INFLUENTIAL NODE IN ONLINE SOCIAL NETWORKS USING SPARK

[1]Aishwarya T, [2]Suprena Mathew, [3]Dr. S.Subburam, [4]Kapila Vani R.K.
[1]Student, [2]Student, [3]Professor, [4]Assistant Professor
[1]Department of Computer Science and Engineering,
[1]Prince Shri Venkateshwara Padmavathy Engineering College, Chennai, India

Abstract : Social Networks play an important role in each and everyone's life in this world. Every individual spends most of his time in Social Networks each day to gain and provide knowledge. A particular individual is said to be the influential user if he has the maximum number of followers. To find the influential node in each domain dynamically we use the Apache Spark framework for the processing of the data dynamically and the algorithm we use is the UBI (Upper Bound Greedy Interchange) Algorithm. This helps to find the influential node in a domain at the particular instance of time. The performance of the system is improved and the scope of viral marketing is increased. We use Apache Spark, a Real-time stream processing framework of Big Data to process the data in real time. Hence we find the most influential user in the online social networks in each domain dynamically at that particular instance.

***IndexTerms* - Apache Kafka, Apache Spark, Big Data, Greedy Interchange Algorithm, Influential Node in OSNs**

I. INTRODUCTION

The structure of the social network evolves along with the strength of the influence between the people. At each and every point of time people influence each other in many ways and this strength of influence is used in various fields. This requires the tracking of the nodes dynamically for the tracking to be efficient. Targeting different nodes at different points of time becomes very essential for viral marketing to be success. When a node is said to be influential, it refers to the number of nodes that it is attached to. Considering the social networks such as Twitter and Facebook, it is the number of followers or friends that a particular user has at that moment. The strength of the influence also keeps changing every moment; we are more influenced by the friends whom we contact more frequently and regularly [1].

With the development of web technology, the social media network is evolving at a greater rate. With the users spending most of their time in social networks it becomes easy for viral marketing. The users may interact with other users or vice versa. Researchers have already made some research to find the influential node and provide user ranking based on the number of followers the user has. Twitter Ranking algorithm is one such algorithm, which provides the ranking for the Twitter users based on their interaction with the others. Twitter Rank is based on PageRank algorithm which ranks the users based on their influence [2]. Here we have to tackle two major problems, one is influence computation and the other is the dynamics in the network. These two are the challenges that have to be overcome in order to provide a suitable and efficient solution for the dynamic influential node tracking in online social networks. Influential node is nothing but finding the leader in the network. The one with maximum number of votes from his fans will be usually the leader. Similarly, in Twitter, the user with maximum number of followers is the leader with the consideration of the domain to which he belongs.

Motivated by such technical scenarios and also the challenges we have proposed a method to dynamically track the influential node in the Online Social Networks. We use an efficient algorithm, that is, the Upper Bound Greedy Interchange (UBI) algorithm, to find the Influential Node in the network along with the Big Data framework, Apache Spark, which is a real-time stream processing engine. This algorithm involves both the Greedy and Interchange algorithms in which the Influential node is found and interchanged with the already existing node of the network [3]. Also we have added an extra benefit by finding the influential node based on the domain the user is interested in. For example, a user is interested in Entertainment domain and he wants to know and follow the influential node in that particular domain for regular updates. This also helps the user to be more specific in which domain he is interested.

II. BACKGROUND AND RELATED WORKS

Big Data as the name suggests, relates to the huge amount of data that is generated in everyday life. Each and every day people from all over the world contact each other and get new friends on the social networking sites and also they are able to share all the information with their friends by setting up privacy if needed. How are these data managed? How are they stored and processed? These questions are easy to answer if one goes in depth and finds the realities of Big Data and performs experiments to make it more simple and efficient. Since the amount of data is huge to be processed, it takes enormous amount of time to process that data. Usually this happens statically and takes hours or even days to process that much amount of data that has been collected over an interval of time. Many algorithms have been used to find the influential node statically.

A Shapely Value Based Approach To Discover Influential Nodes in Social Networks [4] where the Target Set Selection problems such as top node problem and coverage problem. In this Shapely algorithm it focuses only on target set of collection nodes .This shapely algorithm uses greedy approach to find the influential node. In greedy algorithm once the influential node is found then changes will not be made after the node is finalized. After some particular time even after finding the node the stable matching or exchanging of the node is not possible .And calculating the node takes more time.

Drawback in this approach is weak stable matching. The influential node that is found is difficult to match to its exact node, and it is not capable of dynamically exchanging of nodes.

Following which the another algorithm was proposed An Upper Bound Based Approach To Discover Influential Nodes in Social Networks[5].It used upper bound lazy algorithm in which the speed was increased was 2-3 times. This algorithm also uses greedy approach but the influential node is found at a faster rate. The max function is given by arg max(u) $\in {}^{\boldsymbol{v}}/_{\boldsymbol{S}}\,(\boldsymbol{\sigma}(\boldsymbol{s}\ \cup \{\boldsymbol{w}\}) - \boldsymbol{\sigma}_{\mathbf{1}(\boldsymbol{s})})$. In this algorithm it independently checks the max function for each node and returns the result the time used for calculating the node is comparatively faster than that [4].The node with minimum threshold is easy to detect.

The drawback of this algorithm mainly focused on the independent set of nodes and it had low linear threshold.

To overcome the independent set problem the algorithm PPRank: Economically Selecting Initial Users for Influence Maximization in Social Networks [6] was introduced in which maximization of diffusion process and economically collect the seeds. In PPRank it uses a PPRank framework to find the influential node. Based on the internet structures on which it is made to find the influential node only on that it works. The speed has reduced to about 2-3 times of the previous algorithm. In PPRank the diffusion nodes are also taken into calculation while finding the influential node.

The drawback assigns a score board based on the internet structure.

III. Existing System

The current system deals with finding the influential nodes statically, that is, to find the influential nodes over a range of time. For example, the data about the users are collected and after a week the user who had maximum number of followers in that week is titled to be the most influential node of the week. This system is not done dynamically and hence we are in search of an implementation which will actually make it possible to find the most influential node in the network dynamically.

The existing system for dynamically finding the nodes deals with the simple algorithm for mobile networks and hepTh and hepPh networks. It has not been applied to the online social networks and this system has been applied for the overall network without any distinction between the domains of the network.

The major drawbacks of the existing system are that they are either statically used to find the influential nodes or if used for finding influential nodes dynamically they are slower than real time processing, that is, they are near real time. This is due to large amount of processing time that is taken to process the amount of data.

IV. Proposed System

The proposed system is the improvement to dynamically find the nodes or the users who are the most influential in the network at that particular moment of time. This can be done using various frameworks and we have chosen Apache Spark for this purpose. Apache Spark is a Big Data framework that deals with the real time streaming and processing of the data. This improves the finding of the most influential node at any instant with the more efficient algorithm. We have used the greedy approach using Apache Spark for the better performance of the system. In our proposed system, we mainly focus on Twitter which is one of the most popular social networking websites and consists of millions of nodes. We have added an extra filtering for finding the influential user in the network in each domain. For example, a person who is the most influential generally would be a playback superstar or someone related to entertainment field and it would be hectic to find the most influential node in the domain in which a person needs information on. So in our system, if a person wants to follow the most influential in any domain in which he is interested in he can find and follow that person or the user. This makes the job easy!

For a dynamic social network, to identify the influential node within a short span of time and to retrieve the results in seconds is not likely to produce it. But with the help of the framework Apache Spark and Apache Kafka we are able to find the influential node in a short span of time. By using this framework the performance is shown in graph without any difficulty.

Table 4.1: Notations Used and Its Description

Notation	Description
$\partial_{u,vs}(S)$	The replacing of gain from vs to u
$\acute{\partial}_{u,vs}(S)$	The upper bound of replacing gain $\boldsymbol{\partial}_{\boldsymbol{u},\boldsymbol{vs}(\boldsymbol{S})}$
S^t	The seed set at time t
α(S)	The expectation of the nodes influenced by S

ϑ_s(T)	The marginal gain and the probability of the node
Q(T)	The upper bound node.
AQ(S)	The probability that the v is activated exactly after the seed set under i.
AQ$v, i^{(S\|T)}$	The probability that the node is activated but without the help from the node T.

Based on the above framework to find the most influential node we are using UBI (Upper Bound Interchange)algorithm, in which we find the seed set that maximizes already found graph $\boldsymbol{G^{t+1}}$ from the ground ,we start with $\boldsymbol{S^t}$ to improve the influence coverage. Our algorithm first will identify the maximum node based on the network currently then after sometime if it founds some other node more influential than other it is replaces already found node. More detailed explanation about the upper bound interchange algorithm is explained in further section. The UBI algorithm and how it is used is discussed.

```
Algorithm 1: Exchange with neighbouring nodes
1: Calculate "d(x, xi)" i =1, 2,  n; where d denotes the  distance between the points.
2: Arrange the calculated n  distances in non-decreasing order.
3: Let k be a +ve integer, take the first k distances from this sorted list.
4: Find those k-points corresponding to these k-distances.
5: Let ki denotes the number of points belonging to the ith class among k points i.e. $k \geq 0$
6: If ki >kj $\forall i \neq j$ then put x in class i.
```

```
Algorithm 2: UBI(G =(V,E),S)
1: Compute $\partial_{v,vs}(S)$  For  $u \in V - S, u_s \in S$
2: for i=1 to |S| do
3: $u_s^* = \arg\max_{us \in S} \{\partial_{us}(S)\}$
4: $S \leftarrow$ Interchange$(G, S, u_s^*, \partial_{us}(S))$
5: Update $\partial_{u,us}$ (S) for any $u \epsilon V - S \in S$ according to the interchange result
6: end for
7: output S
```

4.1 Node Replacement Gain For Upper Bound

In this section, we illustrate the only mysterious part in our UBI algorithm, namely the computation of the upper bound of the replacement gain $\boldsymbol{\partial_{u,vs}}$. Zhou etal. first use the upper bound on influence function to accelerate the greedy algorithm in influential seeds selection. Following their methodology, we propose a tighter upper bound on the replacement gain by excluding the influence along paths, which include incoming edges to the seed set.

Basically, our task is to compute an upper bound on $\boldsymbol{\partial_{u,vs}}$ for any u $\in$ V - S in order to accelerate the Interchange Heuristic subroutine. We have

$$\boldsymbol{\partial_{u,vs}}(S) = \boldsymbol{\vartheta}(S\text{-vs} + u) - \boldsymbol{\vartheta}(S) \tag{4.1}$$

$$= \boldsymbol{\tau_U}(S\text{-vs}) - \boldsymbol{\tau_{vs}}(S - vs) \tag{4.2}$$

where$\boldsymbol{\partial_s}$(T)=$\boldsymbol{\vartheta}$ (S +T) – $\boldsymbol{\vartheta}$(T) is the marginal gain by adding set S to the existing node set T. The major task is to provide an upper bound on the first term $\boldsymbol{\partial_v}$(S-vs) and a lower bound on the second term $\boldsymbol{\partial_{vs}}$(S-vs). In the next two sections we will provide the computation of upper bound and the lower bound of the marginal gain.

4.2 Computation Of the Upper Bound

In this ,the complete illustration of how the upper bound is computed on the marginal gain $\boldsymbol{\partial s}^{\,(\boldsymbol{T})}$ .The probability of the node is given by A$\boldsymbol{P_{v,i}}$(S) which is associated with the node v is activated exactly at the step i in the seed set S.The step is being used in order to achieve a tighter bound of user probability that is instead of A$\boldsymbol{P_{v,i}}$(S) to get AP$\boldsymbol{v, i}^{(\boldsymbol{S}|\boldsymbol{T})}$.This user probability AP$\boldsymbol{v, i}^{(\boldsymbol{S}|\boldsymbol{T})}$stands for all the node v being activated exactly after the step i .The propagation for the probability $\boldsymbol{p_{u,v}^{G(T)}}$ where G(T) refers to the graph to and where (T) is used to indicate a set of nodes if not included in the diffusion process. The diffusion process is used to show if some of the node T is being excluded. The propagation process of G(T) is given by,

$$\boldsymbol{Q_{u,v}^{G(T)}} = \begin{cases} \boldsymbol{0} & \boldsymbol{v \in T} \\ \boldsymbol{Q_{u,v}^{G}} & \boldsymbol{otherwise} \end{cases} \tag{4.3}$$

Then, AQ$\boldsymbol{v, i}^{(\boldsymbol{S}|\boldsymbol{T})}$. Can be defined as the probability that the node is activated exactly in each step of node i of the graph G(T).

A$\boldsymbol{Q_{v,i}}$(S) and AQ$\boldsymbol{v, i}^{(\boldsymbol{S}|\boldsymbol{T})}$ is given at beginning itself so any changes while removing the other incoming nodes like(T) will not affect the node.

4.3 Fast Update of the Node Found:

In the previous section how the node are calculated among a given set of seed S was discussed. Here when the nodes are constantly arriving at the network how will the UBI algorithm can will be given with the help of a probability matrix $\boldsymbol{RS}^{\,G(\boldsymbol{S}-\boldsymbol{T})}$.

Let $\Delta \boldsymbol{P}$ be the different propagation probability between the two graphs G and $\grave{\boldsymbol{G}}$ which is associated with the probability matrices P and $\grave{\boldsymbol{P}}$ The difference is given by $\Delta \boldsymbol{P}$=P-$\grave{\boldsymbol{P}}$.

The updated node is calculated by the function $\acute{\boldsymbol{U}}$ and it is given by,

$$\boldsymbol{K}^{\acute{G}}=(\text{I-}\acute{\boldsymbol{Q}}^{-\mathbf{1}}).1 \tag{4.4}$$
$$=(\text{I-Q-}\Delta \boldsymbol{Q})^{-\mathbf{1}}.1 \tag{4.5}$$
$$\sim(\text{I-}\boldsymbol{Q})^{-\mathbf{1}}.1 + (\Delta \boldsymbol{Q} + \Delta \boldsymbol{Q}.\boldsymbol{Q} + \boldsymbol{Q}.\Delta \boldsymbol{Q} + \Delta \boldsymbol{Q}.\Delta \boldsymbol{Q}).1 \tag{4.6}$$
$$=\boldsymbol{U}^{\boldsymbol{G}} + (\Delta \boldsymbol{Q} + \Delta \boldsymbol{Q}.\boldsymbol{Q} + \boldsymbol{Q}.\Delta \boldsymbol{Q} + \Delta \boldsymbol{Q} + \Delta \boldsymbol{Q}.\Delta \boldsymbol{Q}).1 \tag{4.7}$$

Below algorithm gives the updates only for upper bound and for the UBI+ where $\Delta = \{(\boldsymbol{u}, \boldsymbol{v}) | \Delta \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}} \neq \boldsymbol{0}\}$ and the updating algorithm for UBI and UBI+ is given in [4][5].

```
Algorithm 3: Update Bound For UBI(P,$\Delta \mathcal{P}$, $\mathcal{U}$)
1: Initial $\acute{Y} \leftarrow Y$
2:  for(i,j) $\in \Delta$ do
3: // For each node v in N that is not i
4: for v $\in N_i(i)$ do
5: $\acute{U}$ + =$P_{v,i} \Delta P_{i,j}$
6: end for
```

This algorithm is applied and a filter is used to filter out the domains of the most influential user, that is, out of the nodes that we have first we filter out the nodes based on their domain and then find the node with maximum number of followers, that is, the most influential user of that domain.

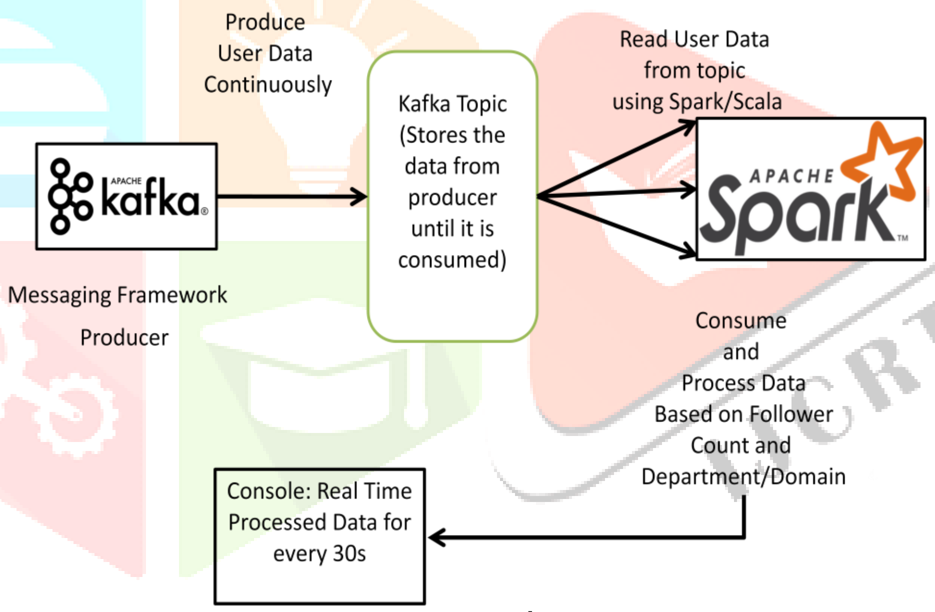
Produce
User Data
Continuously
Kafka Topic (Stores the data from producer until it is consumed)
Read User Data from topic using Spark/Scala
APACHE kafka
APACHE Spark
Messaging Framework
Producer
Consume and Process Data Based on Follower Count and Department/Domain
Console: Real Time Processed Data for every 30s

Figure 1: System Architecture

The Twitter data is produced as simulation by using the Apache Kafka Framework which is a fault tolerant Messaging Framework. The streamed data will be partitioned and stored in a Kafka Topic and maintained by a Kafka Broker and then it is read by the Apache Spark Framework for the Real time processing of the data. The processed data is then displayed in the console. Scala Programming Language is used for Spark and Java for Kafka.

The Apache Kafka uses publish-subscribe system. It is used for real-time data pipelines [7]. The Kafka producer produces the data and stores it in the Kafka Topic. Once the consumer program requests for data and subscribes to a topic it can consume as much amount of data that has been produced. The Spark framework consumes the data and performs the processing steps required and then produces the output for every interval of time mentioned by the user in the program.

V. Results and Discussions

The proposed system is more efficient than the existing system in terms of performance. For every 30 seconds the system handles nearly 7000 to 8000 users and processes that much amount of information to find the most influential node. The graph of our system as compared to that of Hadoop is as given below.

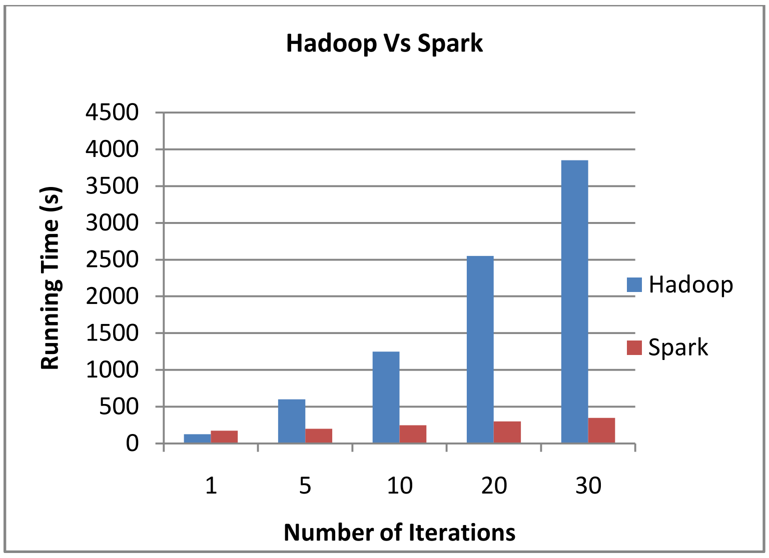

Figure 2: Hadoop versus Spark based on running time(s)

The x-axis denotes the number of iterations that spark gets the data for and Hadoop to read data and process each batch. The y-axis denotes the running time of each framework. Hadoop takes nearly 127s for each iteration whereas Spark takes 174s for the first iteration and then 6 seconds nearly for each rest of the iterations. This shows that Spark is faster than Hadoop and is more efficient in processing the data than Hadoop.

VI. Conclusion and Future Enhancements

We have proposed a system that helps to find the most influential nodes in Online Social Networks within each domain by real time stream processing of the data. This special feature enables us to find the influential node in each domain and helps us to make viral marketing easy. The future enhancements can be made by getting an efficient algorithm for finding the most influential node on the network without the need to go through all the node details. This can be done by simply finding the nodes in which alterations are made and then finding the most influential node from the nodes that have undergone changes rather than going through all the nodes again and again. This will make the system more efficient and improve its performance at a greater level.

References

[1] Guojie Song, Yuanhao Li, Xiaodong Chen, Xinran He, and Jie Tang, "Influential Node Tracking on Dynamic Social Network: An Interchange Greedy Approach", IEEE Transactions on Knowledge and Data Engineering, Vol. 29, No. 2, February 2017
[2] Richang Hong, Chuan He, Yong Ge, Meng Wang and Xindong Wu, "User Vitality Ranking and Prediction in Social Networking Services: a Dynamic Network Perspective", IEEE Transactions on Knowledge and Data Engineering, Vol. 29, Issue 6, 2017
[3] Yu Yang, Zhefeng Wang, Jian Pei and Enhong Chen, "Tracking Influential Individuals in Dynamic Networks", IEEE Transactions on Knowledge and Data Engineering, Vol. 29, Issue 11, 2017
[4] Ramasuri Narayanam and Yadati Narahari, "A Shapely Value-Based Approach to Discover Influential Nodes in Social Networks", IEEE Transacntions on Automation Science and Engineering, Vol. 8, No.1, January 2011
[5] Chuan Zhou, Peng Zhang, Jing Guo, Xingquan Zhu and Li Guo, "UBLF: An Upper Bound Based Approach to Discover Influential Nodes in Social Networks", IEEE 13th International Conference on Data Mining, 2013
[6] Yufeng Wang, Athanasios V. Vasilakos, Qun Jin and Jianhua Ma, "PPRank: Economically Selecting Initial Users for Influence Maximization in Social Networks", IEEE Systems Journal
[7] Navdeep Singh Gill, "Data Ingestion and Processing of Data for Big Data and IoT Solutions", March 03,2017, Available online: https://www.xenonstack.com/blog/data-engineering/ingestion-processing-data-for-big-data-iot-solutions
[8] K. Jung, W. Heo, and W. Chen, "IRIE: Scalable and robust influence maximization in social networks," in Proc. IEEE 12th Int. Conf. Data Mining, 2012, pp. 918–923.
[9] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2010, pp. 1029–1038.
[10] H. Zhuang, Y. Sun, J. Tang, J. Zhang, and X. Sun, "Influence maximization in dynamic social networks," in Proc. IEEE 13th Int. Conf. Data Mining, 2013, pp. 1313–1318
[11] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2009, pp. 199–208