a massive amount of data. According to Forbes magazine, "Social media has become the main source of news online with more than 2.4 billion internet users, nearly 64.5 percent receive breaking news from Facebook [67], Twitter [48], YouTube [16], Snapchat [10] and Instagram [41] instead of traditional media" [63]. Thus, today, social media plays a significant role in the diffusion of information [8]. Twitter, a micro-blogging service, is one of the most popular social media that allows people to share their ideas, opinions, and news with a large number of people in the network. Capturing, analyzing, and interpreting the information from social media has been an active research area [1, 26], popularly known as `social media analytics`. Twitter is very popular for researchers for its popularity and data availability. Many analyses have been done on Twitter data including but not limited to information credibility analysis [18], sentiment analysis [34], spam detection [11], and identification of the most influential spreaders [75, 87].

Information diffusion offers a widespread research area, and even researchers from different fields such as physics, biology, etc. are attracted to find out the insights of this area. Information diffusion has been studied for centuries in order to find out how innovation is spread over a network [58] and how contagious diseases are spread over any population [7, 38, 44]. It is highly possible to expedite the diffusion of beneficial information, obstruct or at least delay the spread of diseases, and accelerate the awareness among people by identifying the spreading pathways over any social network. According to Guille et al. [35], the information diffusion is studied based on three fundamental focus: (i) *identifying the mostly diffused information or topic*, (ii) *identifying the process, reason, and pathways of information diffusion during the past, and also prediction of the future*, (iii) *identification of the members on the network who play the most important roles in this diffusion process*. In this paper, we mainly concentrate on the third focus.

Identifying the most influential spreaders in a network is critical for ensuring efficient diffusion of information, which allows spreading awareness during the outbreak of any kind of epidemic [23, 77], successful e-commercial advertisements [50, 56], optimize the use of limited resources to facilitate information propagation [21], etc. It can also help to prevent fake news propagation [53], and controlling the spread of infectious diseases [7]. In a large social network graph, the nodes having the largest degree are often considered as the most influential

spreaders [76]. However, another important factor for a user's high spreadability is that he/she belongs to a *cohesive community*. The cohesiveness of a community is typically measured by k–core [80], which indicates how strongly the members of the community are connected. Briefly, the value of k in a k–core indicates the minimum number of edge removals required to disconnect a connected k–core. In general, a centrality measure is used to identify the most influential spreaders on a social network [25]. Such measures can be broadly classified into two kinds: *global* and *local*. Local measures take into account the local information of a node only (i.e., the neighborhood of the considering node) while global centrality measures consider the whole network structure during the centrality measurement of each node.

Initially, Borgatti et. al. [14] introduced the local measure *degree centrality* to model the spreadability of a user. Later, Kitsak et al. [46] suggest that the most efficient spreaders are located within the core of the network where the core can be identified using the k–core decomposition. Liu et al. [54] propose a more efficient method of using k–core decomposition by removing the unnecessary links which have a low diffusion significance. Another improved method was proposed by Wang et al. [86] who used k–core iteration factor as a tie-breaker of nodes with the same k value to evaluate the influence capability of a node. Local centrality measures, like, degree centrality [14] and k–core decomposition [45] are generally faster and more feasible, but less effective. They fail to find the most influential spreaders accurately because of omitting the global structure of the network. On the other hand, global measures such as *betweenness centrality* [68] and *closeness centrality* [72] are more accurate in finding the influential nodes in a network. However, global centrality measures require a single centralized machine to hold the entire network data and process them to generate the result which makes them very expensive in terms of both time and memory. Despite being more accurate, global measures are neither efficient nor feasible for very large networks like Facebook [67], and Twitter [48].

Additionally, the current literature only uses the network topology to find the influential spreaders and completely ignores the user-specific information (beyond the available social graph), which can play a vital role in information spreading. With the popularity of social media, it is possible to gather a surprising amount of user-specific information that may lead to more accurate techniques for finding the most influential spreaders. Inspired by this idea, we propose a novel method, `UACD` – User Attributed Core Decomposition – which identifies a new measure, `UACN` – User Attributed Core Number – which considers user-specific information to compute the spreadability of a user in the network. We provide intuitive arguments behind choosing each of our selected user-specific attributes. We rank the users according to their spreadability computed by our `UACN` measure and generate the ranks of the same users using each of the state-of-the-art methods. To generate the golden ranking (most accurate), we use the `SIR` (Susceptible-Infected-Recovered) epidemic model [83], which simulates the whole structure of the network for modeling the spread of infectious diseases in the network. As discussed earlier, the disadvantage of methods which run on the global structure of the network, is their inefficiency to process large networks. Therefore, we simulate a network generated from a small dataset on the SIR model to find out the actual most influential spreaders. We utilize widely accepted metrics like `Jaccard Index` [13], `Kendall Tau Correlation Coefficient` [71], `Spearman's Rank Correlation Coefficient` [3], `Normalized`

`Discounted Cumulative Gain` [43] to compare our ranking as well as the rankings generated by the state-of-the-art methods with the golden ranking. The experimental evaluation suggests that our proposed method is **12.5**% more (on average) accurate than any of the local methods while generating the results in a **175**× faster running time (on average) than our evaluating popular global methods.

Our method only requires local information (i.e., information of the node and its neighbors) making it suitable for a distributed setting, which helps to meet the objective of finding more accurate results in less time. We incorporate our measure with a distributed implementation of k–core decomposition [4] to find the most influential spreaders in Twitter social network. We deploy the implementation on `Amazon EC2 Platform` [6] and demonstrate how our proposed technique scales with the increment in cluster size. We observe that our algorithm can process large datasets (consisting of 41.7M Twitter users) in a reasonable time.

In summary, we make the following contributions:

- We propose a novel measure of finding the most influential spreaders on Twitter social network [48] which incorporates the user specific information with the traditional core value obtained by the more conventional k–core decomposition method. We refer to our measure as `UACN`: User Attributed Core Number and our measuring method as `UACD`: User Attributed Core Decomposition.
- Our proposed measure only uses the local information of the nodes (users), while being able to maintain at least the same accuracy as the global measures.
- To the best of our knowledge, we are the first to incorporate user-specific information into the network topology in order to find the most influential spreaders on a large social network.
- We evaluate our newly proposed `UACD` on three real Twitter datasets. We compare the performance against the currently available measures of finding the most influential spreaders on a social network. Our experimental result indicates that our new method is able to provide on average $mathbf12.5\%$ better accuracy than the existing metrics and methods while being on average $mathbf175\times$ faster.
- The proposed measure is suitable for a distributed setting since it requires only local information of each of the nodes. A distributed implementation of our method on the `Amazon EC2 Cluster` [6] shows that the method is scalable and can process a significantly large network.

The rest of the paper is organized as follows. We introduce a brief technical background in Section 2. Section 3 discusses the state-of-the-art methods of finding the most influential spreaders on a social network. We present the methodology of our study in Section 4. In Section 5, we discuss the metrics used to evaluate our proposed method. We present our experimental setup in Section 6 and evaluate our proposed method in Section 7. Section 8 demonstrates the performance of our method in a distributed environment (Amazon EC2). Finally, we mention some potential scopes of improvement in Section 9, and conclude the paper in Section 10.

2 Technical Background

In recent years, the number of users of social network sites have been increased by a large scale. People now can publish their statuses and get connected with

other users which we can refer to as social relationship [33]. As we have already discussed in the earlier section, we can formally represent an online social network using a graph, where each user is represented by a node of the graph and the social relationships among them are represented by edges (directed or undirected) [70]. In this graph, the nodes play an important role to disseminate information. This knowledge of node *spreadability* is very significant while it comes to develop an efficient method of accelerating the spreading in the case of information diffusion [58] as well as decelerate it in the case of diseases [38, 44]. Therefore, in recent years, the microscopic study of *spreadability* for each node has caught attention of the researchers. Moreover, it can be very useful in case of finding out the initial spreaders of a contagious disease [7] or any information [32]. In this section, we define some terminologies related to our work and then we provide a brief description of the existing techniques of finding the most influential spreaders in a network. Finally, we discuss the `SIR` model [83] which we use to evaluate the merit of our proposed technique.

2.1 Twitter Social Network

Twitter allows users to publish tweets containing short texts (within 140 characters) and/or multimedia content. *Twitter* employs the "following" model, where each user is allowed to choose who he/she wants to follow without seeking any permission. The user who is following the tweets of another user is called a "follower" while the other user is called his/her "friend". In this way, the users of Twitter form a social network with their friend-follower relationship. Typically, such networks are conceptualized as a graph, i.e. a set of vertices (or nodes) representing the users of the social network and a set of edges representing the social relationship among the vertices. In this study, we model the Twitter network using the Friend-Follower graph, where a vertex represents a user, and there exists a directed edge from vertex u to vertex v if and only if the user represented by v is a follower of the user represented by u. This also implies that the user represented by u is a friend of the user represented by v. For simpler modeling, these directed edges are often considered undirected. Figure 1 shows a simple friend-follower graph of Twitter social network. Here every node is labeled with the name of the user along with the Twitter screen name. The bidirectional edges are shown in bold lines implying the two users represented by the incident vertices are following one another.

2.2 Centrality Measurement

In graph theory, centrality is a term to describe the importance of an individual vertex within a graph or a network. Centrality measures [25] were initially developed for social network analysis since it helps answering the question, "Which vertices are the most influential in a graph?" The popular centrality measures for such analysis can be divided into two types: *local centrality measures* and *global centrality measures*, briefly described below.

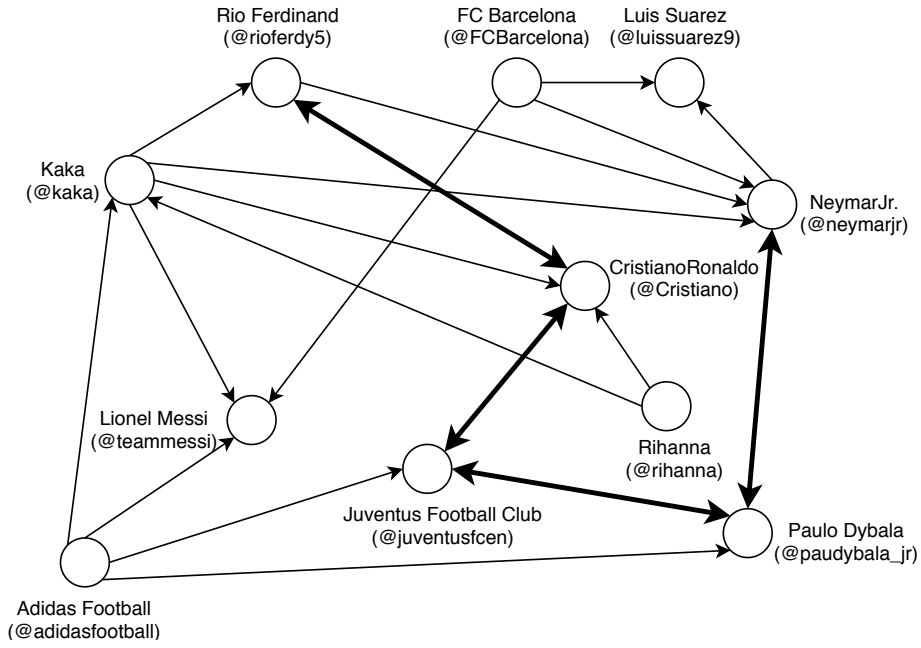


Fig. 1: Sample Friend-Follower Graph of Twitter Social Network

2.2.1 Local Centrality Measures

To compute the centrality of a node, local measures generally use the information of a node and its neighborhood only. The number of neighbors (degree of a node) plays the main role in such local measures and they are more suitable for networks modeled as undirected graphs. The two most popular local centrality measures are described below:

i) Degree Centrality: Among all the centrality measures, Degree centrality [25] is the simplest one. It assumes the nodes with maximum number of neighbors to be the most influential in the network. If, $G(V, E)$ is a graph with the set of vertices, V and the set of edges, E, then the degree of a vertex v can be denoted by d_v, and is defined as the total number of edges incident to it (i.e., the number of neighbors of v). Now, if the number of vertices, $|V|$ is n, then the degree centrality of node v is:

$$DC(v) = \frac{d_v}{(n-1)} \tag{1}$$

Here $(n-1)$ is used to normalize the value of degree centrality within 0 and 1. The most important reason for using degree centrality for finding the most influential spreaders is its simplicity and low computational complexity. However, this measure typically fails to identify the most influential spreaders accurately. Despite this, there are several cases where degree centrality can provide surprisingly good performance. For example, if the spreading rate is very small, degree

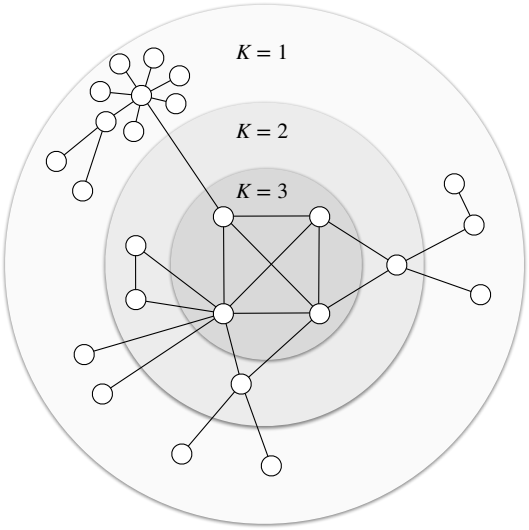


Fig. 2: A simple network with core numbers (k) of the nodes

centrality is reported to perform better in finding the spreadability of nodes than other well-known centrality measures [47, 52].

ii) K-Core Decomposition: In case of degree centrality, the number of neighbors is solely responsible in determining the the influence of any node in the network. Later Kitsak et al. [46] determined that the location of the nodes in the network can impact more significantly in their spreadability. They identified that nodes located at he center of the network possess higher probability to be the most significant spreaders in a network than the nodes located at the perimeter of the network. To sum up, they suggested that the *core number* of a node should be considered as a more suitable measure in order to identifying the most influential spreaders of the network, and this *core number* can be determined by the k–core decomposition [5, 27] of the network.

At the very beginning of the k–core decomposition method, all the nodes with degree 1 are removed. This process is recursively continued until no node with degree 1 is remaining in the network. These removed nodes are grouped together to form 1–*core*. After that, in a similar way, all the nodes with residual degree 2 are removed recursively until no node with degree 2 is remaining in the network. All these removed node are then grouped together to form 2–*core*. This process is continued until all the nodes are assigned to some core group. In this conventional k–core decomposition method, nodes having the largest core number are considered to be located at the center of the network and they are assumed to have the most spreadability. Figure 2 presents a simple network and the core number for the nodes.

The drawback of k–core decomposition method is that it has a tendency to assign the same core number (k value) to multiple nodes in case of large networks. Therefore, we may end up having a huge number of nodes as the most influential spreaders, which may not be a desired outcome in many cases. However,

the simplicity and lower computational complexity make this measure very useful. We suggest that incorporating user-specific information while computing the core decomposition helps to find the influential spreaders more accurately. Based on this, we propose `User Attributed Core Decomposition` (`UACD`) method, and show that our proposed method significantly improves the accuracy without incurring noticeable computational overhead.

2.2.2 Global Centrality Measures

Global measures consider the whole network topology while computing the centrality of the nodes. Some of the most popular global centrality measures are briefly described below:

i) Closeness Centrality: Closeness centrality [25] is a measurement which identifies the closeness of a node from all the other nodes in the network. If the network can be represented by a connected graph, then the normalized version of closeness centrality of any node u of the graph is calculated as the average of all the shortest paths between u and all other nodes of the network.

Let l_{ij} be the length of shortest path between any two nodes (v_i, v_j), and n be the number of nodes in the network, then the average shortest distance of node v_i from all other nodes i.e. closeness centrality can be defined as [79],

$$L_{v_i} = \frac{1}{n-1} \sum_{i \neq j} l_{ij} \tag{2}$$

The closeness centrality of node v_i can be thought as the inverse of the average shortest distance, L_i and defined as,

$$CC(v_i) = \frac{1}{L_{v_i}} = \frac{n-1}{\sum_{i \neq j} l_{ij}} \tag{3}$$

However, this equation will not be suitable to use for a disconnected graph where some nodes may be unreachable from a node v_i. For graphs with multiple connected components, Wasserman and Faust [73] revise the definition of closeness centrality. The closeness centrality of node v_i is now measured as the ratio of "the fraction of the nodes in the network which are reachable from v_i", to the "average shortest distance of v_i from the reachable nodes". Let n_i be the number of reachable nodes in the network from node v_i, then the modified formula of measuring closeness centrality is,

$$CC(v_i) = \frac{n_i - 1}{n-1} \frac{n_i - 1}{\sum_{i \neq j, v_j \text{ is reachable from } v_i} (l_{ij})} \tag{4}$$

ii) Betweenness Centrality: Betweenness centrality [25] determines how many times a node falls along the shortest path between two different nodes (i.e., acts as a bridge between those two nodes). Linton Freeman [30] introduces this measure for quantifying the control of a user on the communication between other users in a social network.

Let v_s, v_f, and v_i are three different nodes in a network represented by $G(V, E)$. Now we define $n_{sf}^i = 1$ if node v_i lies on the shortest path between v_s and v_f, and 0 otherwise. The betweenness centrality of node v_i is defined as:

$$BC(v_i) = \sum_{(s,f) \in V} n_{sf}^{i} \tag{5}$$

However, there can be more than one shortest path between v_s and v_f and that may count a node for centrality measure more than once. For this reason, if total number of shortest paths between v_s and v_f is g_{sf}, the equation for finding betweenness centrality of node v_i is updated as,

$$BC(v_i) = \sum_{(s,f) \in V} \frac{n_{sf}^{i}}{g_{sf}} \tag{6}$$

iii) Eigenvector Centrality: Eigenvector centrality [25] computes the centrality of a node in a network based on the centrality of its neighbors. The weights of the neighbors are assigned in such a way that a high scoring neighbor contributes more to the centrality of the node. Assume that, a graph $G(V, E)$ is represented by an adjacency matrix $A = \{a_{ij}\}$, where $\{a_{ij}\} = 1$ if nodes v_i and v_j are neighbors and 0 otherwise. If λ is the eigenvalue of graph G, the eigenvector centrality for node v_i is the i^{th} element of the vector $\vec{x}$ defined by the equation:

$$A\vec{x} = \lambda\vec{x} \tag{7}$$

There can be multiple eigenvalues i.e. multiple values of λ and for which multiple solutions can be found. However, for the largest eigenvalue of the adjacency matrix A, according to the Perron–Frobenius theorem, there exists a unique solution of x which contains all positive entries [69].

2.3 Susceptible-Infected-Recovered (SIR) Model

The SIR model [83] is one of the widely used models in epidemiology. In an SIR model, a node in the network can be at one of the three states: Susceptible, Infected, and Recovered. For better understanding, we explain these states using "people" instead of "nodes".

- S (Susceptible): The group of people who have not been infected with the disease yet. Additionally, they are not immune to the disease, and therefore, they are under the threat of being infected in the future.
- I (Infected): The group of people who have already been infected with the disease. Moreover, they can transmit the disease to the susceptible neighbors with a probability of β.
- R (Recovered): The group of people who have either recovered from the disease or dead. The recovered people are immune to the disease and no longer can transmit the disease to the susceptible neighbors.

The SIR model can simulate the spread of an infectious disease based on the topology of the network and two parameters: infection rate (β) and recovery rate (γ). The simulation stops when the network has no more infected nodes.

The SIR model is capable of finding the spreadability of a node accurately by considering the node as the only infected node at the beginning, running the simulation multiple times, and considering the average. If the network contains n

nodes, one needs to simulate the entire network n times (with multiple runs) to find the spreadability of all the nodes. Although this is the most accurate method of finding the most influential spreaders, it is not realizable for a large network due to its high computational cost.

3 Related Works

In this section, first, we show some existing methods that use the previously discussed measures to find the most influential spreaders on a social network. The later part of this section presents the significance of distributed analysis of large networks to assess the spreadability of the nodes as well as shows some currently existing approaches to perform this in distributed platforms.

3.1 Basic Centrality Measures and k–core decomposition

It is generally believed that the most connected nodes (higher degree centrality) and nodes with high betweenness centrality are the most influential nodes in the network. However, Kitsak et al. [17, 46] proposed that the k-core decomposition (also known as the k-shell decomposition) method performs better in identifying the best individual spreading nodes in the network. The k–core decomposition method assigns a core index k_s to each of the nodes in the network. Nodes having the lowest k_s value are located at the edge of the network. On the other hand, nodes with the highest k_s value are located at the center of the network and are assumed to possess the maximum spreadability.

3.2 PageRank and Related Improvements

Cataldi et al. [19] proposed a method of finding the distribution of influence of nodes across the overall network by using the well known *PageRank* algorithm [74]. Each node is assigned a *PageRank* value in proportion to the probability that the node is visited during a random walk (set of nodes) of the network. However this approach only uses the topology of the network ignoring other significant properties such as the native features of the nodes and their way of processing the information. In case of directed networks, a simple variant of *PageRank* has been proposed by Lü et al. [57], namely the the *LeaderRank*. *LeaderRank* introduced a bidirectional link connected with every node in the network. This newly introduced node is named as a so-called ground node, and a standard random walk is applied to the updated network in order to find out the influential spreaders. *LeaderRank* was further improved by Li et al. [51] where they allowed nodes to have more fans from the ground node to get more scores. They named this updated version as $WeightedLeaderRank$ which performs better than the traditional *LeaderRank* with almost similar convergence speed.

Table 1: Summary of influential spreaders identification methods

Algorithm	Network Topology	User Info	Topic Info	Distributed
PageRank	✓			
Topic-sensitive PageRank	✓		✓	
IP		✓		
Topical Authorities		✓	✓	
HybridRank	✓			
UACD	✓	✓		✓

3.3 TwitterRank

Romero et al. [78] proposed IP(i.e. *Influence-Passivity*), which assigns every node (user) two scores; i) a relative $influence$ score and ii) a *passivity* score, according to their ratio of forwarding information. This approach is similar to the well known HITS algorithm [29]. However, there is no such concept of universal influencer, rather it is pragmatic that each individual is influential in one some specific knowledge domain(s). Therefore, Pal et al. [75] proposed a topic sensitive method by defining a set of nodal and topical features, which characterize each members of the network. Now, by using probabilistic clustering method over this defined feature space, they identified the most influential spreaders on a given topic. Another topic-driven algorithm based on *PageRank* was developed by Weng et al. [87] solely for Twitter, namely *TwitterRank*, where they identified Homophily in a community of Twitter. In this way, they identified the topic-sensitive influence among the twitterers.

3.4 HybridRank

Ahajjam et al. [2] present a new measure of centrality in a network namely, hybrid centrality to find the most influential spreader. They calculate an improved version of coreness centrality (ICC) and the eigenvector centrality (EC) to combine them to generate their proposed hybrid centrality. Their proposed hybrid centrality measure of a node v is defined as follows:

$$HC(v) = ICC(v) \times EC(v) \tag{8}$$

Where $ICC(v)$ is defined as:

$$ICC(v) = \sum_{u \in Nei(v)} C(u)$$

Here they associate the coreness of the neighbors' nodes to find the improved coreness of node v i.e. the centrality of a node is relatively higher if its neighbors are highly central.

All these methods described are summarized in Table 1. We can see that no single method combines the network topology and the user info while finding the most influential spreaders on a network. On top of that, none of the above

approaches has a proper distributed implementation. Approaches that tend to use the global topology for each of the nodes to find out its spreadability on the network, incur a high amount of computational time which makes them infeasible to use in practice. Our proposed Modified User Attributed Code Decomposition is based on the simplest approach for finding influential spreaders, the k-core decomposition which considers the network topology as well as the user specific information like number of followers, number of friends, whether the user is verified, etc. On top of that, we propose a distributed variant of the algorithm on a popular cloud computing platform Amazon AWS.

3.5 Distributed k–core Decomposition

Batagelj et al. [9] provided a centralized algorithm of k–core decomposition, which recursively deletes the vertices having a degree of less than k, along with their incident edges. This process continues until all the vertices and their incident edges are removed from the network. The required running time complexity of this algorithm is $O(max(m, n))$, and for connected networks, it reduces to $O(m)$.

However, with the higher availability of social media data, the size of the network representing graph can be so that that single host might fail to process them due to memory limitation. On top of that, in some scenario, the graph itself can be distributed into multiple host machines, and therefore, makes it impractical to gather all the partial data in a central repository. For large scale graph analysis, Montresor et al. [66] proposed a distributed algorithm for k–shell decomposition. Their algorithm is based on the maximality of cores, which states that for any vertex u, the *coreness* is the largest among all the k values such that the number of neighbors of u is at least k which belong to at least the k–shell. That means in order to compute the *coreness* of any vertex u, it is sufficient to know the information of *coreness* of its neighbors, which is a local information. Therefore, the distributed k–core algorithm proposed by [66] starts with each of the vertices estimating its own coreness and sharing it with the neighbors. At the same time, it also receives the estimates from its neighbors, and using them rectifies its own estimation once again. In case of any update, a new *coreness* value is generated shared with the neighbors. This overall process continues until a convergence is achieved.

Thomo et al. [85] provided a cluster-based implementation of k–core decomposition on Apache Giraph [61] on Amazon AWS. Their work provides an evidence that Giraph is suitable for large scale graph analysis.

4 User Attributed Core Decomposition (`UACD`)

We propose a method of finding the most influential spreaders in Twitter social network using User Attributed Core Decomposition. Our modified core value is computed by associating user information with the coreness of any node v of the network. We also provide a distributed implementation of our proposed method `UACD`.

4.1 Selection of User Attributes

As per the discussion of the earlier sections, it is evident that the conventional coreness (k value) of a node in a network is not solely good enough to find its *spreadability* in the network since it only considers the network topology. Furthermore, if two nodes have the same k value, there is no way to differentiate the *spreadability* of these two nodes by only k values. In our proposed user attributed core decomposition method, we accommodate various user information with the k (found from the traditional k –core decomposition method) value of each of the nodes. This user information can be extracted from the social network account (from the Twitter account in our case) of the corresponding user represented by the node in the network. In case of a standard social network dataset, it is most likely to be a subset of the real social network and it is not possible to analyse the global structure of the network using such kind of dataset. Furthermore, the larger the dataset i.e. the more it provides the global structure of the whole network, the higher the complexity becomes to identify the most influential spreaders in an efficient manner.

Therefore, it is evident that user information that provides significant additional information of the node on the real network (beyond the topology of the available dataset), associating them to the calculation of finding the spreadability of the corresponding user in the social network certainly can play a noteworthy impact.

The types of user information that are used to find the modified k value are as follows:

- **Number of followers**: The naive approach of finding the most influential spreader on a network simply declares the node with the maximum number of neighbours as the most influential. Intuitively, the most connected node is very likely to be an influential spreader of the network. Hence the *number of followers* is a key parameter which should be taken into consideration with k value.
- **Number of friends**: The argument of incorporating this parameter is very similar to the above argument. The users that have a higher number of friends, are likely to disseminate information among a large number of people.
- **Number of tweets**: This parameter can determine the *activeness* and *significance* of the user in a social network. The users that have a higher number of tweets, are likely to be more active in the social network or they have been in the social network for a large time which indicates his importance as an influential spreader. Consecutively, information diffusion through such a node is likely to be higher.
- **Verified status of the user**: Since a verified node is more trusted, it is likely to be an efficient spreader than a non-verified node.

4.2 Deriving `UACN` – User Attributed Core Number

We devise a formula to get the `UACN` score for each node. We first determine the traditional coreness of each of the nodes i.e. the k value, using the more

Table 2: Impact of the 4 parameters in calculating UACN

Rank	Followers Count	Friends Count	Tweets Count	k	UACN
1	80143987	124226	9009	713	43.686
2	107260970	297555	30458	678	43.589
3	110044764	611101	15683	655	42.185
4	28598476	1737	240848	698	40.624
5	108232757	221	10130	713	37.646

conventional k–core decomposition method. After that, we normalize each k value by the following formula

$$k_{norm} = \frac{k}{k_{max}}$$

After that, we used the following formula to get the UACN values. We refer to this equation as *UACN Formula* . The main idea is to emphasize those nodes that have a higher number of followers, friends, and tweets. In addition, we emphasize the nodes that are verified. We are assigning equal weights to all these 4 parameters.

$$\begin{aligned} \texttt{UACN} = k_{norm} \times (\log(n_{followers}) + \log(n_{friends}) \\ + \log(n_{status}) + isverified \times C) \end{aligned} \tag{9}$$

We used logarithm for the first three parameters because generally, these values increase exponentially and an efficient spreader is likely to have a higher value for these parameters. Table 2 shows the values of these parameters for some influential nodes and the intuitive rationale behind our formula. As we can see, a higher k value does not necessarily mean a higher UACN value. In Section 7, we'll discuss in detail about the performance benefits of using this UACN. As mentioned earlier, in this formula our goal was to boost up the k_s values of those nodes who are verified and which have a higher number of followers, friends, and statuses. As a result, the k_{norm} value is multiplied by the sum of these factors, and UACN is calculated. We set $C = 5$ as the multiplication factor with the $isverified$ attribute which actually provides it the equal weight of having 100,000 followers. We set the value of C empirically which is further described in Section 7.

4.3 Algorithm Description with Complexity Analysis

For implanting UACD, first, we have to identify the conventional core numbers of the nodes using k–core decomposition method. We have used an efficient implementation of k–core decomposition proposed by Khaouid et al. [45]. The Algorithm 1 shows the steps of deriving traditional coreness of the nodes using the algorithm proposed in [45] as well the steps of applying *UACN Formula* for finding user attributed core numbers of the nodes. The notations $|V|$ and $|E|$ respectively denote the size of the vertex set and edge set of considering graph G. The algorithm begins with initializing five necessary arrays: d, b, D, p, and UACN. For the sake of the algorithm, the vertices of our considering social network graph G is labels

with numbers starting from 1. Now, index i of array d holds the degree of node i in the graph. Array D is simply a sorted array of the labels of nodes based on their degree in G. Array b is populated in such a way that index i of array b holds the starting position of i degree nodes in array D. Finally, index i of array p holds the position of the i^{th} node in array D. The helping function $\texttt{ifExist}(d, i)$ returns 1 in case i exists in the array d. On the other hand, $\texttt{occurrence}(d, i)$ returns the number of occurrence of i in the array d.

The first for loop at line 21 runs for $|V|$ times, and at each time it picks a vertex v from D. The coreness of v at this time is $d[v]$. At this time vertex v is detached from graph G. Consequently, the for loop at line 23 scans for each vertex u from its neighbors, and the degree of u is decremented by one if it has a higher degree than v in graph G. Accordingly, the position of u in D is redetermined. This is achieved by swapping u with the first vertex w with the same degree value in D. Also, the arrays b and p are updated accordingly. When the for loop of line 21 is completed, the array d contains the coreness values of each of the nodes of graph G. The third for loop at line 37 normalizes the coreness values of the nodes by diving them with the maximum coreness among the nodes. Finally, the forth for loop at line 41 derives the `UACN` values of the nodes by associating the user attributes of the nodes using the *`UACN` Formula* of Equation 9. At line 43, the array holding `UACN` values is sorted and returned at the end of the algorithm.

The running times of the initialization of arrays d, b, D, and p are $O(|V|)$, $O(\texttt{max}(d_G))$, $O(\texttt{max}(d_G) \cdot |V|)$, and $O(|V|)$, respectively. The outer for loop of line 21 and the inner loop of line 23 run for $O(|V|)$ and $O(\texttt{max}(d_G))$ times respectively, which make an overall running time of $O(\texttt{max}(d_G) \cdot |V|)$. Finally, the `UACN` values are calculated and sorted at line 41 and 43 in $O(|V|)$ time and $O(|V|log_2|V|)$ time, respectively. Since $\texttt{max}(d_G) \gg log_2|V|$, the running time of Algorithm 1 is $O(\texttt{max}(d_G) \cdot |V|)$.

Table 3 shows a comparison of running time complexities of the comparing methods mentioned in the upcoming Section 6. Brandes [15] provided a BFS-based algorithm to determine the distance and shortest-path counts from each vertex. Using this algorithm as a pre-processing part, the complexity of Betweeness Centrality can be obtained as $O(|V| \cdot |E|)$. Iyer et al. [42] provided a faster algorithm for Closeness centrality which costs a running time of $O(|V|^2 \cdot log_2|V|)$. The Degree Centrality measure needs to count the incident edges on each of the vertices, and therefore, has a time complexity of $O(|E|)$. The main contributing part to the time complexity of HybridRank algorithm by Ahajjam et al. [2] is the measurement of eigenvector centrality. Therefore, both the HybridRank and Eigen Vector centrality have a time complexity of $O(|V| + |E|)$.

5 Evaluation Metrics

In this section, we briefly present the metrics we use to evaluate the merit of our proposed ranking technique against the already established ones. In the following two sections, we present the environment we use to run our experiments, the datasets we use, and the output of our experiments. We use multiple evaluation metrics so that the comparison can be performed from every possible aspect.

Algorithm 1: User Attributed Core Decomposition

Input : Vertex Set of Graph G: V
Edge Set of Graph G: E
Degree Set of Graph G: d_G
Follower Count of Vertices of Graph G: $followers$
Friend Count of Vertices of Graph G: $friends$
Status Count of Vertices of Graph G: $tweets$
Verified Status of Vertices of Graph G: $verified$

$d \leftarrow [0] \times |V|, b \leftarrow [0] \times \texttt{max}(d_G), D \leftarrow [0] \times |V|, p \leftarrow [0] \times |V|, \texttt{UACN} \leftarrow [0] \times |V|$
for $i \leftarrow 1$ **to** $|V|$ **do**
  $d[i] \leftarrow d_G[i]$
end
$b[1] \leftarrow \texttt{ifExist}(d, 1)$
for $i \leftarrow 2$ **to** $max(d_G)$ **do**
  $b[i] \leftarrow d[i-1] + \texttt{occurrence}(d, i-1)$
end
$index \leftarrow 1$
for $i \leftarrow 1$ **to** $max(d_G)$ **do**
  for $j \leftarrow 1$ **to** $|V|$ **do**
    if $d[j] = i$ **then**
      $D[index] \leftarrow j$
      $index \leftarrow index + 1$
    end
  end
end
for $i \leftarrow 1$ **to** $|V|$ **do**
  $p[D[i]] \leftarrow i$
end
for $i \leftarrow 1$ **to** $|V|$ **do**
  $v \leftarrow D[i]$
  for $u \in neighbour(v)$ **do**
    if $d[u] > d[v]$ **then**
      $du \leftarrow d[u]$, $pu \leftarrow p[u]$
      $pw \leftarrow b[du]$, $w \leftarrow D[pw]$
      if $u! = w$ **then**
        $D[pu] \leftarrow w$, $D[pw] \leftarrow u$
        $p[u] \leftarrow pw$, $p[w] \leftarrow pu$
      end
      $b[du] \leftarrow b[du] + 1$
      $d[u] \leftarrow d[u] - 1$
    end
  end
end
$k_{max} \leftarrow \texttt{max}(d)$
for $i \leftarrow 1$ **to** $|V|$ **do**
  $d[i] \leftarrow d[i]/k_{max}$
end
for $i \leftarrow 1$ **to** $|V|$ **do**
  ▷ Applying UACN Formula
  $\texttt{UACN[i]} \leftarrow d[i] \times (log(followers[i]) + log(friends[i]) + log(tweets[i]) + verified[i] \times C)$
end
$\texttt{sort(UACN)}$
return UACN

Table 3: Comparison of running time complexity among algorithms of finding the most influential spreaders on a network

Method	Time Complexity
Betweenness Centrality	$O(\|V\| \cdot \|E\|)$
Closeness Centrality	$O(\|V\|^2 \cdot log_2\|V\|)$
Degree Centrality	$O(\|E\|)$
Eigen Vector Centrality	$O(\|V\| + \|E\|)$
HybridRank	$O(\|V\| + \|E\|)$
Traditional K–core Decomposition	$O(\|E\|)$
User Attributed Core Decomposition	$O(\texttt{max}(d_G) \cdot \|V\|)$

5.1 Modified Jaccard Similarity Coefficient

Jaccard similarity coefficient is used in order to measure the similarity between any two finite sets. This metric can be measured as the ratio of the size of the intersection of the two comparing sets to the size of the union of them. If A and B are our two comparing sets, then the conventional Jaccard similarity coefficient $J(A, B)$ is defined as,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{10}$$

In this paper, we intend to measure the similarity of top n items from two different rankings. Let a and b are two comparing methods and A and B are the two generated rankings respectively, A_n and B_n are subsets of A and B respectively with top n elements, then we define modified Jaccard similarity coefficient $J_m(A, B)@n$ as follows,

$$J_m(a, b)@n = \frac{|A_n \cap B_n|}{n} \tag{11}$$

We use this modified metric mainly to test the proportion of the common users in the two n sized sets containing the top n most influential spreaders identified by any two comparing ranking algorithms. While comparing with an established and accurate method, the higher the overlap is, the more reliable the comparing ranking algorithm.

5.2 Rank Correlation Coefficient

In order to measure the strength of association and the direction of relationship between two variables, we use correlation analyses. It provides a quantitative measurement of the strength of relationship, which is referred to as correlation coefficient, and varies between $+1$ and -1. A perfect degree of association between the two comparing variables is denoted by a value of ± 1. As the relationship between the two variables gets weaker, this correlation coefficient value goes closer to 0. The $\pm$ signs of the correlation coefficient is the indicating factor of the direction of the association.

In Section 7, we measure the correlation between the ranked list of users (based on their spreadability) generated by our User Attributed Core Decomposition method and the rankings generated by other methods. We use two non-parametric rank correlations: Kendall tau and Spearman's rank correlation coefficient.

5.2.1 Kendall Tau Correlation Coefficient

In order to identify the similarities in the ordering of any two comparing rankings, we use the Kendall tau rank correlation coefficient. To do so, Kendall Tau correlation coefficient picks each pair of observations between the two comparing rankings, and then identifies the strength of association according to the concordance and discordance between the pairs. Assume that L_1 and L_2 are the two rankings that are to be compared. Then Kendall analysis takes the following two properties into consideration:

- **Concordant**: Any pair of items (x_1, y_1) in L_1 and (x_2, y_2) in L_2 are considered as concordant if and only if they meet one of the following two conditions:
  - $(rank\_in\_L_1(x_1) > rank\_in\_L_2(x_2)$ and $rank\_in\_L_1(y_1) > rank\_in\_L_2(y_2))$
  - $(rank\_in\_L_2(x_2) > rank\_in\_L_1(x_1)$ and $rank\_in\_L_2(y_2) > rank\_in\_L_1(y_1))$
- **Discordant**: Any pair of items (x_1, y_1) in L_1 and (x_2, y_2) in L_2 are considered as discordant if and only if they meet one of the following two conditions:
  - $(rank\_in\_L_1(x_1) > rank\_in\_L_2(x_2)$ and $rank\_in\_L_1(y_1) < rank\_in\_L_2(y_2))$
  - $(rank\_in\_L_2(x_2) > rank\_in\_L_1(x_1)$ and $rank\_in\_L_2(y_2) < rank\_in\_L_1(y_1))$

Kendall Tau correlation coefficient is denoted by τ. If L_1 and L_2 are two different rankings with n similar elements, $N(C)$ and $N(D)$ represent the number of concordant and discordant pairs respectively, then τ can be calculated using the following equation:

$$\tau(L_1, L_2) = \frac{N(C) - N(D)}{\frac{1}{2}n(n-1)} \tag{12}$$

5.2.2 Spearman's Rank Correlation Coefficient

Spearman's Rank correlation coefficient is represented by R_s, which is used to measure the strength and direction of the association between any two variables. The value of R_s varies from -1 to $+1$. The correlation is assumed to be stronger as R_s reaches closer to $+1$. A perfect positive correlation is $+1$ and a perfect negative correlation is -1. Assume that L_1 and L_2 are two rankings of same n elements. For any element x, if the rankings of x in L_1 and L_2 are $RNK_{L_1}(x)$ and $RNK_{L_2}(x)$ respectively, then the distance of ranks, $d = RNK_{L_1}(x) - RNK_{L_2}(x)$. This value is squared to remove any negative values and when written in mathematical notation the Spearman Rank formula looks like this:

$$R_s(L_1, L_2) = 1 - \frac{6\sum d^2}{n^3 - n} \tag{13}$$

5.3 Normalized Discounted Cumulative Gain, $NDCG$

In this paper, we use Normalized Discounted Cumulative Gain, $NDCG$ which is one of the widely used techniques to evaluate ranking systems. Let GT represent

the weighted set of all users who generate the network. The weights are the relevance of the nodes to be selected as the most influential spreader on the network and the set GT is sorted according to this relevance value. We can refer to these relevance values and the ranking of the nodes in this set as our ground truth.

Now let X be the ranking of nodes generated by any comparing methods. In X, nodes are ordered according to their spreadability in the network. We define cumulative gain for the first m rankings in X, $CG@m$ as:

$$CG@m = \sum_{i=1}^{m} rel_i \tag{14}$$

Where rel_i indicated the relevance value of X's i^{th} ranked node in the ground truth GT.

However, CG does not take into consideration the rank i of the elements in X. Discounted cumulative gain (DCG) penalizes each relevance value based on its rank in the results. Therefore, we define Discounted cumulative gain for first m rankings in X as:

$$DCG@m = \sum_{i=1}^{m} \frac{rel_i}{log(i+1)} = \sum_{i=1}^{m} \frac{2^{rel_i} - 1}{log(i+1)} \tag{15}$$

$IDCG$ is the DCG of the best possible results based on the ground truth. Therefore we define Ideal Discounted cumulative gain for first m rankings in GT as:

$$IDCG@m = \sum_{i=1}^{m} \frac{rel_i}{log(i^{(I)}+1)} = \sum_{i=1}^{m} \frac{2^{rel_i} - 1}{log(i^{(I)}+1)} \tag{16}$$

Where $i^{(I)}$ indicates the Ideal rank of a node in GT. $NDCG$ is obtained by dividing DCG by Ideal DCG ($IDCG$), which normalizes the gain within $[0, 1]$. Therefore $NDCG$ for first m rankings in X can be defined as,

$$NDCG@m = \frac{DCG@m}{IDCG@m} \tag{17}$$

In our evaluation section, we use the metric $NDCG@m(a, b)$ as the Normalized Discounted Cumulative Gain for the first m elements of a ranking generated by method a, taking the ranking generated by method b as our ground truth (GT).

5.4 Running Time

Apart from the ranking related metrics, we also provide the comparison in running time of our proposed method with the existing methods. As we have already discussed, the global techniques incur a large amount of time to determine the ranking of the nodes in a network. Therefore, to make them run within a feasible amount of time, we perform this running time comparison on datasets with a smaller number of nodes and edges.

5.5 Infection Rate on SIR Model

We use infection rate function on SIR model to evaluate our proposed method. This metric was introduced by Ahajjam et al. [2], which we use to compare different methods of finding the most influential spreaders by simulating the network on SIR model by setting the top 10 spreaders as the initially affected set. At any time t, the infection rate can be defined as,

$$IR(t) = \frac{N_I(t) + N_R(t)}{n} \tag{18}$$

Where $IR(t)$ is infection rate at time t, $N_I(t)$ is number of infected nodes at time t, $N_R(t)$ is number of recovered nodes at time t and n is the total number of nodes.

6 Experimental Setup

As we already have discussed, the global methods of finding the most influential spreaders take a large amount of time to generate the final result. Therefore, to evaluate the performance of our proposed User Attributed Core Decomposition method of finding the most influential spreaders on Twitter social network, we need to keep the size of the dataset considerably small. On the other hand, we run the experiments in a distributed environment with large scaled social network data which fails to run on a single computer since they experience memory overflow error and/or take an infeasible amount of time due to larger computational complexity. Consequently, we run our experiments on two different environments and they are described below:

6.1 Cross Validation on a Single Computer

Global metrics like closeness or betweenness centrality possess a very high computational complexity which makes them infeasible to apply on large datasets i.e. networks with a large number of nodes and edges. Therefore, in order to cut off the time requirement during the evaluation of the performance of our proposed method against such global techniques with higher time complexity, we run all the comparing methods on networks with a smaller number of nodes and edges on a single computer with simple commodity hardware.

6.2 Large Network Analysis on Distributed Environment

MapReduce [24], introduced by Google in 2004, is a very popular framework for parallelizing computational tasks for the processing of large-scale data-sets. Although it is possible to use Map-Reduce for graph processing, its structure is not natively optimized for this kind of tasks. That is why, another framework called Pregel [60] was developed by the Google researchers, which is highly optimized for processing graph data [37]. Apache Giraph [61, 62] is the open-source version of Pregel [12] built on top of Hadoop. The main idea of Giraph is "think like a vertex". The computation of Giraph is iterative, where each iteration is called a

superstep. At every superstep, Giraph executes a user-defined function on each of the vertices. During the execution of this user-defined function, at each superstep, each node sends messages to its neighbors, to be received at the upcoming superstep. Similarly, at each superstep, each node receives messages from its neighbors sent during the previous superstep. Each superstep is kept apart from the other by using barrier synchronization [81]. Whenever any node reaches the convergence, it can leave the computation. We use Apache Giraph for implementing our proposed User Attributed Core Decomposition method on a distributed environment as described in the previous subsection 3.5.

6.3 Methods Compared

We evaluate the merit of the ranking generated by our method against the ones generated by the following well established methods:

- Betweenness Centrality (BC)
- Closeness Centrality (CC)
- Degree Centrality (DC)
- Eigenvalue Centrality (EC)
- HybridRank (HR)
- User Attributed Core Decomposition (`UACD`): Our method

We simulate the graph networks of the datasets on the SIR model and as initially affected nodes, we use the topmost influential spreaders generated by all the comparing methods. We use the implementation of SIR model from the python module EoN [65]. The input networks of the *EoN* module are *NetworkX* [36] graphs. We also use this python module to find the centrality measures of the network.

6.4 Datasets Used

In order to test the performance of our proposed User Attributed Core Decomposition method of finding the most influential spreaders on Twitter social network, we mainly use three real Twitter datasets. Since we have to compare the ranking generated by our proposed method with the rankings generated by the global techniques which take too much time to generate the results, we consider multiple subsets of the main datasets with a smaller number of nodes and edges. Below we briefly describe the datasets we use for evaluating our proposed method.

- We use the dataset from Kwak et al. [49] which is collected by crawling the entire Twitter site for 6 months in 2009. This dataset contains 41.7 million of user-profiles, 1.47 billion friend-follower relationships, 4,262 trending topics, and 106 million tweets. However, due to Twitter's new Terms of Services, this dataset has removed the tweet contents. Therefore, we only generate the friend-follower graph from this dataset. We refer to this dataset as *Kwak-twitter-2009*. However, for cross validation with global techniques of finding the most influential spreaders, we make two subgraphs from this dataset with a smaller number of nodes and edges and we define them as follows.

  - **Kwak_50K**: We generate a subgraph from the main dataset with 50,000 randomly selected nodes and the edges connecting them in the main graph. Since at every run we select a different set of randomly selected graphs, the number of average incident edges on each node varies from 1.5 to 4. We refer this dataset to *Kwak_50K* in the upcoming sections.
  - **Kwak_100K**: Similar to the previous one, this dataset is another subgraph generated from the main one with randomly selected 100,000 nodes and their incident edges. We refer to this dataset to *Kwak_100K* in the upcoming sections.
- Another Twitter dataset we use is collected by Kristina Lerman [39] in the year of 2010. It is a dataset containing 736,930 users and 36,743,448 links of social relationship among them. In the following sections of this paper, we refer to this dataset as *Lerman-twitter-2010*. To make it feasible to run the global metrics on this dataset, we also generate a smaller dataset out of this one with randomly selected 100,000 nodes and their incident edges and we refer to this dataset as *Lerman_100k* in our upcoming sections.
- **Twitter-Dynamic-Net**: Lou et al. [55] and Hopcroft et al. [40] collected this dataset for their research works. To collect this dataset, one of the known popular users on Twitter was selected and then 10,000 of his/her followers were randomly collected. After that, these users were taken as seed users and a crawler was used to collect all followers of these users by traversing "following" relationships. The total number of users is 112,044. The crawler monitored the change of the network structure among the 112,044 users during December, 2010 and finally obtained 443,399 dynamic friend-follower relationships between them. In our evaluation section, we refer to this dataset as *Lou_Hopcroft*.

6.5 Users' Attribute Collection

Since our main goal is to incorporate Twitter's user profile information with the conventional core value found from k–core decomposition method to identify the most influential spreaders on Twitter social network, we have to collect the relevant user information (as mentioned in Eq. 9) from Twitter. We use Twitter *GET users/lookup* API [59, 84] that returns a json response containing a bunch of information among which we extract our required ones.

7 Experimental Outcome and Evaluation

As we mentioned in the Equation 9, we use a constant multiplication factor of five (5) to associate a weight to the *isverified* attribute of each of the Twitter user IDs. In this experiment section, first, we provide an empirical justification of the decision of setting five (5) as the multiplication factor. The later part of this section provides our experimental outcome and establishes the merit of our proposed User Attributed Core Decomposition method based on the metrics discussed in Section 5.

Table 4: Topological features of each of the used datasets. The value of epidemic infection probability, β is calculated as $\beta = \frac{average\ degree}{average\ second-order\ degree}$

Datasets	No. of Nodes	No. of Edges	Average Degree	Average Second Order Degree	β for SIR Simulation
Kwak_50K	50,000	166,523	3.33	107.45	0.031
Kwak_100K	100,000	487,010	4.87	200.48	0.024
Lerman_100K	100,000	2,043,091	20.43	1484.33	0.014
Lou_Hopcroft	112,044	443,399	3.68	81.78	0.045

7.1 Empirical Identification of Multiplication factor in *UACN Equation*

For each of our four considering datasets, we generate the ranked list of the most influential spreaders using the SIR model and our proposed user attributed core decomposition method. We determine the real ranking of the nodes for every dataset based on their spreadability by simulating the SIR model on the network generated from the corresponding dataset. For each network, the SIR model is simulated for 100 times with $\beta = \frac{average\ degree}{average\ second-order\ degree}$ and $\gamma = 1$ and then by averaging the outcome, we determine the set of users ordered by their spreadability. The $average\ degree$, $average\ second-order\ degree$ and corresponding β values for each of the datasets are given in Table 4. After that, we normalize the spreadability of each of the users dividing them by the spreadability of the first ranked user. These fractions are considered as the weight values and these weights are considered as the relevance of each of the users to be considered as the most influential spreader and this ranked list is used as the ground truth for computing Normalized Discounted Cumulative Gain ($NDCG$) to evaluate the merit of the ranking generated by our proposed method.

Now we determine the spreadability of each of the users by our proposed method. The merit of this ranking is evaluated using the $NDCG$ metric taking the SIR model generated ranking as the ground truth. We vary the value of the constant multiplier, C in the Equation 9 from 1 to 10 and each time determine the ranked list of the most influential spreaders. The merit of each of the rankings is evaluated using $NDCG$ value and compared the SIR model generated ranking and then the results are plotted on Figure 3. The curves are further smoothened by using interpolation. We can observe that the $UACN$ *Formula* provides better performance for $C \approx 5$ which is shown by a straight line in the graph. Therefore, we set $C = 5$ in the $UACN$ *Formula* (Eq. 9).

7.2 Merit Evaluation of User Attributed Core Decomposition Method

The following subsections provide the description of the experiments we conduct and based on that evaluate the performance of our proposed method of identifying the most influential spreaders. First, we compare the coefficient metrics and other evaluation indices among our comparing methods to measure their similarity in determining the most influential spreaders. After that, we simulate the network on SIR model with the topmost influential spreads identified by each of the comparing methods and provide a graphical demonstration of the performance of each of the comparing methods in identifying the most influential spreaders with the help of epidemiology.

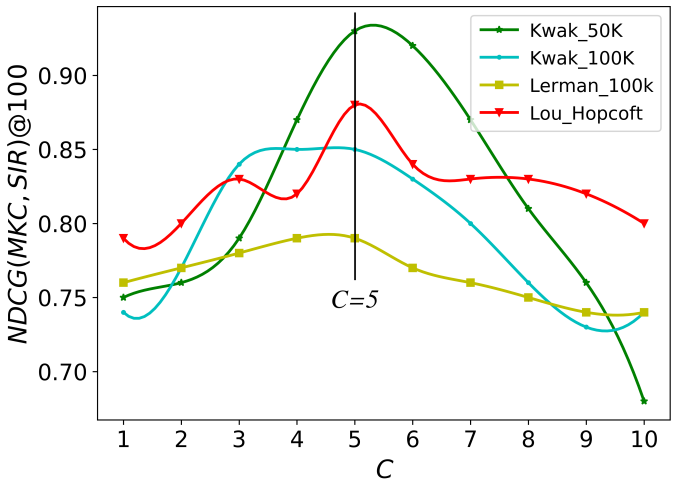


Fig. 3: $NDCG$ values for the top 100 most influential spreaders for the ranking generated by user attributed core decomposition method taking the SIR model generated ranking as the ground truth. The value of the constant, C in Equation 9 is varied to observe the impact. The maximum $NDCG$ value for each of the dataset is obtained around the $C = 5$ line.

7.2.1 Rank Correlational Coefficients

First, we measure Kendall Tau and Spearman rank correlational coefficient between the ranking generated by each of the comparing methods and the one generated by SIR method. Figure 4a and 4b respectively plots the results using vertical bars.

We can see that our proposed user attributed core decomposition method mostly outperforms the existing methods or at least provides a similar better result for each of the datasets. To be specific, `UACD` provides $2.2 - 22\%$ and $1.2 - 22.2\%$ more accurate result than the existing methods while we consider Kendall Tau and Spearman rank correlational coefficient, respectively. Here the closest competitor is the HybridRank (HR) proposed by Ahajjam et al. [2], which provides on an average 1.35% less accuracy for the considering four datasets .

7.2.2 Jaccard Index and NDCG

We evaluate the merit of each of the comparing methods by comparing their generated ranking with SIR ranking using the metric, Modified Jaccard Similarity Coefficient, and the results are demonstrated in Figure 4c using vertical bars. The results are shown for the first 100 most influential spreaders identified by each of the comparing methods. It can clearly be seen that the user attributed core decomposition method outperforms all other methods with $1 - 22\%$ better accuracy.

Figure 4d plots the Normalized Discounted Cumulative Gain ($NDCG$) values when top 100 most influential spreaders identified by each of the considering

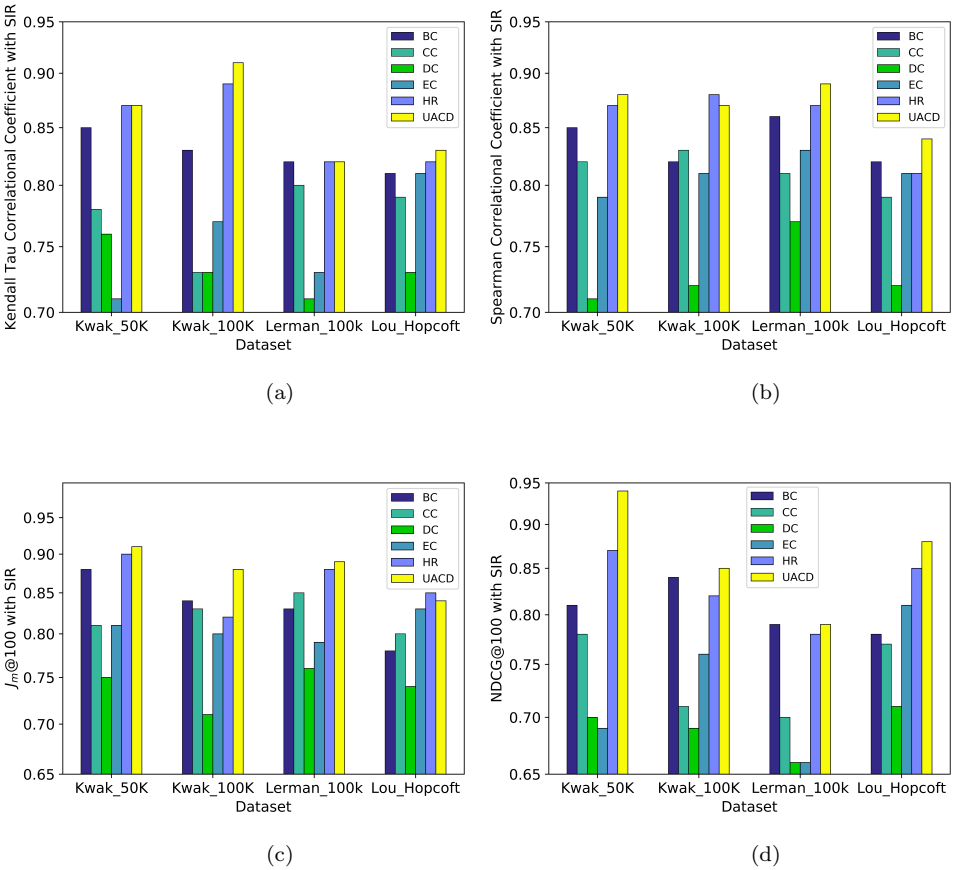


Fig. 4: Comparison between the ranking generated by the simulation of SIR model and by each of comparing methods based on the metrics: Kendall Tau (4a), Spearman (4b) Rank Correlational coefficent, Modified Jaccard Similarity Coefficient (4c), and NDCG (4d). The results of Modified Jaccard Index and NDCG are evaluated for the top 100 most influential spreaders identified by each of the methods.

methods are compared to the top 100 most influential spreaders generated by the SIR model. As earlier, the ranking generated by our proposed user attributed core decomposition method shows the maximum similarity ($2 - 26.6\%$ better than the competitors) with the ideal ranking generated by the SIR model.

To summarize, we can observe that based on the two ranking correlation coefficients, Kendall Tau and Spearman, and two comparison metrics, the Modified Jaccard Similarity Coefficient and NDCG, the user attributed core shows significantly decent performance in comparison to the state-of-the-art methods and provides a 12.5% better accuracy, averaging over all the comparing methods and considering datasets.

Table 5: Comparison of running time (in sec) for `UACD` on different datasets against existing methods for finding most influential spreaders.

Datasets	No. of-Nodes	No. of-Edges	Eigen Vector-Centrality	Closeness Centrality	Betweenness Centrality	HybridRank	User Attributed Core Decomposition
Kwak_50K	50,000	166,523	1.18	57.87	78.67	1.76	0.78
Kwak_100K	100,000	487,010	2.77	312.78	428.67	3.11	1.55
Lerman_100K	100,000	2,043,091	32.67	8165.78	9876.45	55.67	13.85
Lou_Hopcroft	112,044	443,399	3.31	487.77	544.88	3.96	1.28

7.2.3 Comparison on Running Time

We run all six comparing methods on the four datasets mentioned in Table 4, and observe the executing time. The results are shown in Table 5. The running times for degree centrality method are omitted since they do not possess enough significance based on the preceding discussion. It can be observed that our proposed User Attributed Core Decomposition method has the least running time among the comparing methods mentioned in the table. It is very intuitive since we use the traditional k–core decomposition method as the baseline which is the fastest among these comparing methods. The latter part of our method where we apply the *UACN Formula* contributes very little to the running time. Our proposed method can offer $\mathbf{1.5-100}\times$, $\mathbf{1.8-276.5}\times$, $\mathbf{2.4-713}\times$, and $\mathbf{2.6-425.7}\times$ faster running time than the comparing methods for the four listed datasets, respectively with an average of $\mathbf{175}\times$ improvement in running time.

7.3 Simulation on SIR Model

We simulate the SIR model to depict the disease diffusion based on the topology of our experimental datasets. For each method, the SIR model is simulated on the same dataset for hundred (100) times so that a steady result is obtained, and the simulation was run for infection transition probability, $\beta = 0.1$ and recovery probability, $\gamma = 1$. We observe the simulation up to a maximum time, $t_{max} = 16$. The simulations stop when saturation in infection spreading is achieved and Figure 5 shows the infection rate achieved with the flow of time for each of the methods on each of the datasets. Every time we select the top 10 most influential spreaders identified by the comparing methods and assign them as the initially affected nodes on the SIR model. We can clearly see that the infection rate while selecting the top 10 most influential spreaders obtained by our User Attributed Core Decomposition method provides equally better performance or outperforms all other comparing methods. On top of that, our method can generate accurate result in significantly lower computational complexity than the global methods.

8 Large Network Analysis on Distributed Environment

We configure the distributed environment on Amazon EC2 [6], which provides a managed Hadoop [82] framework in order to simplify big data processing. Hadoop makes it easier, faster, and cost-effective to process vast amounts of data in a distributed manner across dynamically scalable Amazon EC2 instances.

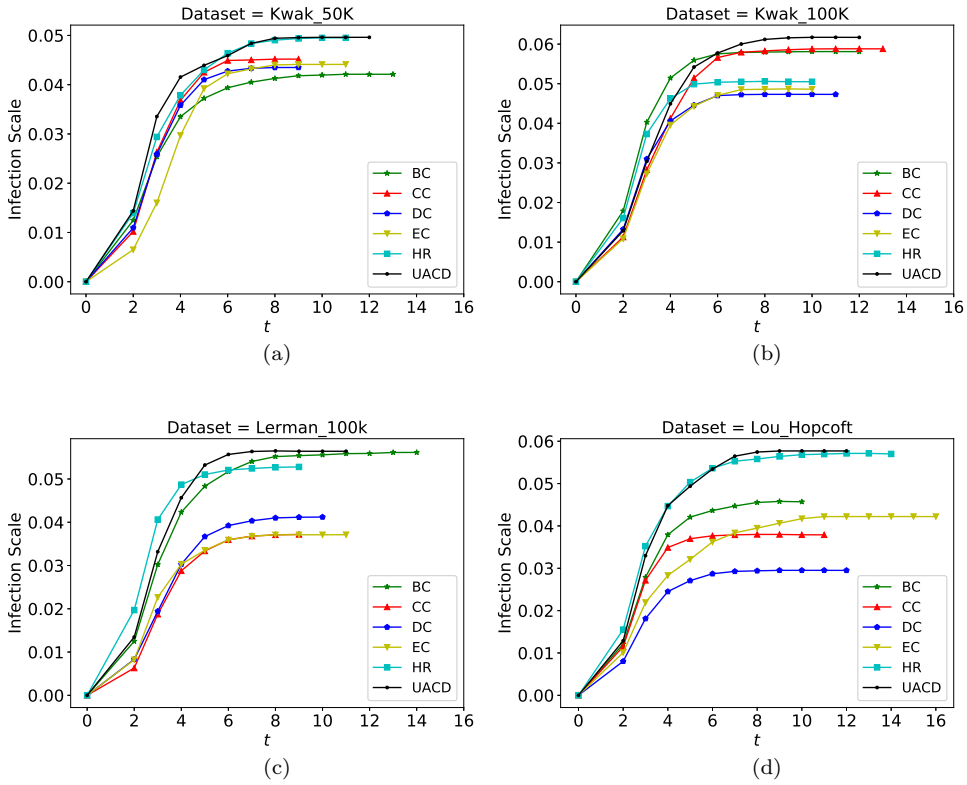


Fig. 5: Simulation of SIR model on 4 datasets using each of the methods.

Dataset	No of Edges	No of Vertices	Running time	No of Iterations	Maximum K value	Average K Value
Lerman-Twitter-2010	0.7M	36M	1218	80	334	2.13
Kwak-Twitter-2009	41.6M	41.7M	2987	98	713	3.87

Table 6: Summary of User Attributed Core Decomposition using Apache Giraph on top of Hadoop Cluster with 4 Slave Nodes

8.1 Frameworks Used

On our Amazon EC2 cluster, we install *Hadoop 2.4.0* for distributed framework. We also install *Apache Giraph* [61] for distributed graph processing. We rebuild the Giraph framework with the source code of k–core decomposition and apply k–core decomposition in a distributed manner. Then we apply our own $UACN$ *Formula* which includes node information and the result shows it can handle big datasets with a large number of users and social links.

8.2 Experimental Outcome

In our distributed environment, we apply the User Attributed Core Decomposition method on our two high dimensional datasets, *Lerman-twitter-2010* and *Kwak-twitter-2009* and the experimental summary is noted on Table 6. This particular experiment is carried out on a cluster with 1 master node and 4 slave nodes. We install a well known scalable distributed monitoring system, Ganglia (version 3.7.2) [64] to keep an eye on various parameters of the cluster during the experiment. We vary the number of slave nodes in our cluster setup from 3 to 12 and Figure 6 shows the impact on the execution time. We can see that with the increase in cluster size, Giraph provides a significant decrease in running time i.e. a good scalability until a certain cluster size and after that, the running time again starts to increase. In Figure 6, we can observe that the execution time is lowest for 7 and 8 slave nodes respectively for the datasets *Lerman-twitter-2010* and *Kwak-twitter-2009*.

In case of distributed cluster computing, when a job is submitted for execution, the job is divided into multiple independent tasks and those tasks are assigned to be carried out by different slave nodes. When all the tasks are completed, the partial results can be aggregated by transmitting messages within the cluster. As we already have discussed in Subsection 6.2, the computation in Giraph introduces supersteps which result in the transmission of messages regarding any modification in vertices' local information during the execution among the worker nodes. Now intuitively we can say that, as the number of slaves increases in our cluster, the job is divided into more tasks and the tasks are distributed in the larger number of slave nodes which eventually decreases the time per task and this contributes to the decrease in execution time. However, at the same time, the increased number of slave nodes increases the number of transmitted messages within the cluster. This contributes adversely to the execution time and eventually, after a certain size of the cluster, the overall running time starts to increase. Figure 7(a) and 7(b) provides a visual explanation of this phenomenon for datasets *Lerman-twitter-2010* and *Kwak-twitter-2009* respectively. Both the figures show the changes in average CPU usage per slave node and the number of transmitted messages within the cluster. As per our preceding discussion, we can observe that average CPU usage per slave node gradually decreases with the increase in the slave count while the number of transmitted messages increases. The rate of decrements in CPU usage is much higher than the rate of increment in message count until the two curves intersect. On the other hand, after the intersection, the rate of increment in message count is much higher than the rate of decrements in CPU usage.

9 Future Work

The performance can be further increased by incorporating topic information. For example, by examining the past statuses of a user, we can gather information about his specialty. This information can increase the performance of finding influential spreaders dramatically. As our future work, we plan to find the topic specialization of a user and try to improve the performance by adding topic specialization. We also plan to evaluate the performance on various other networks as well.

An online implementation of our method can show each user's *spreadability* by some manner in an active or "live" network. If the node's information and his

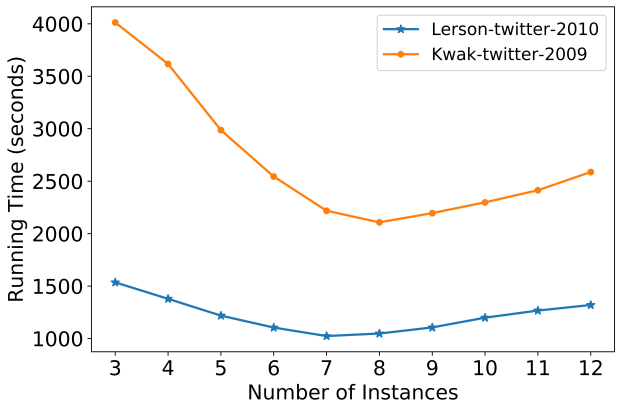
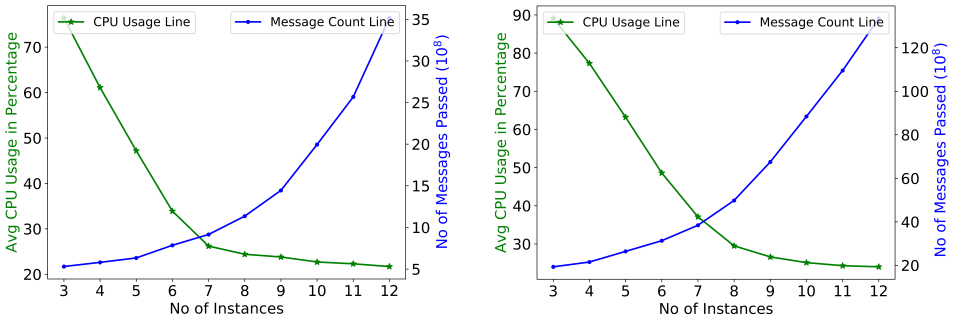


Fig. 6: Speed up with the increase in cluster size



(a) (b)

Fig. 7: The amount of CPU utilization decreases as the number of instances in the cluster increases. On contrary, it gradually increases the intra-cluster message transmission. The intersecting point of these two curves provides the optimum number of instances in a cluster when we run our User Attributed Core Decomposition method in a distributed environment. Figure (a), (b) are shown for datasets *Lerman-twitter-2010* and *Kwak-twitter-2009*, respectively.

neighbor's information are known as apriori, then building something like this will be possible. We plan to implement the online version for a small social networking site for which data is readily available.

10 Conclusion

This paper focuses on finding the most influential spreader of a network in a distributed manner. We have shown how global methods fail to handle large scale datasets for their higher time and space complexity, which makes them infeasible to use on larger networks despite their capability of providing more accurate

results. We have also discussed the limitation in performance in case of local methods even-though they possess very marginal complexities. In order to address these limitations, we propose a new measure for ranking the users on the popular social network, Twitter based on their spreadability. Our proposed method incorporates user attributes that can be extracted from their corresponding twitter accounts with the traditional core value of the user on the network. We refer to our proposed method as User Attributed Core Decomposition (`UACD`). We have evaluated the performance of our approach on a Twitter dataset. We have shown that our approach outperforms other existing methods or at least provides a similar or better performance. To be specific, `UACD` can offer $\mathbf{12.4}\%$ more accurate result (on an average) in $\mathbf{175}\times$ less time (on an average). In summary, our proposed method can provide an efficient result similar to a global method while keeping the time and space complexities lower just like a local approach. We have also provided a distributed implementation of our method and shown that in the distributed environment, it can handle a dataset with a number of vertices of $41.7M$, while all the existing methods fail to handle such large datasets due to their higher time and space complexities.

References

1. Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 international conference on web search and data mining*, pages 183–194. ACM, 2008.
2. Sara Ahajjam and Hassan Badir. Identification of influential spreaders in complex networks using hybridrank algorithm. *Scientific reports*, 8(1):11932, 2018.
3. Haldun Akoglu. User's guide to correlation coefficients. *Turkish journal of emergency medicine*, 18(3):91–93, 2018.
4. J Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in neural information processing systems*, pages 41–50, 2006.
5. José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. k-core decomposition: A tool for the visualization of large scale networks. *arXiv preprint cs/0504107*, 2005.
6. EC Amazon. Amazon. *See https://aws. amazon. com/ec2/(15 June 2018)*, 2006.
7. Roy M Anderson, Robert M May, and B Anderson. *Infectious diseases of humans: dynamics and control*, volume 28. Wiley Online Library, 1992.
8. Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.
9. Vladimir Batagelj and Matjaž Zaveršnik. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification*, 5(2):129–145, 2011.
10. Joseph B Bayer, Nicole B Ellison, Sarita Y Schoenebeck, and Emily B Falk. Sharing the small moments: ephemeral social interaction on snapchat. *Information, Communication & Society*, 19(7):956–977, 2016.
11. Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12, 2010.
12. Vandana Bhatia and Rinkle Rani. A parallel fuzzy clustering algorithm for large graphs using pregel. *Expert Systems with Applications*, 78:135–144, 2017.
13. Data Science Bootcamp. Understand jaccard index, jaccard similarity in minutes. [Online; accessed 17-October-2020].
14. Stephen P Borgatti. Centrality and aids. *Connections*, 18(1):112–114, 1995.
15. Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

16. Jean E Burgess. Youtube. *Oxford Bibliographies Online*, 2011.
17. Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. A model of internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences*, 104(27):11150–11154, 2007.
18. Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684, 2011.
19. Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, page 4. ACM, 2010.
20. Hing Kai Chan, Xiaojun Wang, Ewelina Lacka, and Min Zhang. A mixed-method approach to extracting the value of social media data. *Production and Operations Management*, 25(3):568–583, 2016.
21. Wei Chen, Laks VS Lakshmanan, and Carlos Castillo. Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4):1–177, 2013.
22. Xin Chen, Mihaela Vorvoreanu, and Krishna Madhavan. Mining social media data for understanding students' learning experiences. *IEEE Transactions on learning technologies*, 7(3):246–259, 2014.
23. Reuven Cohen, Shlomo Havlin, and Daniel ben Avraham. Efficient immunization strategies for computer networks and populations. *Phys. Rev. Lett.*, 91:247901, Dec 2003.
24. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
25. Andrew Disney. Social network analysis 101: centrality measures explained. [Online; accessed 17-October-2020].
26. Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Why rumors spread so quickly in social networks. *Communications of the ACM*, 55(6):70–75, 2012.
27. Sergey N Dorogovtsev, Alexander V Goltsev, and Jose Ferreira F Mendes. K-core organization of complex networks. *Physical review letters*, 96(4):040601, 2006.
28. Olfa Belkahla Driss, Sehl Mellouli, and Zeineb Trabelsi. From citizens to government policy-makers: Social media data analysis. *Government Information Quarterly*, 36(3):560–570, 2019.
29. Ayman Farahat, Thomas LoFaro, Joel C Miller, Gregory Rae, and Lesley A Ward. Authority rankings from hits, pagerank, and salsa: Existence, uniqueness, and effect of initialization. *SIAM Journal on Scientific Computing*, 27(4):1181–1201, 2006.
30. Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.
31. Kaiqun Fu, Rakesh Nune, and Jason X Tao. Social media data analysis for traffic incident detection and management. Technical report, 2015.
32. Yu-Hsiang Fu, Chung-Yuan Huang, and Chuen-Tsai Sun. Identifying super-spreader nodes in complex networks. *Mathematical Problems in Engineering*, 2015, 2015.
33. Laura Garton, Caroline Haythornthwaite, and Barry Wellman. Studying online social networks. *Journal of computer-mediated communication*, 3(1):JCMC313, 1997.
34. Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.
35. Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28, 2013.
36. Aric Hagberg, Pieter Swart, and Dan Schult. NetworkX: a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. [Online; accessed 07-December-2019].
37. Minyang Han, Khuzaima Daudjee, Khaled Ammar, M Tamer Özsu, Xingfang Wang, and Tianqi Jin. An experimental comparison of pregel-like graph processing systems. *Proceedings of the VLDB Endowment*, 7(12):1047–1058, 2014.
38. JAP Heesterbeek. *Mathematical epidemiology of infectious diseases: model building, analysis and interpretation*, volume 5. John Wiley & Sons, 2000.
39. Nathan O Hodas and Kristina Lerman. The simple rules of social contagion. *Scientific reports*, 4:4343, 2014.
40. John Hopcroft, Tiancheng Lou, and Jie Tang. Who will follow you back?: reciprocal relationship prediction. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1137–1146. ACM, 2011.
41. Yuheng Hu, Lydia Manikonda, and Subbarao Kambhampati. What we instagram: A first analysis of instagram photo content and user types. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, 2014.

42. K Viswanathan Iyer et al. All-pairs shortest-paths problem for unweighted graphs in o (n2 log n) time. *International Journal of Computer and Information Engineering*, 3(2):320–326, 2009.
43. Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.
44. Matt J Keeling and Pejman Rohani. *Modeling infectious diseases in humans and animals*. Princeton University Press, 2008.
45. Wissam Khaouid, Marina Barsky, Venkatesh Srinivasan, and Alex Thomo. K-core decomposition of large networks on a single pc. *Proceedings of the VLDB Endowment*, 9(1):13–23, 2015.
46. Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888–893, 2010.
47. Konstantin Klemm, M Ángeles Serrano, Víctor M Eguíluz, and Maxi San Miguel. A measure of individual role in collective dynamics. *Scientific reports*, 2:292, 2012.
48. Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600, 2010.
49. Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.
50. Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.
51. Qian Li, Tao Zhou, Linyuan Lü, and Duanbing Chen. Identifying influential spreaders by weighted leaderrank. *Physica A: Statistical Mechanics and its Applications*, 404:47–55, 2014.
52. Jian-Guo Liu, Jian-Hong Lin, Qiang Guo, and Tao Zhou. Locating influential nodes via dynamics-sensitive centrality. *Scientific reports*, 6:21380, 2016.
53. Yang Liu and Yi-Fang Brook Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
54. Ying Liu, Ming Tang, Tao Zhou, and Younghae Do. Improving the accuracy of the k-shell method by removing redundant links: From a perspective of spreading dynamics. *Scientific reports*, 5:13172, 2015.
55. Tiancheng Lou, Jie Tang, John Hopcroft, Zhanpeng Fang, and Xiaowen Ding. Learning to predict reciprocity and triadic closure in social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(2):5, 2013.
56. Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics reports*, 519(1):1–49, 2012.
57. Linyuan Lü, Yi-Cheng Zhang, Chi Ho Yeung, and Tao Zhou. Leaders in social networks, the delicious case. *PloS one*, 6(6):e21202, 2011.
58. Vijay Mahajan. Innovation diffusion. *Wiley International Encyclopedia of Marketing*, 2010.
59. Kevin Makice. *Twitter API: Up and running: Learn how to build applications with the Twitter API.* " O'Reilly Media, Inc.", 2009.
60. Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.
61. C Martella. Apache giraph: Distributed graph processing in the cloud, 2012.
62. Claudio Martella, Roman Shaposhnik, Dionysios Logothetis, and Steve Harenberg. *Practical Graph Analytics with Apache Giraph*. Springer, 2015.
63. Nicole Martin. How Social Media Has Changed How We Consume News. [Online; accessed 15-June-2020].
64. Matthew L Massie, Brent N Chun, and David E Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.
65. Joel C Miller and Tony Ting. Eon (epidemics on networks): a fast, flexible python package for simulation, analytic approximation, and analysis of epidemics on networks. *arXiv preprint arXiv:2001.02436*, 2020.
66. Alberto Montresor, Francesco De Pellegrini, and Daniele Miorandi. Distributed k-core decomposition. *IEEE Transactions on parallel and distributed systems*, 24(2):288–300, 2013.

67. Ashwini Nadkarni and Stefan G Hofmann. Why do people use facebook? *Personality and individual differences*, 52(3):243–249, 2012.
68. Mark EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.
69. Mark EJ Newman. *Networks: An Introduction*. Oxford University Press, 2010.
70. Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the national academy of sciences*, 99(suppl 1):2566–2572, 2002.
71. Gottfried E Noether. Why kendall tau? *Teaching Statistics*, 3(2):41–43, 1981.
72. Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. In *International workshop on frontiers in algorithmics*, pages 186–195. Springer, 2008.
73. Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social networks*, 32(3):245–251, 2010.
74. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
75. Aditya Pal and Scott Counts. Identifying topical authorities in microblogs. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 45–54. ACM, 2011.
76. Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.
77. Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical review E*, 65(3):036104, 2002.
78. Daniel M Romero, Wojciech Galuba, Sitaram Asur, and Bernardo A Huberman. Influence and passivity in social media. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 18–33. Springer, 2011.
79. Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.
80. Stephen B Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.
81. Shisheng Shang and Kai Hwang. Distributed hardwired barrier synchronization for scalable multiprocessor clusters. *IEEE Transactions on Parallel and Distributed Systems*, 6(6):591–605, 1995.
82. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.
83. David Smith, Lang Moore, et al. The sir model for spread of disease: the differential equation model. *Loci.(originally Convergence.) https://www.maa. org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model*, 2004.
84. Twitter Developer Team. Twitter api. [Online; accessed 1-March-2021].
85. Alex Thomo and Fangming Liu. Computation of k-core decomposition on giraph. *arXiv preprint arXiv:1705.03603*, 2017.
86. Zhixiao Wang, Ya Zhao, Jingke Xi, and Changjiang Du. Fast ranking influential nodes in complex networks using a k-shell iteration factor. *Physica A: Statistical Mechanics and its Applications*, 461:171–181, 2016.
87. Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.