

# Influential User Mining for Viral and Target Marketing in Social Network through PageRank and Bayesian Inference

by

Sebonti Mahnaz

16101177

Md. Ashikur Rahman Akash

16101098

Kamrun Nahar Ruchi

18201212

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering  
Brac University  
April 2020

© 2020. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

SEBONTI MAHNAZ

Sebonti Mahnaz  
16101177



Md. Ashikur Rahman Akash  
16101098



Kamrun Nahar Ruchi  
18201212

# Approval

The thesis/project titled “Influential User Mining for Viral and Target Marketing in Social Network through PageRank and Bayesian Inference” submitted by

1. Sebonti Mahnaz (16101177)
2. Md. Ashikur Rahman Akash (16101098)
3. Kamrun Nahar Ruchi (18201212)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on April 07, 2020.

## Examining Committee:

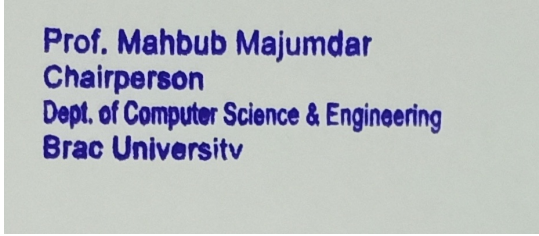
Supervisor:  
(Member)



---

Md. Golam Rabiul Alam, PhD  
Associate Professor  
Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)



**Prof. Mahbub Majumdar**  
**Chairperson**  
**Dept. of Computer Science & Engineering**  
**Brac University**

---

Mahbubul Alam Majumdar, PhD  
Professor and Chairperson  
Department of Computer Science and Engineering  
Brac University

## Abstract

Social networks have become one of the most important focuses for almost all Business strategies due to massive increase of potential sales using Viral marketing. The chief role played in these networks are the influential users, the actual market movers in any critical networks. Finding these users demands suitable approaches to take that oftentimes depends on the criteria of a social network along with the study of user behavior. Target market can be referred to as a community of people who are most likely to purchase some specific products and/or who have the highest odds of spreading the product. They are most likely to buy the product, somehow be in need of it or have a high record of being motivated by their idols, i.e. who they follow. They tend to have some common demo-graphical and behavioral characteristics (in that network) and thus the focus lies on what characteristics they share in that network which the business is interested in. Viral marketing is popular nowadays as it has its own business value. It can be termed as a strategy to find how customers spread messages about the product with other people in their social network, like the same way a virus spreads from one person to another. In this research proposal, we focus on target or viral marketing by studying efficient influential user mining procedures in twitter networks. We propose the famous PageRank algorithm and Bayesian Inference to find the best influential users in the network.

**Keywords:** influence; viral; maximization; social network; Bayesian; PageRank.

## **Dedication**

We want to dedicate our work to our parents, without whom we could never come so far in life.

## **Acknowledgement**

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor Dr. Md. Golam Rabiul Alam for his kind support and advice in our work. He helped us whenever we needed help.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

# Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Dedication	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	x
Nomenclature	xi
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Research Problem . . . . .	1
1.3 Research Objectives . . . . .	2
1.4 Scope and Limitations . . . . .	3
1.5 Thesis Report Outline . . . . .	3
<b>2 Literature Review</b>	<b>5</b>
2.1 Maximization spread through SN . . . . .	5
2.2 Cost optimized influence maximization in SN . . . . .	6
2.3 Different approaches for influence maximization . . . . .	6
2.4 Influence Maximization on Target Users . . . . .	8
2.5 Influence Maximization on Large-Scale SN . . . . .	9
2.6 Scalable influence maximization . . . . .	9
<b>3 Methodology</b>	<b>10</b>
3.1 Dataset . . . . .	11
3.1.1 Higgs Twitter Dataset . . . . .	11
3.1.2 Dataset Description . . . . .	11
3.2 Data Pre-processing . . . . .	12
3.3 Dataset Creation . . . . .	13
3.3.1 Data Extraction . . . . .	13

3.3.2	Data merging . . . . .	14
3.3.3	Creating A New File . . . . .	15
3.4	Problem Formulation . . . . .	15
3.4.1	Kempe’s Greedy Algorithm . . . . .	15
3.4.2	PageRank Algorithm . . . . .	16
3.4.3	Bayesian Inference . . . . .	16
3.5	Algorithms . . . . .	17
3.5.1	Kempe Algorithm . . . . .	19
3.5.2	PageRank Algorithm . . . . .	22
3.5.3	Bayesian Inference . . . . .	24
3.6	Implementation . . . . .	31
3.6.1	Kempe Algorithm . . . . .	31
3.6.2	PageRank Algorithm . . . . .	34
3.6.3	Bayesian Inference . . . . .	37
<b>4</b>	<b>Results and Analysis</b>	<b>40</b>
4.1	Kempe Algorithm . . . . .	40
4.2	PageRank Algorithm . . . . .	41
4.3	Bayesian Inference . . . . .	43
4.4	Comparison . . . . .	50
<b>5</b>	<b>Conclusion and Future Work</b>	<b>53</b>
	<b>References</b>	<b>55</b>



# List of Figures

1.1	Representation of nodes from a Social network . . . . .	3
3.1	Visual representation of Higgs twitter-Retweet . . . . .	12
3.2	Visual representation of Higgs twitter-Reply . . . . .	12
3.3	Visual representation of Higgs twitter-Mention . . . . .	13
3.4	Snap of Higgs-Reply edgelist File . . . . .	14
3.5	Snap of Higgs-follow edgelist File . . . . .	14
3.6	Snap of the new .csv File . . . . .	15
3.7	Workflow Diagram for Kempe Algorithm . . . . .	18
3.8	Workflow Diagram for PageRank Algorithm . . . . .	18
3.9	Workflow Diagram for Bayesian Inference . . . . .	19
3.10	Representation of an example in Kempe's Context-01 . . . . .	20
3.11	Representation of an example in Kempe's Context-02 . . . . .	21
3.12	Kempe Greedy Algorithm . . . . .	22
3.13	Representation of an example in PageRank's Context . . . . .	23
3.14	Representation of an example in PageRank's Context-02 . . . . .	23
3.15	Sample Dataset . . . . .	27
3.16	Interaction Probability of the Sample Data . . . . .	28
3.17	Influenced weight of the Sample Data . . . . .	28
3.18	Blanket of Node0 . . . . .	29
3.19	Blanket of Node1 . . . . .	29
3.20	Result of the Sample Data . . . . .	31
3.21	Seed Set of the Sample Data . . . . .	31
3.22	Independent Cascade Function . . . . .	32
3.23	A function to Find Neighbour . . . . .	32
3.24	Kempe-Greedy Function for Algorithm . . . . .	33
3.25	Main Handler of Kempe-Greedy Algorithm . . . . .	34
3.26	Making A Graph From The File . . . . .	34
3.27	Directed Graph of The New File . . . . .	35
3.28	Sorting The Dictionary File . . . . .	36
3.29	Code of Finding Seed Set with it's PageRank value . . . . .	36
3.30	Pseudocode for Bayesian Inferenece . . . . .	37
3.31	Blanket A . . . . .	38
4.1	Seed Set For Kempe . . . . .	41
4.2	Child response Probabilty of Kempe . . . . .	41
4.3	Graph of Kempe . . . . .	41
4.4	Seed Set For PageRank Algorithm . . . . .	42
4.5	Output Graph For Pagerank Algorithm . . . . .	42

4.6	Weighted probability of Node78 . . . . .	43
4.7	Weighted probability of Node769 . . . . .	43
4.8	Weighted probability of Node4665 . . . . .	44
4.9	Weighted probability of Node799 . . . . .	44
4.10	Weighted probability of Node640 . . . . .	45
4.11	Weighted Probabilty of Some of the Nodes . . . . .	45
4.12	Blanket of node640 . . . . .	46
4.13	Blanket of node40 . . . . .	46
4.14	Blanket of node383 . . . . .	46
4.15	Blanket of node138 . . . . .	47
4.16	Blanket of node1026 . . . . .	47
4.17	Blanket of node77 . . . . .	47
4.18	Blanket of node1134 . . . . .	48
4.19	Blanket of node765 . . . . .	48
4.20	Blanket of node715 . . . . .	48
4.21	Seedset Found from Bayesian Inference . . . . .	49
4.22	PageRank seed set's child response probability . . . . .	50
4.23	Kempe seed set's child response probability . . . . .	51
4.24	Bayesian seed set's child response probability . . . . .	52
4.25	% of zero response probability child-among all 3 models . . . . .	52

# List of Tables

3.1	Attributes of Higgs-Twitter Dataset . . . . .	12
-----	---	----

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

*CELF* Cost Effective Lazy Forward

*ICM* Independent Cascade Model

*IDCF* Information Diffusion Cost Function

*IM* Influence Maximization

*IP* Interaction Probability

*IW* Influential Weight

*LDAG* Local Directed Acyclic Graph

*LTM* Linear Threshold Model

*MIA* Maximum Influence Arborescence

*NP – Hard* Non-Deterministic Polynomial-Time Hardness

*SN* Social Network

# Chapter 1

## Introduction

### 1.1 Background

Right now, mechanical progression, online networking is endlessly spreading among us to speak with one another. Individuals are utilizing these interpersonal interaction site on account of their accessibility, flexibility and easy to understand highlights. Presently a day we have practically 7.7 billion individuals everywhere throughout the world among them practically 3.484 billion individuals are dynamic in internet-based life. From a research it shows that, in 2019 the normal time for utilizing online life is two hours and 23 minutes in a day for each dynamic client. So, any substance can spread through online networking rapidly as individuals are dynamic in these stages. Additionally, individuals from different culture, religion, sexual orientation, district or mindset can be found right now. We live in an era of technology and science; and among all the technological inventions social networking is the one that influence is the most in almost every aspect of our life. It affects our everyday choices. Basically, word of mouth, sharing meme on the social network, and viral marketing are all examples of how our social behavior can affected through social network. These influences happen in social networks via sets of nodes. These sets of nodes can easily reach the largest audience indirectly when propagating information for various aspects. People in various socials networking sites like Facebook, Twitter, Instagram gets influenced by some highly active nodes. Thus, take or change their decisions. Here, to detect the nodes which have the highest influence over the network is our main problem. Previously, there are no solutions available that is learning based and detect the nodes all by themselves; those solutions were based on a statistical report or databases; but our solution will be learning based. To detect the most influencing nodes, we will choose the approach that will lead us to the most precise results efficiently. For this we have chosen PageRank and the Bayesian Inference approach to detect the influenced nodes. Because the by using PageRank and Bayesian inference we can get many precise results without using much data.

### 1.2 Research Problem

Online networking is where individuals of everywhere throughout the world assemble to speak with each other. So that, every type of people can communicate over here, and they can search here their daily necessities. Moreover, on recent days most of the people spend their leisure time on social media or online whatever we say. People

not only spend their time, but they buy products from here getting influenced by some influencers. Those are active people, or we can say that those are active nodes. And this is one of the most common problem of online that to find out those active users overall users. To find out by which nodes are mainly reason to influence other nodes. Influence maximization is one of the basics problems on social media and its an optimization problem. Influence maximization problem is the kind of problem which can appoint a subset of  $k$  clients as (seed hubs) in a chart which could help the spread of impact by amplifying the normal number of affected clients. So here some methods living those influential nodes for maximizing the issue. Moreover, here the system is learning based whereas some previous publications are not learning based. Also, the uses of PageRank and Bayesian inference can make this research paper different from others.

### 1.3 Research Objectives

There are billions of individuals use social media in their daily basis. The amount of users is rising day by day. People like more spending their leisure here. Here in social site, there are some active users who are followed by other users. They are mainly are the foremost active users. But it is too hard to search out those users. During this research paper, the main purpose is to detect those nodes from all other nodes. Moreover, to spot those nodes is basic purpose as if it's possible to detect those nodes only then it is possible to succeed in main goals. Besides, the main goal is to find the most active users from all other users on social media by applying some useful methods. These methods help to know there are some nodes which are most active, some nodes which are active and some nodes which are not active in social site. This research paper proposes PageRank and Bayesian inference method which is not used previously. Based on their impact weights and using the dataset, the subset will form by utilizing PageRank calculation. Then apply Bayesian inference in these subsets to decide the most active nodes in social site. Through Bayesian inference the outcome will be more precise. The main objectives of this research work are:

- We have to detect the active nodes in the networks that have the most influence over people. Off all other approaches to detect the nodes we choose PageRank and Bayesian inference method.
- We have to make out the system learning based. Previous solutions are not learning based.

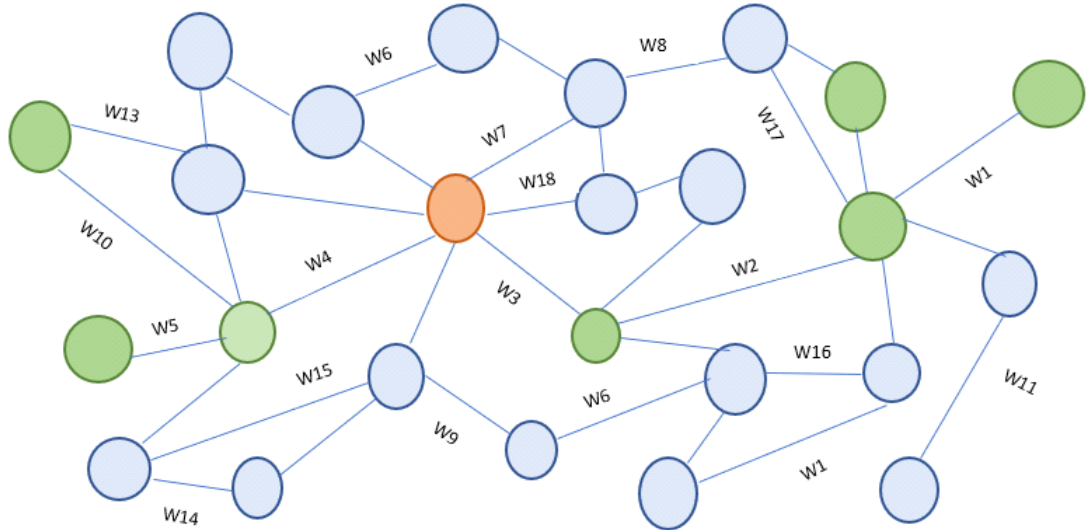


Figure 1.1: Representation of nodes from a Social network

Figure 1.1 Here represents the node of social network. Here, the nodes are divided into active, most active and inactive category. And the edges are describing here the relationships or connectivity between the nodes. It shows that the “orange”-colored node connects mostly with all other nodes and it makes its most influential. Then the “green”-colored nodes are connected at most twice with other node, and these are active nodes and the “blue”-colored nodes are just normal inactive nodes.

## 1.4 Scope and Limitations

Through this thesis paper we have tried to discuss how identify the most influential nodes over the social site and those influential nodes are the most active users. To identify those users, we apply here some methods. But here we have some limitations. Like, in our thesis paper we used Twitter dataset, and it works in one directional way. Someone can see anyone’s tweet only if he follows that specific person over Twitter which means if anyone doesn’t follow then he cannot see any tweet. But if we see into Facebook then we will understand that it works in bi-directional way. So, here are the limitations of us, we can say.

## 1.5 Thesis Report Outline

The rest of the portion of this research paper contains as following. Chapter 2 describes the background study, and detailed literature review. It will describe the

existing works on these topics and different types of methods. Chapter 3 portrays the proper outlook of our dataset. This bit will contain the information about our datasets, procedures that we followed to pre-process our data to make it usable for further research. Chapter 4 contains the methods that we have used. It illustrates about our proposed model thoroughly as well as the already established ones. Chapter 5 gives us the result and analysis of our work. In chapter 6 we summarize our work and give a hint of what we can do to move this research further.



# Chapter 2

## Literature Review

Online life has been ending up being an incredible development since it was designed. In any case, as the quantity of clients is expanding day by day, it has become very hard to know that which nodes are active and which are not. For identifying most active nodes here we apply some methods like Bayesian inference and PageRank. There has been a research which is not that much broad in this specific field. A portion of the significant research works are given underneath.

### 2.1 Maximization spread through SN

Social network consists of nodes and ties. Nodes are the individual actors within the networks and ties are the connection between the actors. Influence maximization basically decides the influential users who maximize the profit over the network defined by the most number of nodes which will be activated by a given seed set. According to Kempe, Kleinberg, and Tardos, influence maximization is an optimization problem which is mainly NP-hard under Independent Cascade model and Linear Threshold model as well[1]. They proposed simple linear model where the answer to the optimization problem is obtained and it will be obtained by solving a system of linear equations. They also focused on related NP-hard models. NP-hard models that are studied within the social network's community, and procure the primary provable approximation guarantees for efficient in a number of general cases.

In [12], main propose is finding an optimal set of influence users. One main propose is to find an optimal set of most active user. However, it is hard to detect, use two models used Linear Threshold model and Independent Cascade model. Both the models are estimate in size and maximize the hardness. Besides, greedy algorithm used to select influencers. The main goal is to identify social influencers a good quality. So, here introduced new influence which combines influence indicators and it maximized influence. Here, take into account the reliability of each and every influence indicator and present distance-based process that helps to calculate the reliability of each indicator. The measure of this impact is used with an influence maximization model to elect some users which are actually able to maximize the influence.

## 2.2 Cost optimized influence maximization in SN

In this paper [15], The influence maximization problem means find to set seed nodes that can help to maximize the influence. Here, the method word of mouth used as marketing. Other models like an Independent Cascade model and Linear Threshold model are used here. Also, greedy algorithm and heuristic methods used to solve the problem efficiently. Target market focuses on split the market. In this paper, here mainly focused on both target market and diverse profits and propose labeled influence maximization problem. Moreover, in this paper some problems for labeled influence maximization discussed and to solve that three approaches are used which divided two categories. These are labeled new greedy, labeled discount. Also, the new greedy algorithm uses to improve the ability of general greedy algorithm. Moreover, the new greedy algorithm degree discount heuristic which finds effective seed nodes. The proposed maximum coverage system allowance for offline computing influence potential.

In this paper [17], Nowadays social site is the only way to communicate people and great research area. Influence maximization is one of the great areas for social network research. One effect word of mouth uses here. Influence maximization problem has many applications and maximum researches are based on those direction. But here discussed RIM problem to calculate opportunity cost. RIM works like opposite of influence maximization problem and for that R-RIM and RLT-RIM will be used for solving RIM problem. Some more models like a linear threshold, independent cascade, and cost-effective lazy forward. Here in this paper, also discussed multiple products and also estimate the cost. In graph here consider each vertex user and edge means relationship. The RIM problem estimates the opportunity cost. There are some challenges like to set stopping criteria to handle three BNC's and to handle insufficient influence. The RIM problem consider as NP-Hard because it reduced the RIM problem. The marginal cost is used here to find opportunity cost. MC simulation for evaluating performance and degree centrally technique also used here.

## 2.3 Different approaches for influence maximization

A comprehensive latent variable model is used here which captures community-level topic interest, item topic relevance and community membership distribution of each user. A collapsed Gibbs sampling algorithm which mainly uses for training the model. Then they deduct community to community and user to user influence strength. Finally, they introduce a community-based heuristic algorithm to mine influential nodes which chooses the most active nodes with a divide-and-conquer strategy [18].

In this paper [5], Influence maximization is one of common problem and word of mouth used for promotion. Here, two models used Linear Threshold model and Independent Cascade model. Both IC model and LT model is NP hard in influence

maximization. Greedy algorithm used here which picks the maximum marginal gain and adds it to seed set. But this algorithm has some drawbacks itself. One effective algorithm SIMPATH used under linear threshold model for addressing those drawbacks. Used two optimization which cut down the number of calls which happened in first iteration. Secondly, using look-ahead optimization that basically address the problem and keeps running time of the subsequent iterations small. Moreover, backtrack algorithm uses for computing all simple paths. Here, they use four real world data sets to evaluate the ultimate performance of SIMPATH.

The basic idea of influence maximization is to select influential users among users to maximize the total influencer. Here, presents that conduction model that unifies, generalizes and extends the existing non-progressive models which also includes non-progressive in Linear Threshold and Voter model. As in Linear Threshold model and Independent Cascade model can't capture reversibility of choices. So that, this new model presents in this paper. To solve the influence maximization problem with scalable established the sub modularity of influence spread. Also, three essential properties are working to maximize influence the work can done within very short time even if here used thousands of nodes [14].

In this paper [11], PageRank algorithm is implemented on signed social networks and then the integrated PageRank was imposed to study influence maximization. Main goal of this paper was to find a smaller subset of nodes for influencing the largest number of people. Classic voter model was used to determine positive and negative relationships between nodes. As the basic PageRank algorithm cannot compute in signed networks because there are edges both negatives and positive. That's why an Integrated PageRank algorithm is used. Because the social network is dynamic, that's why there's a timestamp. Certain calculations are used to measure the above-mentioned aspects and changes should be treated as norms not exception using temporal smoothness principle.

In this paper [13], here basically proposed about a bounded linear approach for influence computation and influence maximization. Their work basically divided into two parts: first one is- to give the description about the influence propagation by using a linear and traceable approach; second to maximize social influence by using Group PageRank. A set of the most influential nodes basically traced by linear and bound and both are scalable over the large-scale social networks. Previously Independent Cascade and Linear Threshold models require running Monte-Carlo simulation for a significant number of times. It's very time-consuming, also not capable for a large scale.

In this paper [9], Social influence on maximize viral marketing, and has been one of the most worked topics now. Social influence basically means the behavioral transformation of one person and it is caused by his/her relationship with people or network. User's impact is weighted based on their shopping behavior and the influence in the social media. By choosing the influential leaders there would be huge improvement in the marketing process, reduction in expenses and so on. In viral marketing there have mainly two modes of active and inactive. At the beginning, each neighbor is inactive. If every neighbors of the node choose the marketing

product then that node will change into active mode.

In this paper [19], Knapsack based solution under linear threshold model is also used to resolve reverse influence maximization problem efficiently and optimize the seeding cost. Influence maximization checks which nodes are activated by the seed nodes whereas, the reverse influence maximization look into by whom the seed nodes are activated. It has better seeding cost using the greedy optimization. Moreover, KRIM model uses linear threshold model in the reverse order for the node actuation process. Node activation procedure is concluded by the influence decay concept indicating the impact of influence is reduced with the hop distance from the influence node.

In this paper [10], an influence maximization problem is considered which might solve by the link activation under the Independent Cascade model. Unlike the previous papers, the decision variable that they can choose for maximizing the influence spread are the active links in the network. They used link activation for influence maximization under the Independent Cascade model. The goal of this paper is maximizing the final influence dissemination under a limited budget. To solve this problem, they propose a heuristic associated with a cost-degree coefficient. Furthermore, experimenting on a real network shows that their methodology works effectively and efficiently.

In this paper [6], they propound the labeled influence maximization problem, which main target is to discover a set of seed nodes. The maximum spread of influence can be triggered on the target customers in a labeled social network. Three algorithms are proposed to solve such labeled influence maximization problem. The label information is widely available over current social networking services. For solving the labeled influence maximization problem under the independent cascade model, they declared three approaches. Two methods, Labeled New Greedy and Labeled Degree Discount, are modified to consider the targeted labels and profits.

## 2.4 Influence Maximization on Target Users

In this paper [16], it determines about formulating algorithms that aim for maximizing influence unto target users over the social networks based on the IDCF and influence spreading paths. In diffusion information on social networks, diffusion cost is gained here. Information diffusion cost function (IDCF) considered as the number of times then a message is being expanded. It is a function of the quantity of steps in the graph that the message moves through and the normal number of times the data was shared at each progress with respect to the amount of influencers. Target clients are important in viral marketing in light of the fact that the data is diffused to them, which needs to reach them in limited time cycle and least diffusion cost as well. This problem goal is to identify the minimal seed set of influencers. It is figured under both the independent cascade model (ICM) and linear threshold model (LTM) whereby a node propagates an data based on a probability.

In this paper [4], they applied a set of algorithms, including general greedy, hill-climbing and centrality-based algorithms, on the real-world social network which

mainly identify the key users with better influence. They also proposed an approximation searching algorithm based on the heuristic's information from the above methods. In this paper, they choose to identify key users through the online social network of Epinions. On Epinions, users can give his review and rate the products which they have purchased or used. Firstly, they randomly choose  $m$  users from the network. Secondly, they applied centrality such as degree and betweenness. The two measures describe different dimensions of centrality in social networks. Degree centrality-based algorithm work well in all experiments in their model which is even better than the general greedy and hill-climbing in different points.

## 2.5 Influence Maximization on Large-Scale SN

In this paper [8], a novel frame that take advantage of the temporal dynamics of the network to choose an optimal subset of users which maximize the marketing influence over the network. Users are selected based on intrinsic value of user and connectivity value of the network. The intrinsic value mentios to the individual attributes of an individual user while the connectivity value calculates the network structured as a whole. Also, they discussed "Influence flow"- a function of live edges i.e. Probability of a user getting influenced is proportional to the number of live edges.

## 2.6 Scalable influence maximization

In this paper [3], Influence maximization is the problem which is basically finds a small set of most influential nodes over the social network, so that their total influence in the network is maximized. It has application on viral marketing and here companies promote by using word of mouth. Some social sites are also important platforms. Here, social network demonstrated as directed graph where  $V$  means edges and  $E$  means relationships. In Independent Cascade model, each edge has an activation probability and influence propagate by activated nodes and also active their nodes which are inactive. On the other hand, in Linear Threshold model there has weight on edge and vertex have threshold and vertex only active weighted sum of active neighbors crossed certain threshold. Moreover, a new heuristic algorithm used and it works faster than greedy algorithm.

In this paper [2], they used a new heuristic algorithm which is faster than the greedy algorithm and it is more scalable. Moreover, other heuristic algorithm does not provide satisfactory performance. They mainly focused on running time and influence spread of algorithm. Moreover, this new heuristic algorithm makes the scalable solution and it can work with large number of graphs and also performs well. The new heuristic algorithm outperforms far better than any other algorithm which is used in previous paper. The main idea of this algorithm is use local bore scene structure of each node to approximate the influence propagation. By using MIA model (maximum influence arborescence) this heuristic algorithm works.

# Chapter 3

## Methodology

With a purpose of making a model that will be able to detect the most influential set of nodes in a social network, we have come across several methods that can be used to do it. Social network is a large domain. In it there are hundred and thousands of nodes connected with each other having various types of interactions. That is why we shortened our research to only finding the influential nodes of a network based on some specific criteria.

Methodology can be referred as a course of action of procedures used in a particular region of study or activity. The system depicts the wide philosophical supporting to our picked investigating methods, including whether we are using subjective or quantitative procedures, or a fusion of both, and why. Sometime we get our expected outcome just following one methodology vis-a-vis one algorithm. Moreover, we might need to merge two or more methodologies in order to get the expected outcome. Now, which methodology to follow depends on several variables. First of all, the dataset and complexity of the dataset. If the dataset is larger, then we need to find a way that does the job in a comparatively less time. On the contrary, if the dataset is of adequate size we need to focus on accuracy.

Furthermore, methodologies do vary on parameters that are being used and the techniques of data pre-processing. We have partitioned our work into several sections to obtain the results accurately. They are as follows:

- Data set
- Data set pre-processing
- Data set extraction
- Problem Formulation
- Apply Kempe-Greedy Algorithm
- Apply PageRank Algorithm
- Apply Bayesian Inference Model

These subsections are described below:

## 3.1 Dataset

Data collection is the essential and most basic work is informational index assortment. Mainly this is the most challenging part during the whole research to collect all those data according to our research. So, it was hard for us to collect all the data. As our work mainly revolves around social media, so dataset collection from those social sites was the first impediment that we had to face because of the privacy issues. Also, it might be easier to collect data-sets from various non-authorized websites; but it would not fit our criteria. Two of the most famous social networking sites are Facebook and twitter. Unfortunately Facebook does allow researchers to use their data. On the other hand twitter does allow researchers to use their dataset. We collected Higgs-Twitter dataset from Stanford University. The dataset met our needs; though it needed some prepossessings to make it usable for us.

### 3.1.1 Higgs Twitter Dataset

The Higgs dataset has been made in the wake of checking all the spreading structures on Twitter beforehand, during and after the assertion of the disclosure of another particle with the features of the unobtrusive Higgs Boson on fourth July 2012. Here we use directional networks those are available and have been extracted by monitoring the user activities. In this research, we use Higgs Twitter dataset and we use 4 datasets here which are Higgs mention dataset, Higgs reply dataset, Higgs retweet dataset, Higgs social dataset. We use Higgs network dataset as it fulfills our requirements. By using Higgs dataset, we can modify it by our own way. And also, Higgs dataset matched our criteria by which we can fulfill our aim to research. The interactions that are present amongst the nodes can be labeled as RT(retweet),MT(mentions) or RP(reply).Each and every link between them is directed. They represent the connections between them.In this dataset, direction between every nodes depends on the application by and large. As such, somebody assembles a system of how data streams; at that point the bearing of RT must be turned around and the clients whom referenced in retweeted tweets are considered as mentions.

### 3.1.2 Dataset Description

From Stanford University's resources, we found out four data-sets; each containing one attribute of a node. They are- followers of a node, how many times they have been mentioned, how many retweets they have got and how many replies they got. The dataset we have now is quite illustrated. The dataset that have the followers number has 456626 nodes and 14855842 edges between them. This refers that these nodes have 14855842 number of relations between them. Similarly the Retweet Dataset, the Mention dataset and the Reply dataset have 256491 nodes and 328132 edges, 38918 nodes and 32523 edges and 116408 nodes and 150818 edges between them respectively. The nodes of these data-sets also contain cycles between them.

Name of the dataset	Number of Nodes	Number of Edges	Number of cycles
Follower	456626	14855842	83023401
Retweet	256491	328132	21172
Reply	38918	32523	244
Mention	116408	150818	23068

Table 3.1: Attributes of Higgs-Twitter Dataset

## 3.2 Data Pre-processing

Our goal is to find influential nodes in a social network. That's why we need a dataset that consists of every possible interactions between two nodes. Those interactions can be of any kind. For example, if we consider twitter, we can have some relationship between two nodes(here, a node means a user) like, "follows", "replies", "retweets", "likes", "mentions", "hit counts" etc. Each of the interactions can be used as a parameter for our research. Among all these parameters, in our existing dataset we have "follows", "mentions", "replies" and "retweets" interactions between them. Here are some visual representations of first 100 nodes our existing data.



Figure 3.1: Visual representation of Higgs twitter-Retweet



Figure 3.2: Visual representation of Higgs twitter-Reply



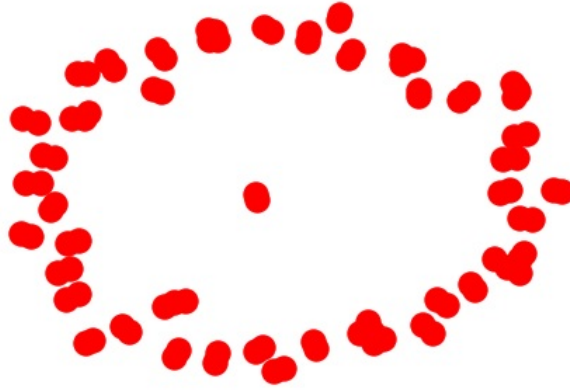


Figure 3.3: Visual representation of Higgs twitter-Mention

### 3.3 Dataset Creation

As the dataset was huge in number we had to break it down to 5000 nodes for all the interactions. That is why we needed to create our own dataset from the existing one. Because among the four portion of our dataset, the least amount was nearly 40,000 nodes and 32523 edges between them. That is why we decided to take 5000 nodes from each of the portion and make it feasible for us to proceed further. To do so, we broke down our job into two parts:

1. Data Extraction
2. Data Merging
3. Creating a New File

#### 3.3.1 Data Extraction

From these huge number of nodes we decided to take 5000 nodes for our research. For that we have made a program that can take all four files and a specific number and that will create new four files with exact amount of nodes that we have already mentioned. Our four files were named as HiggsRT, HiggsRP, HiggsFl and HiggsMt. These four files and hundreds and thousands of node. Here is a snap of how our data looked like in an .edgelist file.

```

161345 8614 1
428368 11792 1
77904 10701 1
124554 286277 1
194873 194873 1
341375 16460 1
436133 220 1
274148 274149 1
12866 22252 1
425029 35248 1

```

Figure 3.4: Snap of Higgs-Reply edgelist File

Here, the data represents who has replied to whom. For example node 161345 has replied to 8614; and the remaining 1 is the edge value between them is 1, which is uniformly given to all the nodes of all the other portions of the dataset. In the next step we merged these three files.

```

1 11
1 12
1 13
1 14
1 15
1 16
1 17
1 18
1 19
1 20
1 21
1 22
1 23

```

Figure 3.5: Snap of Higgs-follow edgelist File

Here, the data shades the light on which node has connection with which node in terms of "follow" relation. For example, node 1 has relation with all the corresponding nodes. That means node 1 has this much number of followers. We use this data to calculate their followers and their influence weight in later stages.

### 3.3.2 Data merging

Then we took the four files that were extracted in the previous step and merged them altogether. Here, we needed to keep in mind that, there will always be a possibility of a node being present in "reply", "retweet" and "mention" portion simultaneously. Also it may be present in at least one of those. In addition if a node is not present in any of these three portion, we look for it in the "follows" portion whether it is

there or not. To remove this problem, we had to look at the first three portion all together to be accurate in our findings. we merge them together in a new listArray named Allnodes[].

### 3.3.3 Creating A New File

We check all the nodes in Allnodes with the "follow" portion. After that we amalgamated each nodes with its follower number and the three attributes that we have extracted already. Then a new .csv file was created. Here is a snap of the newly created csv file.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Node	Follower	RT10	RP10	MT10	RT1001	RP1001	MT1001	RT1002	RP1002	MT1002	RT1003	RP1003	MT1003	RT1006	RP1006
2	10	447														
3	1001	16														
4	1002	74														
5	1003	172														
6	1006	21														
7	1007	11														
8	1009	136				1										
9	101	15														
10	1011	24								1						
11	1013	2														
12	1014	124														
13	1016	148														
14	1018	18						1								
15	1021	33														
16	1022	26														
17	1023	32														
18	1025	112														
19	1026	68														

Figure 3.6: Snap of the new .csv File

It illustrates that, node 1009 has a retweet from node 1001, node 1011 has a reply from node 1002 and node 1018 has been mentioned by node 1001. This .csv file also shed lights on which node has how many followers. We needed all these information to move forward. By creating this new file, we got all what we needed to proceed further.

## 3.4 Problem Formulation

### 3.4.1 Kempe's Greedy Algorithm

The social network is represented by a directed graph  $G$  whereas  $u \rightarrow v$  refers the existence of an interaction from node  $U$  to node  $V$ . It can be anything, such as retweets, mentions etc. According to Kempe's greedy approach [12], for the maximization of influence from a node depends on whether a node is active or not. Also, whether a node can activate its neighboring nodes or not. Here it is first focused on the case where a node can be activated from being inactive. The procedure will look somewhat as follows from the perspective of an initially inactive node  $V$ : as time goes by, more and more of  $V$ 's neighboring nodes gets activated and this may cause  $V$  to get activated.

In the Independent Cascade Model, we start with an initial set of active nodes, and the procedure gets going as follows. When node  $V$  first becomes active in  $step_t$ ; it is given a single chance to activate each currently inactive neighboring nodes  $W$ . Then it can be succeeded with a probability of  $P_v, w$  which is a parameter of the system and independent till now. This implies that, if  $W$  has multiple newly activated

neighboring nodes, the attempts to make them activated is done in an arbitrary sequential method. If  $V$  succeeds, then in the next step, i.e. in  $step_t + 1$ ,  $W$  will become active. But if  $V$  does not get activated, it can never activate its neighboring nodes in  $W$  in corresponding steps. This continues until there is no activation left. As a result, the problem of finding a best seed set in IC model is NP-hard to approximate the most influential set to within better than a factor  $11/e$  [12].

If we consider we have an Independent Cascade model  $\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$  and a constant integer  $K$ , find a seed set  $\phi_0 \subseteq \mathcal{V}$  -of  $K$ -numbers, such that the final influence spread ( $\phi_0$ ) is maximized, then we get :

$$\phi_0 = \operatorname{argmax}_{\phi_0 \subseteq \mathcal{V}} \{ \sigma(\phi_0) \mid |\phi_0| = K \} \quad (3.1)$$

### 3.4.2 PageRank Algorithm

PageRank is invented by Larry Page and vastly used in Google. Google's PageRank technology plays a significant role in how any link or information shows up in search results. Understanding how this ranking system works will help us to find the most influential seed set of nodes in a social network.

To begin working with PageRank algorithm we must first understand the PageRank equation which is as follows:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \quad (3.2)$$

Here,  $d$  is damping factor,  $N$  refers to the total number of pages and  $PR$  refers to the PageRank value of a page. It implies that the summation of all the PageRank values will be computed in accordance with the page number that a certain page contains. In our social networking context, we will consider nodes as pages and page number as the link number a node has with other nodes surrounding it. Also, we have to keep in mind about the damping factor,  $d$ .

### 3.4.3 Bayesian Inference

Bayesian Inference can be termed as a method of statistical inference in which we use Bayes' theorem. The main goal of Bayesian Inference is to find the probability of an event as more evidence or information becomes available. This revealed information can be termed as "prior" knowledge. Bayesian Inference can also be derived as the posterior probability as a result of two or more events. To continue using this inference in our proposed model we first need to understand what does Bayes Theorem imply to.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \quad (3.3)$$

Where,

" $A$ " means for any hypothesis whose probability may get affected by the prior knowledge.

" $P(A)$ " stands for the prior probability. That means before observing the data what probability our hypothesis  $A$  had.

“ $B$ ” is the events or evidence that will have effects on the  $P(A)$ .

“ $P(A-B)$ ” derives the posterior probability. It is the probability of  $A$ , after  $B$  is observed.

“ $P(B-A)$ ” the probability of observing  $B$  given  $A$ . It is also called the likelihood.

“ $P(B)$ ” is sometimes termed the marginal likelihood or ”model evidence”. This factor is the same for all possible hypotheses being considered.

In our proposed model, we considered a directed graph denoted as  $G$ . We assume there are hundreds and thousand of nodes in that. There will be nodes that maybe interconnected with each other via directed edges. This implies that, the connected nodes are somehow getting influence by one another. So, we make a new graph from it by following the equation:

$$P(X_i|Y_{1,2,3,\dots}) = IP_{X_i} * IW_{X_i} \quad (3.4)$$

Here,  $X$  is the node that we will be focusing on,  $Y_i$  are the nodes that are influencing the node  $X_i$ ,  $IP$  means Interaction Probability and  $IW$  stands for Influenced Weight. This denotes the fact that the node  $X_i$  will be influenced by the node  $Y_1, Y_2, Y_3, \dots$ .

In the next step we apply Markov’s blanket theorem in a modified way so that it counts a node, its parents and the parents of the parents instead of the child’s parents. Then for all the nodes in the network, we apply Bayesian inference in the context as follows:

$$P(X_i) = \sum_j [(P(X_i|P(X_j) * P(X_{j+1}) * P(X_{j+2}) * \dots)] * P(X_{j+1}) * P(X_{j+2}) * \dots) \quad (3.5)$$

After getting the probability of each node we set a threshold and by doing union over all the seed nodes we get our result.

### 3.5 Algorithms

In this paper, to find the most influential set of nodes we have applied two established algorithms at first; namely Kempe-Greedy Algorithm and PageRank. Later on, we developed an algorithm of our own. In our proposed model we have implemented Bayesian Inference which has been tested with the dataset that we have already pre-processed. The dataset has been extracted from Stanford University. The paragraphs that follows describes our methodologies for each of the algorithms.

For Kempe-Greedy Algorithm:

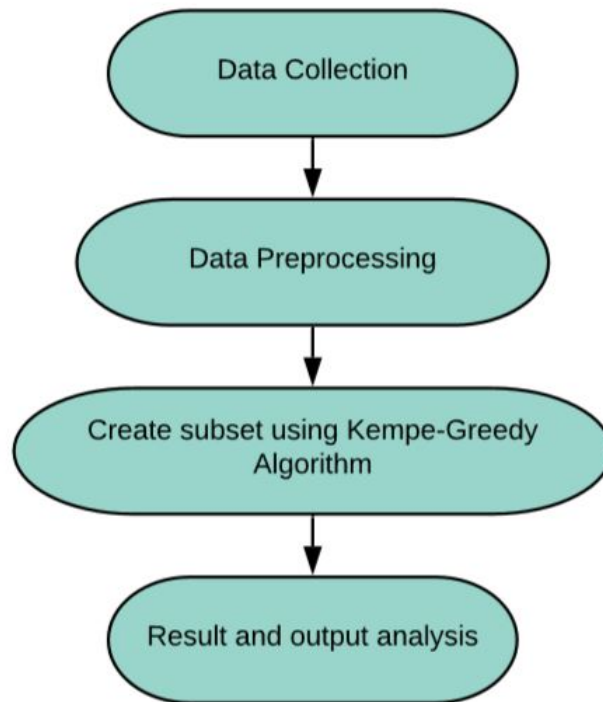


Figure 3.7: Workflow Diagram for Kempe Algorithm

For PageRank Algorithm:

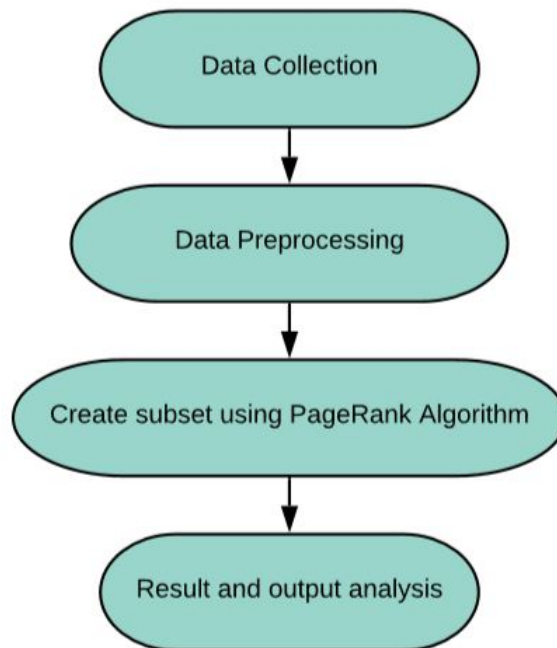


Figure 3.8: Workflow Diagram for PageRank Algorithm

For Bayesian Inference:

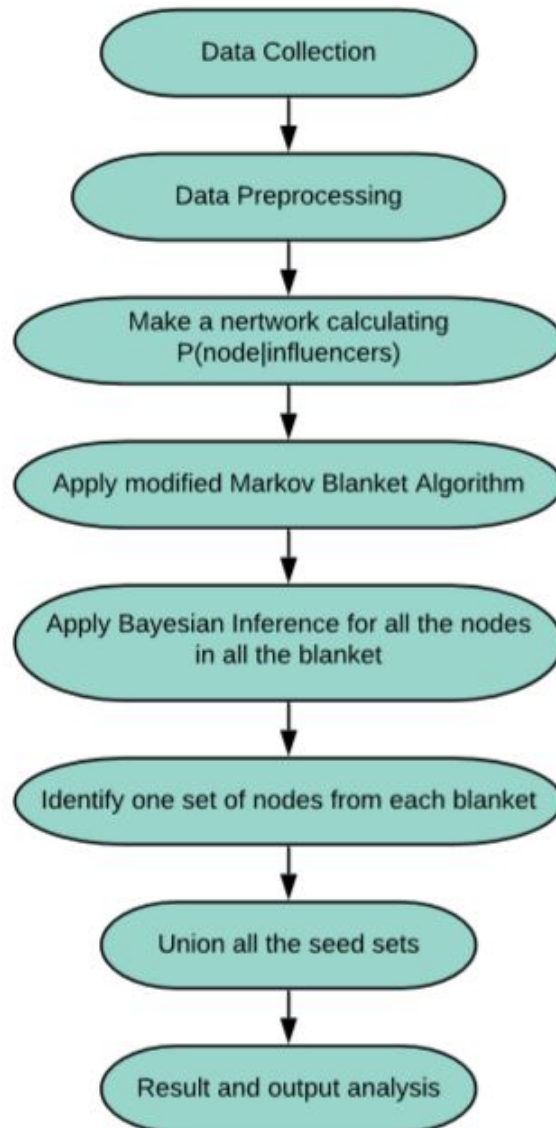


Figure 3.9: Workflow Diagram for Bayesian Inference

### 3.5.1 Kempe Algorithm

Kempe-Greedy algorithm is established by David Kempe. It basically follows a greedy approach to solve a problem. We all know, a greedy algorithm refers to a paradigm which follows an algorithm and that implies to solve the problem piece by piece. In this process, we have to keep in mind that we have to choose the next piece so that it offers more obvious and immediate benefit in reaching the ultimate solution.

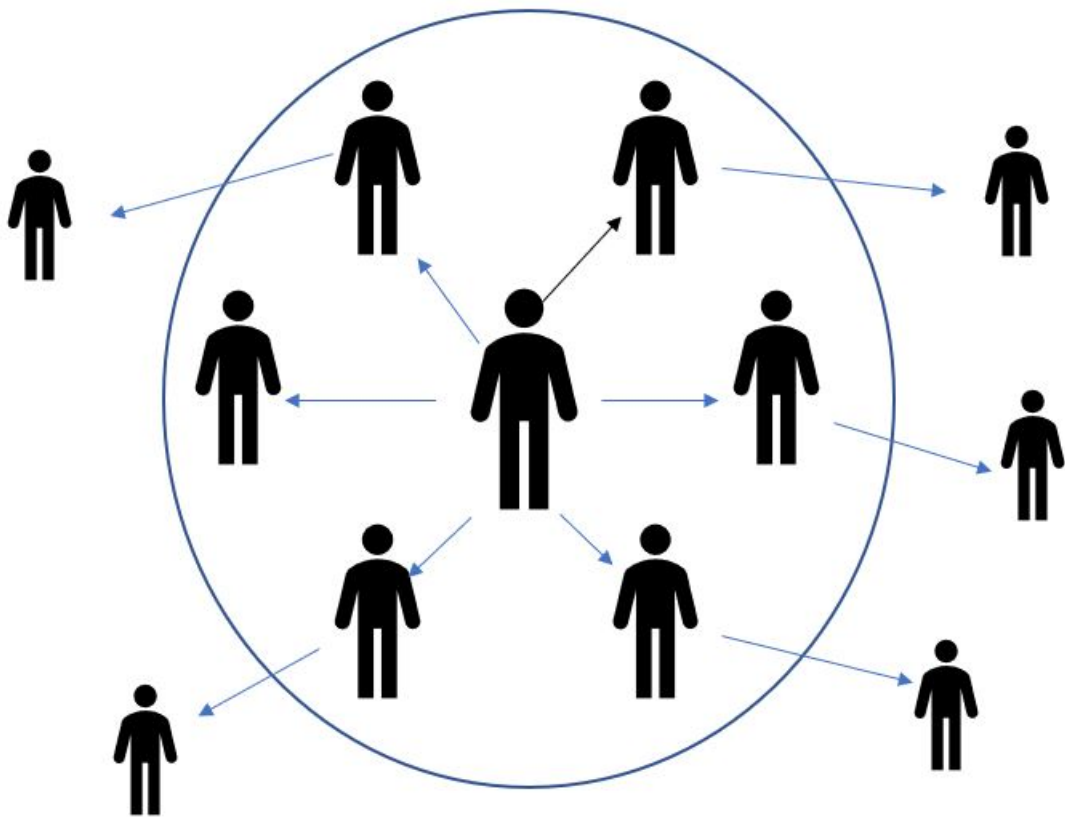


Figure 3.10: Representation of an example in Kempe's Context-01

Greedy solution a problem can be defined with hundreds and thousand of examples. As our context is to find most influential nodes in a social network, for instance, let's consider the following example. A person named X [fig 4.4] has a product to sell. So he informs all his followers about his new product. To reach the ultimate goal he also tells his followers to inform others too. All these were happening in a normal method, without any calculation.



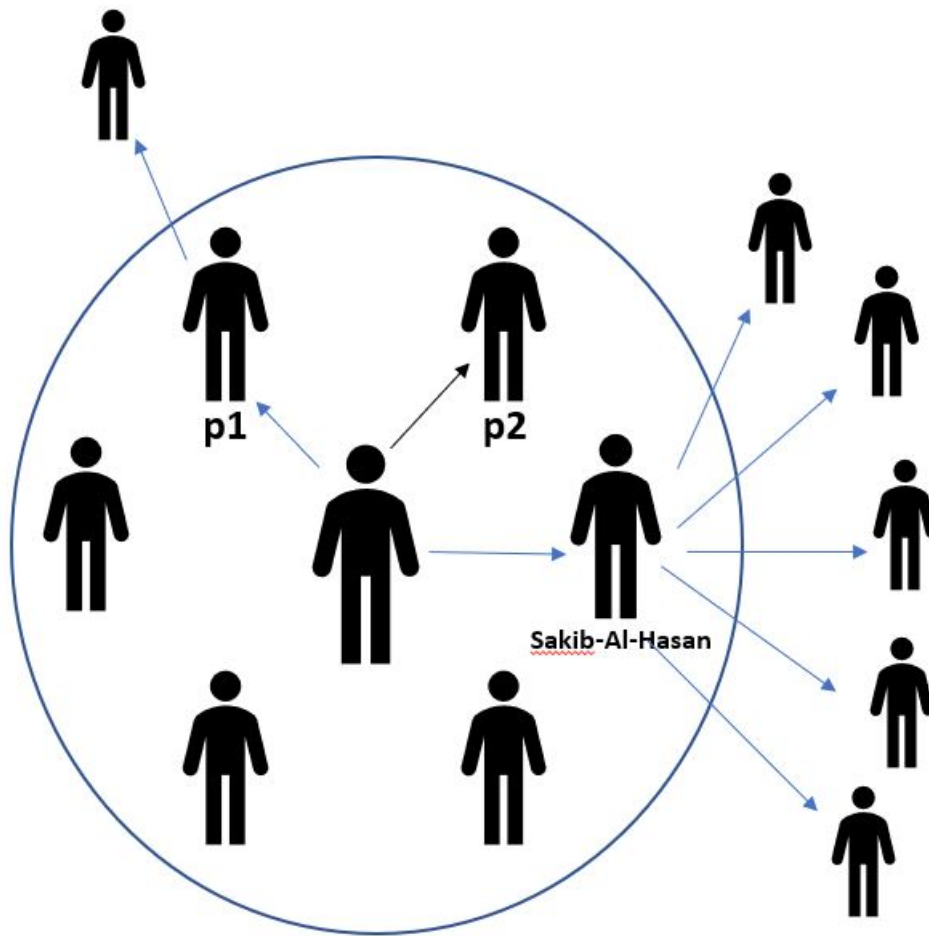


Figure 3.11: Representation of an example in Kempe's Context-02

Suppose another person Y [fig 4.5] wants to do the same thing, i.e. to sell a product. So he followed a greedy approach. He started to inform his followers in from p1. Just like previous person X, he also informs his followers to inform their respective ones. So, p1 informs his one follower. Now as Person Y is following a greedy approach, he breaks his solution into pieces. In this case, each piece of solution can be referred as to inform each of his followers. So, he has to perform the next piece of solution where he has obvious and immediate benefit. He then considers informing p2 and Sakib-Al-Hasan. In this case, p2 has no follower and Sakib has a few ones. So, he informs Sakib, because he can inform his followers. As he has more followers than p2, this step was of obvious and immediate benefit. That is how Greedy algorithm works.

```

KempeGreedy(graph, prob = 0.2, iteration = 1000)
{
    seed_set = []
    nodes = graph.nodes - seed_set;
    for node in nodes{
        inf = IC(seed_set, node, prob, iteration);
        if inf > seed_set.max_inf
            max_inf = inf
            selected_node = node;
        }
        seed_set.append( selected_node)
    }
    return seed_set;
}

```

Figure 3.12: Kempe Greedy Algorithm

To execute this algorithm perfectly, we need few functions. IC(Independent Cascade), Neighbour Finder and Kempe-Greedy. Among them Kempe-Greedy is the most important one. this function call IC function to find the new activated nodes.

**NeighbourFinder**- This function is called in the IC function to find the nodes that are corresponding to a specific node. Neighbour node refers to all the nodes that only on edge away from the given node. This function finds them and returns them in an arrayList.

**IC(Independent Cascade)**- This function is called in the KempeGreedy function to search for the activated seed nodes and their influence over the social network. First it takes in a graph, threshold of probability and number of simulations as input. Then on the graph, it takes one node and search for it's neighbour nodes. It also checks if they are activated or not to make sure there is no repetition of nodes in the arrayList that this function will return.

**KempeGreedy** - This is the function that calculates and compares each of the seed set's influence over the network and place them in the output arrayList. First of all it takes a set of nodes and calls IC function on them to find out their influence. Then it compares this value with the existing one. If the new value is greater then it puts the seed set in the output arrayList and if not, it takes the next node and do till the number of iterations that has already been set. After that we get an arrayList of seed set and their corresponding influence in the network by using a Greedy approach.

### 3.5.2 PageRank Algorithm

PageRank is an algorithm used by Google in their search engine results. This algorithm defines what will be a page's rank in the search result. Basically, This result

is based on how many links a page does contain. The links importance plays a huge role in determining the PageRank.

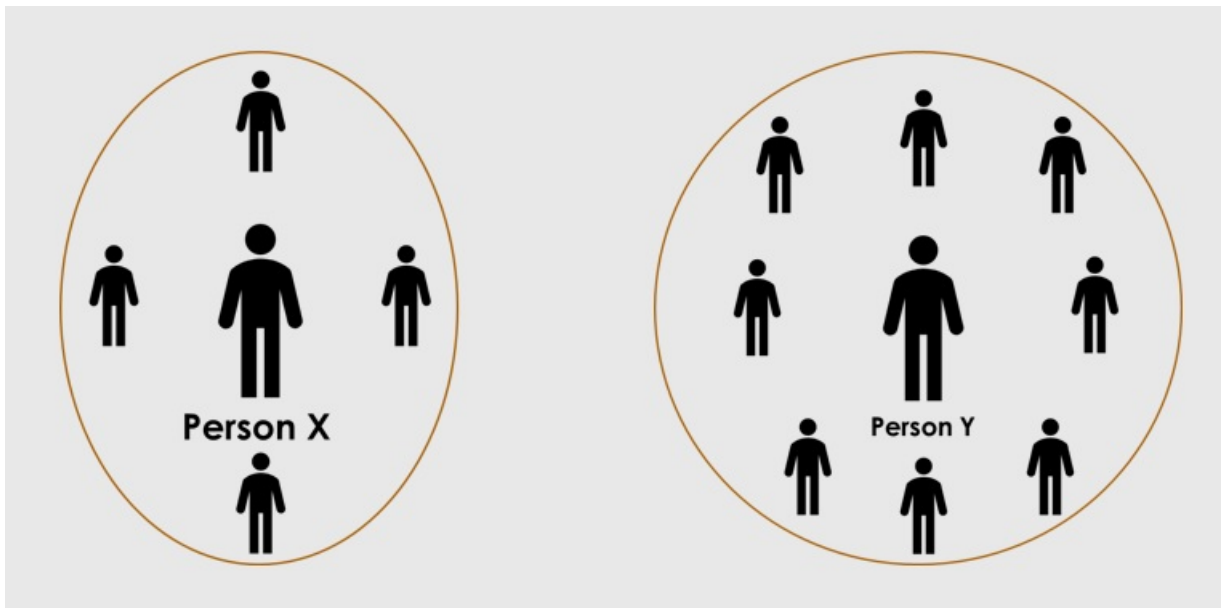


Figure 3.13: Representation of an example in PageRank's Context

To understand how PageRank algorithm actually works, we can think of two person: person X and person Y. Let's assume, person X and Y knows or has a link with 4 and 10 person respectively. So, their PageRank value will be  $1/4=0.25$  and  $1/10=0.1$  respectively. In PageRank algorithm, the lower the value is, the higher position it gets on ranking. That's why person Y will be positioned higher in the ranking.

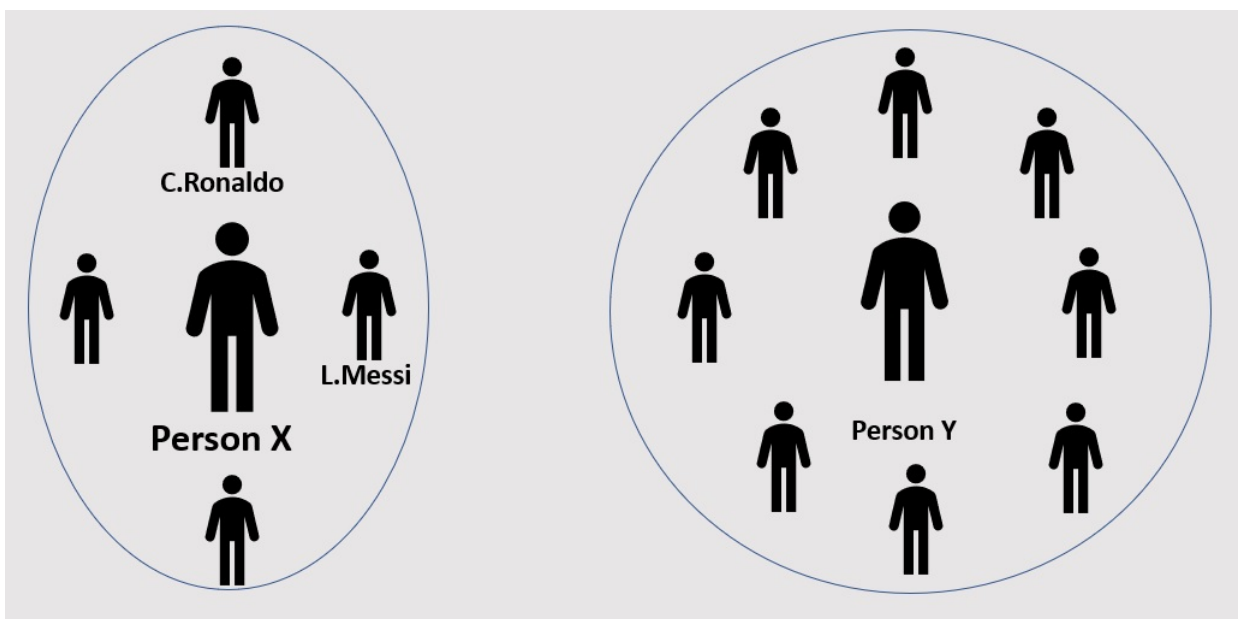


Figure 3.14: Representation of an example in PageRank's Context-02

As mentioned before, not only the connections or links play a role in determining the PageRank value, but also the importance of the link has an impact on the PageRank.

Again, if we consider the same two persons as mentioned above, but this time, if among the 4 persons, person X has links with, two of them are Cristiano Ronaldo and Lionel Messi, then the PageRank Value of person X will be lower than the other one. And he will be ranked higher in ranking.

Now, in our research's context, PageRank algorithm will work by following these steps:

1. Initialize the PR value of each page to  $1/N$ ; where as  $N$ = number of total pages in the network.
2. Choose a node randomly. Calculate it's PR as follows:

$$PR(a) = \sum \frac{PR(\text{a node which is connected to chosen node})}{\text{number of outgoing links the node has}}$$

3. For each node calculate PR() value in every iteration. After every iteration the sum of PR() values of every node should be 1.
4. After a certain number of iteration, the lower the PageRank value, the more likely it is to be appeared in search result.

### 3.5.3 Bayesian Inference

In Bayesian inference, here we split our task in some subsections. Like,

#### Calculate Weighted Probability of a node

The first step is to identify the weighted probability of a node given the nodes which are influencing it. We will identify it based on influenced weight and interaction probability. We have followers' number in our dataset which is used as weighted influence.

The formula will be-  $p(A/BC) = \text{interaction Probability} * \text{Influenced weight}$   
 $p(X_i/Y_{123} \dots)$  IP of  $X_i$   $IW_{x_i}$ ; which means  $X_i$  will be influenced by  $Y_{123} \dots$

#### Applying MST

After that, we will apply MST into our dataset to remove the cycle as in Bayesian inference can not be used in cyclic graph. MST stands for minimum spanning tree. But in our research we will be applying maximum spanning tree. Because, we need maximum route to know which nodes are influenced most. We have to identify the edge value to remove the cycle.

The steps of maximum spanning tree are given below:

- Create a new graph to which our version of MST will be saved.
- Arrange the edge values from higher to lower.
- Take the highest value. Enter the nodes that are connected to with it in the new graph.

- Then take the second one and so on. Only if the both the nodes are already added in the new.

That's how we will be able to remove the cycles.

### Modified Markov blanket

The Markov blanket for a node in a graphical model consists of all the variables that shield the node from the rest of the network. Mainly, a Markov blanket consists of a set of states, internal and external states which conditionally independent of one another. For example, any variable A, A is conditionally independent of B, given another variable, C, if and only if the probability of A and B given C can be written as  $p(A/C)$  and  $p(B/C)$ . But here in our research, Markov blanket consists with a node, its child, its parent and parent's parents. Normally Markov blanket does not work like this way but we modified it like this way for our twitter dataset as it works in that way.

### Influenced Probability of a Node

We modified because as we took here Twitter dataset so if we took child and parent of child then it will not able to work and not give us expected result. So, we applied the Markov blanket in a modified way. Mainly, Markov blanket used for specific area identification. For example, we consider A as a node now we will identify its area by applying Markov blanket. As we don't need other nodes area identification. So, for that reason we use Markov blanket.

Then the total probability of a node A given B and C are its' influencer is given below:

$$p(A) = \sum p(A|BC)(p(B)p(C));$$

Then we have to choose a threshold. Then according to that threshold, we will create several sets of nodes. Those nodes who have the most influence over the selected ones will be in this threshold. Then from those nodes we will take the parents who are influencing them.

After that, we union the parents of those nodes which are more influenced. Because these parents are most influential and that is our goal, to find the most influential user. Here, we have 8 nodes and we mention them as A, B, C, D, E, F, G, and H.

For example, if we add like this then,

Node RTA, RPA, MTA ...

A  
 B 9 1 8...  
 C  
 .  
 .

Here, RT means retweet, RP means reply and MT means mention. Then it means, B is an influencer of A; A is influenced by B with 9 RT, 1 RP, 8 MT. If there is at least one kind of interaction; for example, A is influenced by B with 1 RP.

Then the row has to look like this:

```
Node RTA RPA MTA
A
B 0 1 0
```

if there is no influence from B to A then the row will look like, Node RTA, RPA, MTA ...

```
A
B
```

Here, the slots can be empty.

Now,

$PnI[0]$  will hold  $P(\text{Node } 0/\text{Influencers})$

The Node 0 here will be corresponding to Node A in the data. A's Influencer's are B and C.

So,  $PnI[0]$  is  $P(\text{Node } 0/\text{Influencers})$  which is  $P(A/BC)$ .

$PnI[1]$  is  $P(\text{Node } 1/\text{Influencers})$  which is  $P(B/\text{B's Influencers})$ .

After that, we are computing  $PnI[0]$  which means we are computing  $P(A/BC)$ .

Firstly, we figure out who influences A and we find that is B and C is the influenced A.

We will be able to figure out that by seeing A's column.

So, for  $PnI[0]$  A's column will only be counted as we look for which rows are not null.

It means RTA, RPA and MTA, we see which rows are not null for RTA, RPA and MTA and we got B and C.

Now we want to sum B and C row values for RTA, RPA, MTA, let that is  $s_i$ .

Then we compute B's row for RTA, RPA, MTA, let this be  $s$ .

So, now to compute B's Interaction Probability, we divide  $s/s_i$ .

We store B's Interaction Probability in  $IpE[1]$ , as  $b$  corresponds to 1.

Now we compute C's row for RTA, RPA, MTA, let this be  $s$ .

So now to compute C's Interaction Probability, we divide  $s/s_i$ .

We store C's Interaction Probability in  $IpE[2]$ , as  $c$  corresponds to 2.

for  $IwB$ , we divide B's follower by summation of all follower.

for  $IwC$ , we divide C's follower by summation of all follower.

So now, we have got  $IpE$  and  $IwE$  after those iteration.

Now, we compute PnI by this formula –  

$$IpE [1] *IwE[1] +IpE[2]*IwE[2]$$

But we need the summation of Follower column and for that we apply some formula.

For identifying A,  
 we need  $B/B+C *IwB$  where  $B/B+C$  consider as  $IpB$ .  
 We need  $C/B+C *IwC$  also.

For any influencers  $B/B+C$ ; each  $i$  or sum of  $i$  will consider as  $IpE$  and Influence weight( $IW$ )consider here as  $IwE$ .

So the calculation for A will be to find-

$$p(A/Influencers)= \text{Sum of } IpE*IwE \text{ for all the Influencers.}$$

That is how we calculate a node’s weighted probability given the nodes which are influencing it.

Now, if we are given a sample data like this-

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
1	Node	Follower	RT0	RPO	MT0	RT1	RP1	MT1	RT3	RP3	MT3	RT4	RP4	MT4	RT4567	RP4567	MT4567	RT5	RP5	MT5	RT6	RP6	MT6	RT7	RP7	MT7	
2	0	1																									
3	1	3	1	1	1																					1	
4	3	6							1																		
5	4	6																									
6	4567	2	1	3	4											0	1	2	2	2	4	6		4	1	3	3
7	5	4											1	2	3												
8	6	4					9	4	5																		
9	7	5														8	6	8									
10																											
11																											
12																											
13																											
14																											
15																											
16																											
17																											
18																											
19																											
20																											
21																											
22																											
23																											
24																											
25																											
26																											
27																											
28																											

Figure 3.15: Sample Dataset

After doing some calculations, we get interaction probability and influenced weight of the sample data.

```

1 0 0.2727272727272727
4567 0 0.7272727272727273
6 1 1.0
0 3 0.875
3 3 0.125
1 4 0.3333333333333333
5 4 0.6666666666666666
3 4567 0.12
7 4567 0.88
3 5 0.36363636363636365
4 5 0.6363636363636364
4 6 1.0
1 7 0.125
4567 7 0.875

```

Figure 3.16: Interaction Probability of the Sample Data

```

File Edit Format View Help


---


1 0 0.0967741935483871
4567 0 0.06451612903225806
6 1 0.12903225806451613
0 3 0.03225806451612903
3 3 0.1935483870967742
1 4 0.0967741935483871
5 4 0.12903225806451613
3 4567 0.1935483870967742
7 4567 0.16129032258064516
3 5 0.1935483870967742
4 5 0.1935483870967742
4 6 0.1935483870967742
1 7 0.0967741935483871
4567 7 0.06451612903225806

```

Figure 3.17: Influenced weight of the Sample Data

After that we apply MST and remove the cycle. Then we create the blankets of the nodes consisting of the child, parent and the parent's parent.



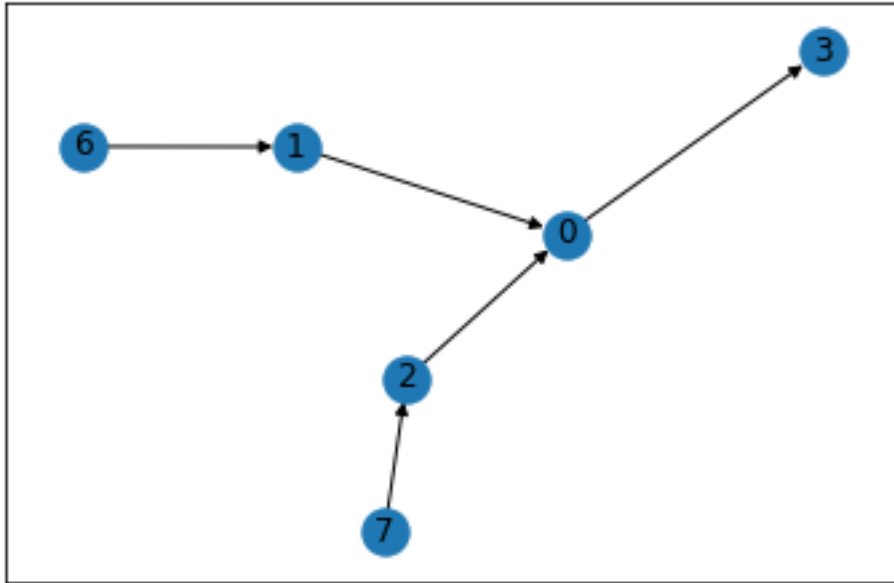


Figure 3.18: Blanket of Node0

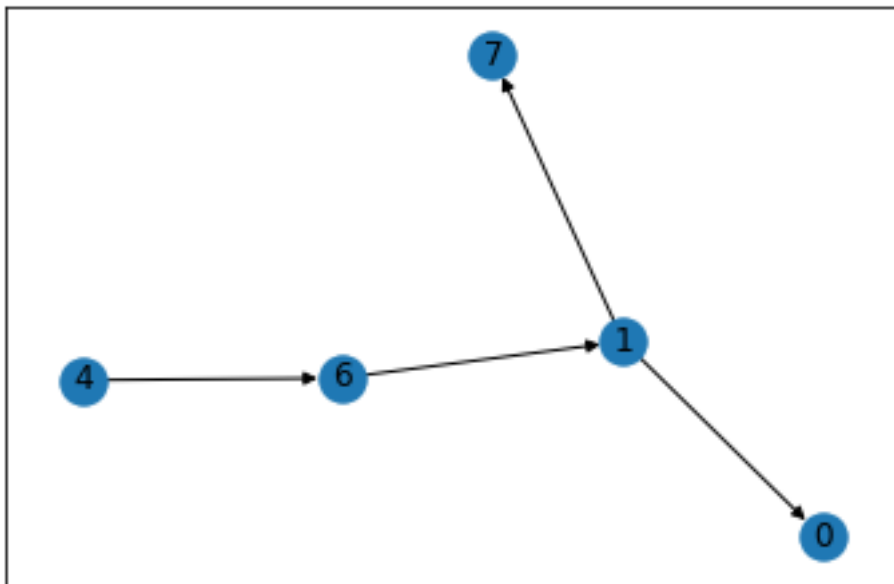


Figure 3.19: Blanket of Node1

Now, let us find out the  $P(0)$  and  $p(1)$  by calculation then match it to the result which we get from the code.

$$\begin{aligned} \text{Weighted } P(0/12) &= (\text{interaction probability of } 1^* \text{ influenced weight of } 1) + \\ &(\text{interaction probability of } 2^* \text{ influenced weight of } 2) \\ &= (0.2727272727272727 * 0.0967741935483871) + (0.7272727272727273 * 0.06451612903225806) \\ &= 0.07331378299 \end{aligned}$$

We know,

$$P(0) = P(0/12) P(1) P(2)$$

Now,

$$P(1) = P(1/6) * P(6)$$

$$P(1/6) = 1.0 * 0.12903225806451613$$

Here,  $P(6) \& P(7) = 1/6$

In this blanket, we are seeing that there are six nodes.

Among them node 3, 6 and 7 are not edges who does not have any dependency.

Therefore, the values of the edge nodes are  $1/6$  respectively without any dependency.

So,

$$P(1) = P(1/6) * P(6)$$

$$= 0.12903225806451613 * 1/6$$

$$= 0.02150537634$$

$$P(2) = P(2/7) * P(7)$$

$$= 0.161290322588064516 * 1/6$$

$$= 0.02365591398$$

$$\text{Now, } P(0) = P(0/12) P(1) P(2)$$

$$= 0.07331378299 * 0.02150537634 * 0.02365591398$$

$$= 0.0003729687191$$

$$\text{Again, } P(1) = P(1/6) * P(6/4) * P(4)$$

Here,

$$P(1/6) = 1.0 * 0.12903225806451613$$

$$= 0.12903225806451613$$

Then,

$$P(6/4) = 1.0 * 0.1935483870967742$$

$$= 0.1935483870967742$$

$$P(4) = 1/5$$

In this blanket, we are seeing that there are five nodes.

Among them node 0, 7 and 4 are not edges who does not have any dependency initially.

Therefore, the values of the edge nodes are  $1/5$  respectively without any dependency.

$$\text{Now, } P(1) = (1/6) * P(6/4) * P(4)$$

$$= 0.12903225806451613 * 0.1935483870967742 * 1/5$$

= 0.004994796928

We will now check this calculations with the code.

```
In [6]: runfile('E:/13TH SEM/Full code with Short Sample/final.py', wdir='E:/13TH SEM/Full code with Short Sample')
['1', '0', '4567', '6', '3', '4', '5', '7']
{'4567': 0.02838709677419355, '1': 0.004994797086368366, '6': 0.004162330905306971, '4': 0.0012108598997256645, '5': 0.00039731340459748377, '7': 0.0003121748178980229, '3': 8.27736259578091e-05, '0': 3.7296871911352786e-05}
<Figure size 432x288 with 0 Axes>
```

Figure 3.20: Result of the Sample Data

The results match with the codes. After that, a threshold is selected which shows the most influenced nodes thus from their parents we get the desired seed sets.

```
In [7]: runfile('E:/13TH SEM/UserMining-Short Data/done2.py', wdir='E:/13TH SEM/UserMining-Short')
['1', '0', '4567', '6', '3', '4', '5', '7']
most influenced nodes are: {'4567': 0.02838709677419355, '1': 0.004994797086368366}
seed set : {'3', '7', '6'}
<Figure size 432x288 with 0 Axes>
```

Figure 3.21: Seed Set of the Sample Data

## 3.6 Implementation

### 3.6.1 Kempe Algorithm

We used Python3.7 as our programming language and Spyder Notebook as our IDE to run our Program. To apply Kempe-Greedy algorithm, we had to use the dataset that we have already created. We extracted all the attribute in our dataset for instance, "mentions", "replies", "retweets" and "follows". Nonetheless, we do not need these attributes in performing Kempe-Greedy Function. We are using only the "followers" section of our main dataset "out.csv" file for this greedy approach. We are correcting the orientation of the "out.csv" file thus creating the pag\_kem\_data for running the Kempe algorithm on it. We need to find out the connections between each and every node. Because Kempe-Greedy function activates a set of nodes given on a specific threshold.

We need some built-in library functions to get our expected result. that is why we import three libraries :

- networkx
- numpy
- matplotlib.pyplot

## Independent Cascade Function-

```
def IC(Networkx_Graph,Seed_Set,Probability,Num_of_Simulations):
    spread = []
    for i in range(Num_of_Simulations):
        new_active, Ans = Seed_Set[:], Seed_Set[:]
        while new_active:
            targets = Neighbour_finder(Networkx_Graph,Probability,new_active)
            np.random.seed(i)
            success = np.random.uniform(0,1,len(targets)) < Probability
            new_ones = list(np.extract(success, sorted(targets)))
            new_active = list(set(new_ones) - set(Ans))
            Ans += new_active
        spread.append(len(Ans))
    return(np.mean(spread))
```

Figure 3.22: Independent Cascade Function

We define a function named IC. IC stand for Independent Cascade. This function takes a graph, seed set, a threshold probability and number iterations as it's input. IC function takes the seed set and calculates it's influence over the graph and it's neighbours. It also takes in the probability and how many times it should do the same thing over and over again.

We make an arrayList named "spread" where we will store the nodes that have been calculated. It does the same thing for the number of iterations that has already been given to it. It creates a list named "new\_active". Here we store the nodes that have been activated recently. By using "Neighbour\_finder" function we can find the corresponding neighbor nodes of a specific node in the graph. Then we check if any of the neighbouring nodes can be activated. If yes, then we put them in the "new\_active" list. Also, we have to check, whether it has already been added to the list or not to reduce redundancy.

As last step of this function, we add the selected nodes to the arrayList named "spread".

## Neighbour\_finder Function

```
def Neighbour_finder(g,p,new_active):

    targets = []
    for node in new_active:
        targets += g.neighbors(node)

    return(targets)
```

Figure 3.23: A function to Find Neighbour

Neighbour finder function is basically doing as it's name suggests. It finds the neighbour of a specific node. It creates an arrayList named "targets", where it

saves all the neighbouring nodes from the "new\_active" list. And returns all the neighbouring nodes of a specific node as an arrayList.

### Kempe-Greedy Function-

```
def KempeGreedy(graph, num_seed_nodes, prob=0.0001, n_iters=1):
    max_spreads = []
    ultimate_seed_set = []
    for _ in range(num_seed_nodes):
        best_node = -999
        best_spread = -np.inf
        nods = graph.nodes - ultimate_seed_set;
        for node in nods:
            print(ultimate_seed_set+[node])
            each_infl = IC(g, ultimate_seed_set + [node], prob, n_iters)
            if each_infl > best_spread:
                ALL_infl.append(each_infl)
                best_spread = each_infl
                best_node = node

        ultimate_seed_set.append(best_node)
        max_spreads.append(best_spread)
    return ultimate_seed_set, max_spreads
```

Figure 3.24: Kempe-Greedy Function for Algorithm

This is the most important function of this algorithm. This algorithm returns a seed set that has maximum influence and their influences over the network. This function takes the directed graph, number of seed nodes, threshold probability and number of iterations as input.

First of all it creates to arrayList naming "max\_spreads" and "ultimate\_seed\_set". Now for each node, it does the exact same thing for exact number of times that it has been told. We initially set the values of best node and best spread to -infinity to compare and get the ones with highest value. We created a list named "nods" which comprised only with the nodes that are left in the graph after removing the ones that are present in the "ultimate\_seed\_set".

After that we call IC function on each and every node and get their influence value. We then just compare each and every existing node's influence value with the new calculated one. If the newly calculated one is higher than the previous one, we swap the node and influence value. If not, we move forward to the next node in the "nods" list.

After doing this exact thing for all the nodes we get "ultimate\_seed\_set" and their maximum influence over the network.

## Main Handler Part-

```
ALL_inf=[]
k = 15
prob = 0.2
n_iters = 50
g = nx.read_edgelist('pag_kem_data.txt', create_using=nx.DiGraph());
greedy_solution, greedy_spreads = KempeGreedy(g, k, prob, n_iters)

print(prob, n_iters)
print('Seed Set: ', greedy_solution)
print('Maximum Influences: ', greedy_spreads)
```

Figure 3.25: Main Handler of Kempe-Greedy Algorithm

In the main handler part, we set the seed set number as  $k=15$ ; means it will give us the best 15 set of nodes as output. We also set the threshold probability as 0.2 and number of iteration as 50. We load our main dataset and then we created the directed graph of the dataset using `nx.DiGraph()` method from the library. Then we run the "kempeGreedy" function. This function has the other two function as nested in it.

This will give us the set of nodes, their influence and the graph of their influence over the network. This is how Kempe-Greedy Algorithm works over a social network.

### 3.6.2 PageRank Algorithm

We used Python3.7 as our programming language and Spyder Notebook as our IDE to run our Program. To apply PageRank we used the dataset that we have created. This new dataset has 5000 nodes and all the attributes that we needed. Though this dataset have many attributes such as, "mentions", "replies", "retweets" and "follows"; but to apply PageRank we only need the connections or so to say degrees of the nodes.

In the beginning we had to import some libraries into our IDE. They are

- Networkx
- matplotlib.pyplot
- islice from itertools

First of all we import the Higgs-twitter dataset into our project. The dataset is in .csv file format which is out.csv. We are correcting the orientation of the csv file thus creating the `pag_kem_data` for running the pagerank algorithm in it. Now we had to make a graph out of it and also run the PageRank algorithm on it as follows:

```
g=nx.read_edgelist('pag_kem_data.txt',create_using=nx.DiGraph());
print(nx.info(g));
pgr_g=nx.pagerank(g);
```

Figure 3.26: Making A Graph From The File

Here, from networkx library's "nx.read\_edgelist()" function is being used to explore the connected edges in the .csv file and by using "nx.DiGraph()" function we can put it in a directed graph. After making a graph we make a dictionary file that has all the nodes in the graph. In the process of creating a dictionary file, the PageRank algorithm was run on it. As a result the "pgr\_g" file has all the nodes along with their PageRank values. The graph for the new .csv file looks like this:

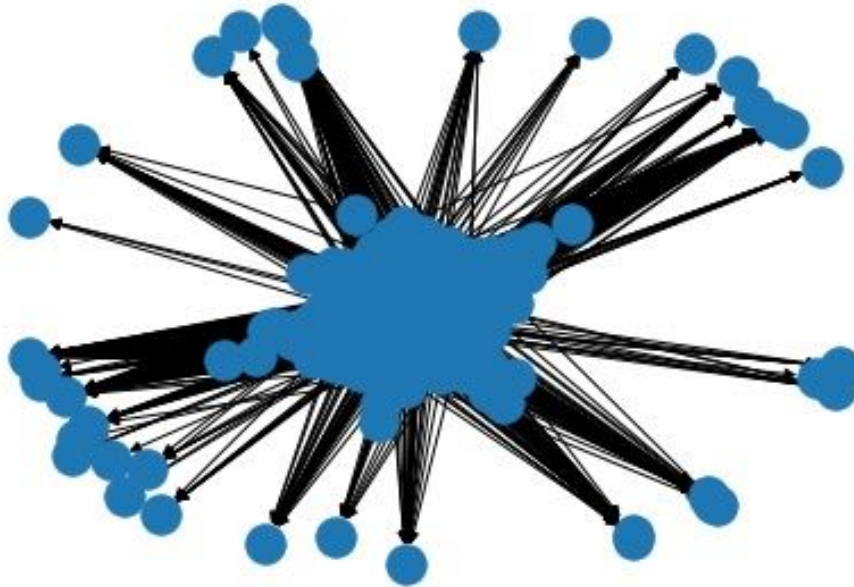


Figure 3.27: Directed Graph of The New File

Also another important aspect of this algorithm is "The Damping Factor". It is a click-through probability of a graph. It is a common phenomena that a graph will have some nodes that has no out-degree as well as no in-degree. Depending on the algorithms, these types of nodes creates an infinite loop. This is called a "sink". The purpose of the damping factor is to prevent these sinks (i.e. pages with no outgoing links) from "absorbing" the PageRank's of those pages connected to the sinks.

It is common to see that an infinite surfer would have to end up in a sink given enough time, so the damping factor allows a heuristic to offset the importance of those sinks. to prevent these sinks, we manipulated the PageRank's equation as follows:

$$PR(a) = \frac{1+d}{n} + d * \sum \frac{PR(\text{a node which is connected to chosen node})}{\text{number of outgoing links the node has}}$$

The PageRank equation has "1+d/n" in the front to calculate the dumping factor. Now if the dumping factor is set to "1", then, the person clicking will click forever and they will always end up in a sink. In this case,the first term is discarded. The second term, given an infinite number of iterations to convergence, is equivalent to finding the steady state of the Markov chain representing pages and links. If it is set to "0", then all clicks are random restarts, which are uniformly distributed (the 1/N coefficient in the first term) by definition. That is why the dumping factor for our research is set to 0.85 to prevent all the sinks in the graph.

```
sorted_pgr_g=sorted(pgr_g.items(), key=lambda kv:kv[1] ,reverse=True)
```

Figure 3.28: Sorting The Dictionary File

After getting the new Dictionary File we noticed that, in the dictionary file all the nodes are in a serial matching with the descending order of the PageRank value. But we needed the values in ascending order and the nodes according to the ascending order of the values. That is why we used "sorted" function to get our expected result. As it was a dictionary file, that means the nodes and and the values were altogether. And we needed to sort the values only. That is the reason to use the "lambda" function which allowed us to sort only the value portion of the file.

```
k=15;
best_k_from_sorted_pgr_g = sorted_pgr_g[0 : k];
print("seed set with its pagerank")
print(best_k_from_sorted_pgr_g);
```

Figure 3.29: Code of Finding Seed Set with it's PageRank value

After sorting the dictionary file, our next step would be to set a threshold and find that much number of nodes. Here we decided to find best 15 nodes that has the less PageRank Value. That is how we can find most influential nodes by using PageRank Algorithm.



### 3.6.3 Bayesian Inference

The pseudo-code for Bayesian Inference algorithm is given below:

```
float node_probability (Blanket, node) {  
    if (node.in_Blanket.indegree==0) {  
        return 1/Blanket.nodes;  
    }  
    else {  
        s=0  
        Parents= node.in_Blanket.parent();  
        for each_parent in parents {  
            c= Compute_conditional_probability (Node, each_parent);  
            s=s+c;  
        }  
        ans=1;  
        for each parent in parents {  
            parent_prob=node_probabilty (Blanket, each_parent)  
            ans= ans*parent_prob;  
        }  
        return s*ans;  
    }  
}
```

Figure 3.30: Pseudocode for Bayesian Inference

Here, if we are given a node A and its parents are B and C. Then P(A) will be,

$$P(A) = P(A/BC) * P(B) * P(C)$$

So, when `node_probability` it is given node A and its blanket.  
First of all, we need P(A/BC).

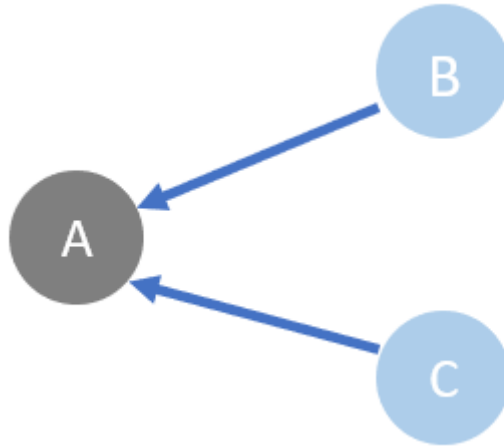


Figure 3.31: Blanket A

In `Parents= node.in_Blanket.parent()`; it is giving B and C.  
This means that it is finding the parents of node A.

In the next step the for loop 'each parent in parents' is taking the parent one by one.

The first parent is B.

```
c= Compute_conditional_probability(Node, each_parent);
```

This means it will call `Compute_conditional_probability` with A and its parents, in our example With A and B, then A and C.

Now it will compute the `Compute_conditional_probability` by multiplying  $IP*IW$  and return the result in S.

So, now  $s=s+c$

$s=0+c$

$s=c$

Again, the for loop starts and now it is taking C as a parent computing  $IP*IW$  which is saved in c.

Thus, it saves the value with the earlier value which was saved in s.

That is how we get  $P(A/BC)$ .

We need  $P(C)$  and  $P(B)$  now.

We will call `node_probability` with each parents of A.

```
parent_prob=node_probabilty (Blanket, each_parent);
```

Which means in this example, the code will call `node_probability` with Blanket and node B and then again `node_probability` with Blanket and node C.

As  $P(B)$  is a corner node and it has no dependency it will go in the if condition.

Th result will be  $1/Blanket.nodes$ .

Same condition will go for C.

The result will be saved in `parent_prob`.

```
ans= ans*parent_prob;
```

here in ans the  $P(B)$  and  $P(C)$  will be saved.

Lastly return  $s*ans$  which means  $P(A/BC) *P(B)*P(C)$  is returning.

If B and C had any dependency then it would not go in the if condition.

Then the same else loop will happen.

This is how the Bayesian Inference algorithm works.

# Chapter 4

## Results and Analysis

In this research paper our target was to find most influential users in a social network. To achieve our goal we used two established algorithms: Kempe and PageRank. And by exploring several sectors of influential user mining, we came across that there was not enough work done in this sector by using Bayesian Inference. So we decided to propose a new model, that integrated Bayesian Inference and Markov Blanket altogether to give us the result.

### 4.1 Kempe Algorithm

The main goal of using Kempe-Greedy function is to mine the seed sets of a social network that has maximum influence over a social network. Kempe's greedy function puts more focuses on the neighbour activation on a given threshold. A greedy algorithm, as the name suggests, can be termed as a process that will always choose the next best possible way to solve any problem. This implies that, with a view to having a solution that will be globally optimal, this greedy approach will make a locally-optimal choice in the first place.

We have already discussed how Kempe-greedy algorithm works. It chooses seed sets and run IC(Independent Cascade Function) function on them to find out the influences the seed sets have on the network. Then the seed set with maximum influence is considers the most influential seed set of nodes from that network.

We set the probability to 0.2 to search for nodes that are activated in between this threshold. And we also set the damping factor to 0.85. Damping factor prevents the sinking of the nodes in a graph. In the handler part of the code, we set the value of "k" to 15. That refers that , this greedy-algorithm will give us the best 15 seed sets according to their influence. Also we have to set the value for number iterations.

With these values, IC will be called upon the graph as many times it has been set to. IC will activate the suitable nodes and will check which are the other nodes that can be activated with the first one. IC sets the neighboring nodes to an uniform probability. This process in one of infinite loop. That is why , we have to set a threshold. As soon as the seed sets between the given threshold are found, then Kempe-greedy Algorithm is run on each of the sets. The function of kempe greedy is to find "k" number of nodes in the new graphs.

After finding seed sets of the given parameter, It gives us the output as follows:

```
Seed Set: ['88', '1988', '677', '3571', '349', '77', '519', '2567', '511',
'1343', '2866', '26', '220', '2342', '8']
Maximum_Influences: [61.06, 75.76, 91.6, 105.22, 117.98, 128.2, 137.74, 145.7,
152.98, 158.32, 163.08, 166.0, 170.42, 174.5, 178.9]
```

Figure 4.1: Seed Set For Kempe

This are the seed sets for Kempe which are most influential one.

```
total child response probability: 4.719893433494587e-73
# of zero probability child: 8
% of zero probability child: 88.88888888888889 %
```

Figure 4.2: Child response Probabilty of Kempe

Total child response probability IS 4.719893433494587e-73.

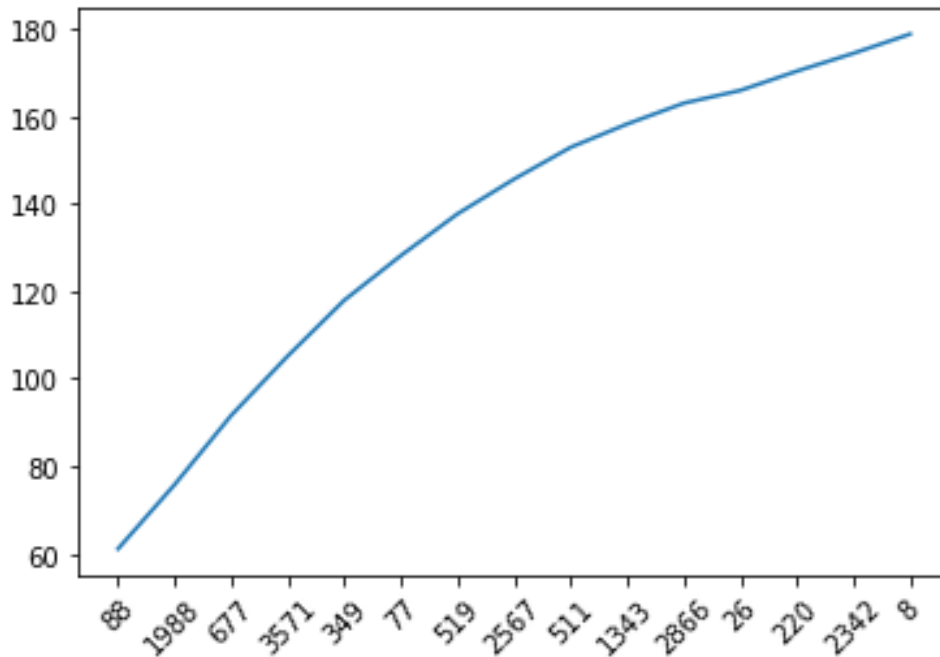


Figure 4.3: Graph of Kempe

Kempe: with k=15 Given 0.2 probability and 50 iteration, Kempe seed set is 88, 1988,...8. This full set is the most influential one.

## 4.2 PageRank Algorithm

The sole purpose of using PageRank Algorithm was to find the seed set of most influential node in a social network. Google's PageRank algorithm put more focuses on the link that a page carries. Similarly, ours also find the best nodes on the basis of in-degree and out-degree edges of a nodes. As in our implementation part, we wanted to find out the best 15 nodes from our dataset, we set the value to 15.

```

Type: DiGraph
Number of nodes: 1863
Number of edges: 2184
Average in degree: 1.1723
Average out degree: 1.1723
seed set with its pagerank
[('2555', 0.004741526375439901), ('1632', 0.004317596314595651), ('2907',
0.003781177417488757), ('3167', 0.0037300094249708053), ('492',
0.003533269743782118), ('2819', 0.002445228007528367), ('2157',
0.002443192398723843), ('2871', 0.002422985645237714), ('429',
0.002421610706547842), ('1701', 0.0022873648162047063), ('1715',
0.0022867300458551706), ('1596', 0.0022824427820831994), ('3910',
0.0022824427820831994), ('4528', 0.002218621543619877), ('2014',
0.0020598149073171052)]

```

Figure 4.4: Seed Set For PageRank Algorithm

This 15 has the highest influence among other nodes. We all know now, that the less the value, the higher rank the node gets in the ranking. Here 2555 has highest Pagerank value. On the other hand 2014 has the lowest value. That is how PageRank algorithm finds the best seed set for most influential user in a social network. The graph output of PageRank is as follows:

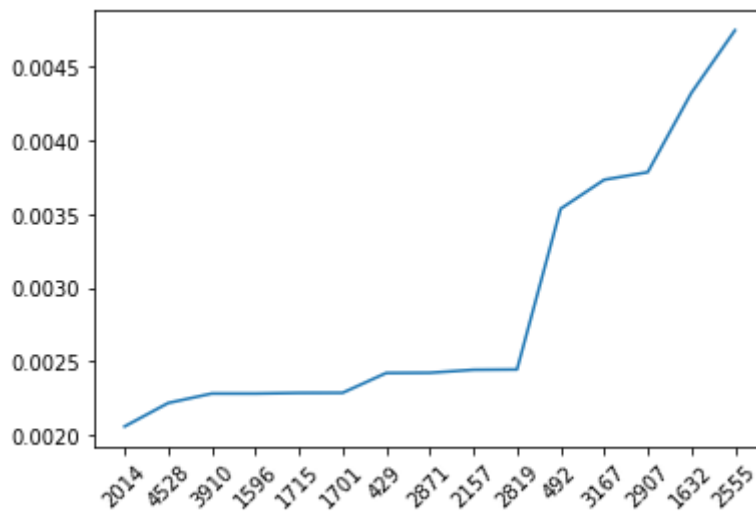


Figure 4.5: Output Graph For Pagerank Algorithm

The graph from the PageRank algorithm shows us the least number of node has the least number of PageRank value. In the graph, the X-axis denotes the node's serial number and the Y-axis denotes the corresponding PageRank value. The graph indicates that the nodes that are selected earlier will have the least PageRank Value. And the least PageRank value a node gets, it is most likely to be present in the list of most influential nodes in that very network. In the graph; X axis, k(=15) most important nodes where 2555 is most important as it has the highest PageRank value.

### 4.3 Bayesian Inference

As we have already discussed, In bayesian inference there are several steps in between getting our target output.

In a graph there are several nodes and few other other nodes are influencing it so our first step was to identify the weighted probability of a node given the nodes which were influencing it. We calculate the weighted probability of each node by keeping in mind the value of retweets, replies or mentions one node gets from the other ones. Also, there may be some cases where a node won't be having any weighted quality because it has no interaction with any other node in the graph.

When we imported our newly created dataset to our code to find out the weighted probability of each node we found out the results as follows:

```
For Node 78 the influencers influence values are

      RT78  RP78  MT78
784      1.0   NaN   NaN
2280     1.0   NaN   NaN

Influencers are
Node 2477 its IpE: 0.5 & IwE: 0.0008850389260483403
Node 550 its IpE: 0.5 & IwE: 0.0006265762308306835
```

Figure 4.6: Weighted probability of Node78

If we look at the results of not 78, we got to know that node number 78 retweets 784, and node number 2280. Also no number 78 are being influenced by node, 2477 and node 550. Here IpE means interaction probability and IwE means influence weight. The results of note 78, we also got to know that the influencers i.e node784 and node550 have IpE of.5 each and IwE of 0.000885 and 0.0006265 respectively.

```
For Node 769 the influencers influence values are
```

```
Empty DataFrame
Columns: [RT769, RP769, MT769]
Index: []
```

```
Influencers are
```

Figure 4.7: Weighted probability of Node769

Now, if We look at the results of note, 769, we will see that it has no influencer or influencee, that is why, it has zero influence over the network.

Here are few more snapshot of our result of the first step.

For Node 4665 the influencers influence values are

	RT4665	RP4665	MT4665
154	1.0	NaN	NaN
523	1.0	NaN	1.0
769	1.0	NaN	NaN
800	1.0	NaN	NaN
1314	1.0	NaN	NaN
1412	1.0	NaN	NaN
1645	1.0	NaN	NaN
1660	1.0	NaN	NaN
2031	1.0	NaN	NaN
2202	1.0	NaN	NaN
2229	1.0	NaN	NaN

Influencers are

Node 128 its IpE: 0.0555555555555555 & IwE: 0.0002193016807907392  
Node 1984 its IpE: 0.1111111111111111 & IwE: 0.0009163677375898746  
Node 2431 its IpE: 0.0555555555555555 & IwE: 0.00014097965193690377  
Node 2513 its IpE: 0.0555555555555555 & IwE: 0.00028195930387380755  
Node 340 its IpE: 0.0555555555555555 & IwE: 0.00013314744905152024  
Node 3593 its IpE: 0.0555555555555555 & IwE: 7.048982596845189e-05  
Node 3972 its IpE: 0.0555555555555555 & IwE: 0.0005090931875499303  
Node 3994 its IpE: 0.0555555555555555 & IwE: 0.0003681135356130265  
Node 4619 its IpE: 0.0555555555555555 & IwE: 0.00021146947790535566  
Node 492 its IpE: 0.0555555555555555 & IwE: 0.0012766490703175176  
Node 4978 its IpE: 0.0555555555555555 & IwE: 1.5664405770767086e-05

Figure 4.8: Weighted probability of Node4665

For Node 799 the influencers influence values are

	RT799	RP799	MT799
861	0.0	1.0	1.0
1360	1.0	NaN	NaN

Influencers are

Node 2630 its IpE: 0.6666666666666666 & IwE: 3.9161014426917716e-05  
Node 3491 its IpE: 0.3333333333333333 & IwE: 0.00021146947790535566

Figure 4.9: Weighted probability of Node799



For Node 640 the influencers influence values are

	RT640	RP640	MT640
465	1.0	NaN	NaN
480	1.0	NaN	NaN
1314	1.0	NaN	NaN
1526	1.0	NaN	NaN
2543	3.0	NaN	NaN

Influencers are

Node 1873 its IpE: 0.1111111111111111 & IwE: 3.9161014426917716e-05  
 Node 1898 its IpE: 0.1111111111111111 & IwE: 2.349660865615063e-05  
 Node 340 its IpE: 0.1111111111111111 & IwE: 0.00013314744905152024  
 Node 3774 its IpE: 0.1111111111111111 & IwE: 2.349660865615063e-05  
 Node 997 its IpE: 0.3333333333333333 & IwE: 0.0001096508403953696

Figure 4.10: Weighted probability of Node640

We also get both the Interaction Probabilty and Influenced Weight of all the nodes that are present in the graph. With these values we can get the Weighted Probability of allnodes in the graph. Here is a snap of IpE and Iwe of few pf the nodes.

```

For Node: 4693, P(Node|Influencers) =8.772067231629567e-05
For Node: 4697, P(Node|Influencers) =0
For Node: 4699, P(Node|Influencers) =0
For Node: 47, P(Node|Influencers) =0.0011082567082817713
For Node: 4700, P(Node|Influencers) =0
For Node: 4704, P(Node|Influencers) =0.00019580507213458857
For Node: 4705, P(Node|Influencers) =0
For Node: 4706, P(Node|Influencers) =0
For Node: 4708, P(Node|Influencers) =9.398643462460251e-05
For Node: 4713, P(Node|Influencers) =0.00025454659377496515
For Node: 4714, P(Node|Influencers) =0
For Node: 4717, P(Node|Influencers) =0.0003132881154153417
For Node: 4718, P(Node|Influencers) =0
For Node: 4719, P(Node|Influencers) =0
For Node: 472, P(Node|Influencers) =0
For Node: 4720, P(Node|Influencers) =2.6107342951278476e-06
For Node: 4722, P(Node|Influencers) =0
For Node: 4724, P(Node|Influencers) =0
For Node: 4727, P(Node|Influencers) =0
For Node: 4728, P(Node|Influencers) =0
For Node: 473, P(Node|Influencers) =0.00012270451187100883
For Node: 4730, P(Node|Influencers) =0
  
```

Figure 4.11: Weighted Probabilty of Some of the Nodes

We can see that some of the node has 0 as their weighted probability; that's because these nodes have no interactions whatsoever with other nodes in the network.

After making a network with these probability of each and every nodes, we made Modified Markov Blanket for every node. We modified the Markov Blanket Model to calculate the node, its parents and parent's only. As all these are in the context of social network that is why we had to modify it to get to the real blanket that will actually have an effect on determining the influence over the network. Here are some of the blankets:

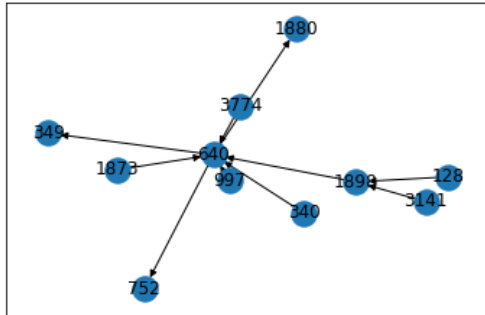


Figure 4.12: Blanket of node640

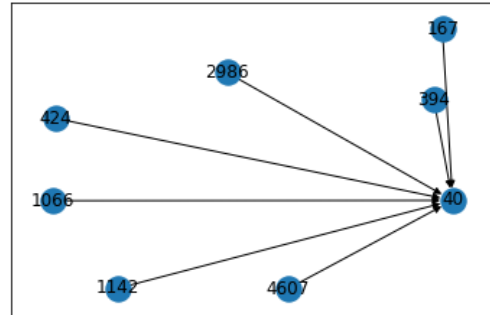


Figure 4.13: Blanket of node40

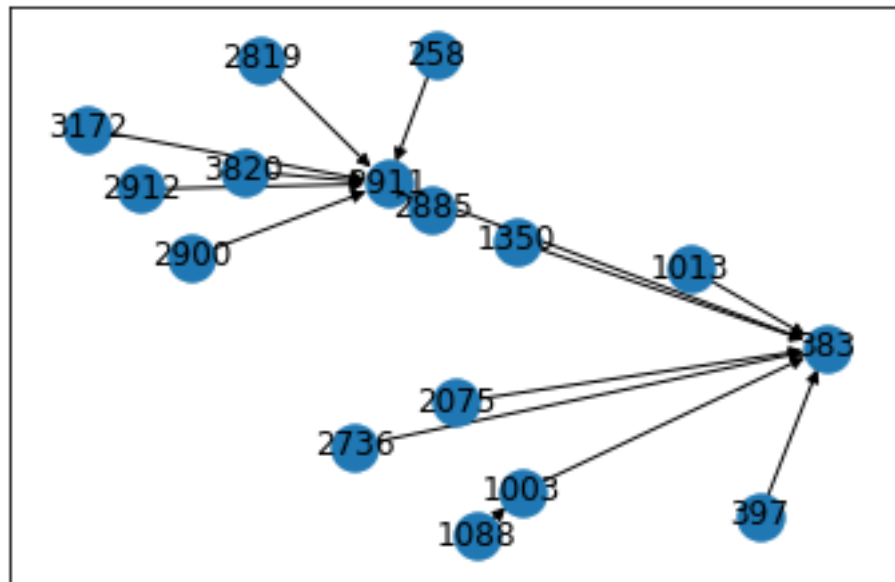


Figure 4.14: Blanket of node383

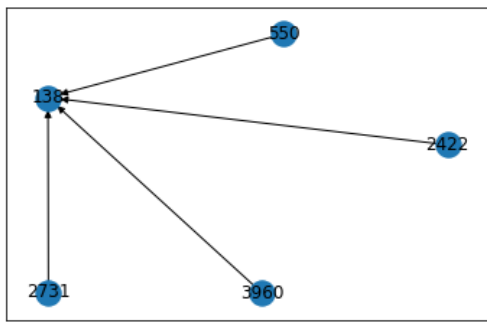


Figure 4.15: Blanket of node138

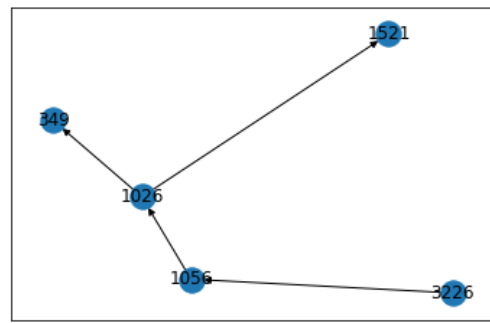


Figure 4.16: Blanket of node1026

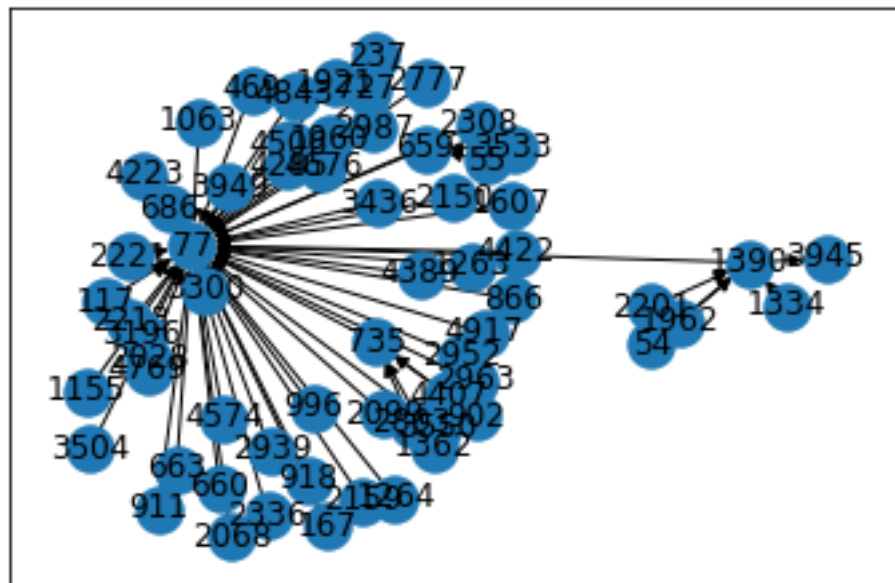


Figure 4.17: Blanket of node77

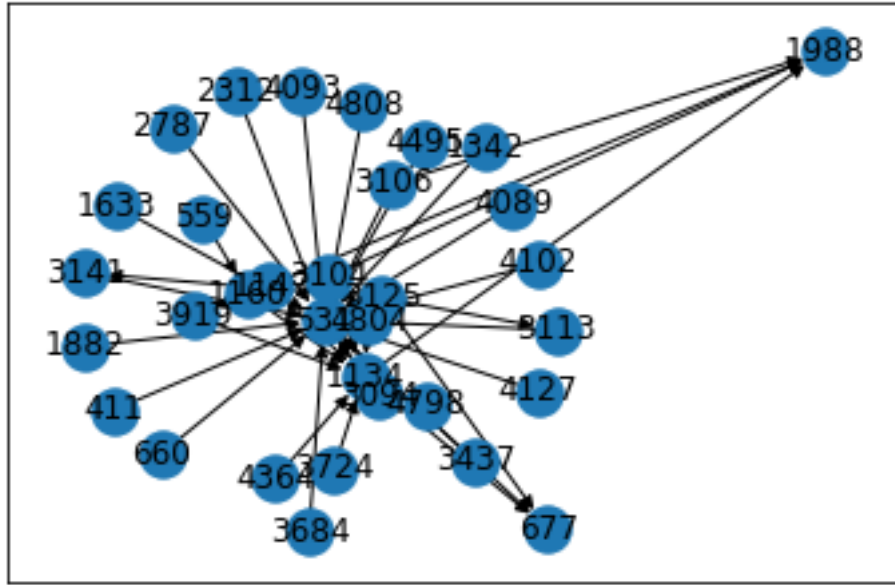


Figure 4.18: Blanket of node1134

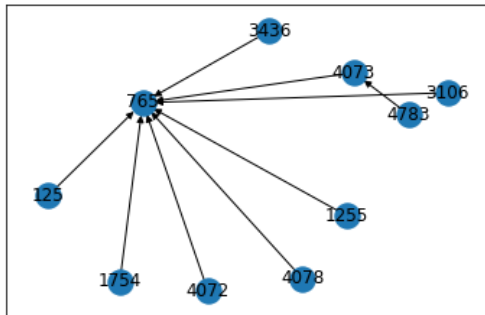


Figure 4.19: Blanket of node765

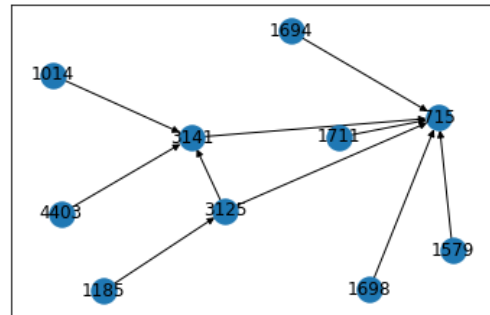


Figure 4.20: Blanket of node715

After making blanket for every node, we move onto finding the parents and their parents for each node. Then we have to do union function to all the nodes because, if we chose to intersect the nodes we would get very few nodes in return. And those node will have the minimum influence of the network. The nodes that are found by doing intersect should be present in every blanket; no matter how less or more influence they had. In the mean time, there would obviously be some nodes who would have way more influence over the network; but not be present in each blanket. So, if we were to take intersect function, then we had the risk of loosing those node. To ensure these nodes' present in the seed set we had to union all the nodes in the last step.

```

most influenced nodes are: {'429': 0.5, '1632':
0.3333333333333333, '2819': 0.3333333333333333, '4586':
0.3333333333333333, '4349': 0.3333333333333333, '3723': 0.25,
'3353': 0.2, '4159': 0.16666666666666666, '2477': 0.05, '23':
0.0017504973448832218, '989': 0.001253152461661367, '443':
0.0009359482448033333, '960': 0.0007753880856529707, '4312':
0.0007166465640125942, '638': 0.00047776437600839607}
seed set : {'2157', '3415', '4304', '3533', '3604', '994',
'3730', '10', '2555', '4182', '140', '818', '2871', '996'}
total child response probability: 2.5071129412004045
# of zero probability child: 4
% of zero probability child: 11.111111111111111 %

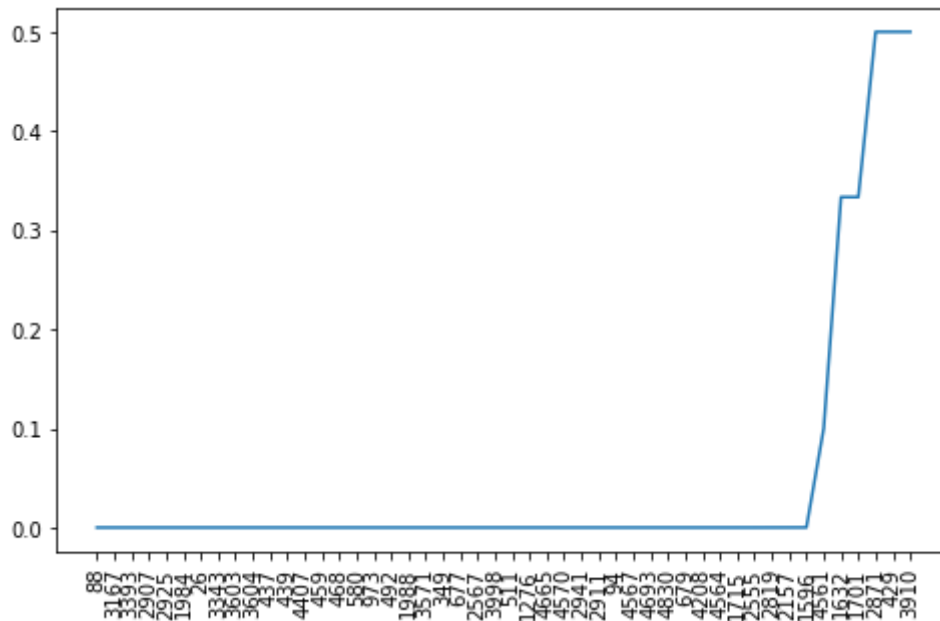
```

Figure 4.21: Seedset Found from Bayesian Inference

Here, we can see, the most influential nodes are given along with their influence on the network. A seed set of 14 nodes can be termed as the most influential nodes of this very network. Total child response probability is 2.5071129412004045.

## 4.4 Comparison

If all child nodes of Kempe’s Greedy Algorithm and Pagerank’s seed set is considered and compared with our Bayesian algorithm, it should proof that a lot of important nodes in PageRank and greedy algorithm have child that oftentimes do not have any probability to interact, i.e. to reply, retweet or mention. Fig 4.22 shows that PageRank algorithm’s seed set have a significantly good number of child nodes that have a 0 or approximately 0 probability of response, i.e. probability of retweets, replies or mentions.



will be. Kempe’s Greedy approach works slowly because it has to check a set of nodes, their neighbors and then calculate the influence value and keep repeating this for a specific pre-set of times for all of the nodes in the graph. That is why it has a NP-hard time complexity. Nonetheless, as PageRank works for a single node at a time, it takes less time to complete the task. Although the model of Kempe considers a given probability and iteratively checks which nodes can activate its neighbor with that given probability but we see a lot of the neighbors of any particle node have often significantly less probability to respond although the seeds are coming up depending on the best activation possible. And thus, even if it takes a huge seed set in consideration, the children’s response probability will be still low. Fig 4.23 shows that clearly.

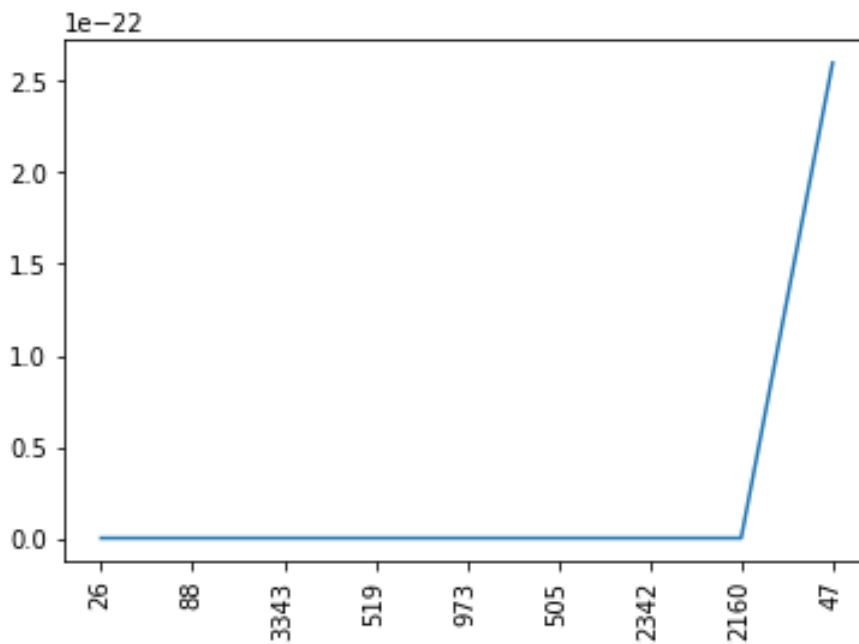


Figure 4.23: Kempe seed set’s child response probability

The Bayesian Inference model takes the interactions among the nodes into account. That is why it works partially different from the other two algorithms. Kempe and PageRank algorithms do not consider any of the criteria which our Bayesian Inference model is taking into account. The number of followers a node has, plays an important role in influencing others in a social network. Besides these, the interactions among them such as, mentions, replies or retweets plays a significant role. Also, Bayesian Inference model considers blankets for every selected node; that is why the influence calculation is more accurate than the former two. These are the reasons why our Bayesian Inference model is worth considering to influence user mining. As in twitter, seed’s child (and sometimes grandchild but no further descendants) is what we are interested in, if we plot the seed set’s child probability of responding, it produces a comparable result between 3 models. As we can see in Fig 4.24, Bayesian’s seed set’s children are way more responding as they have less 0 probability of responding and significantly more >0.3 probability of responding.

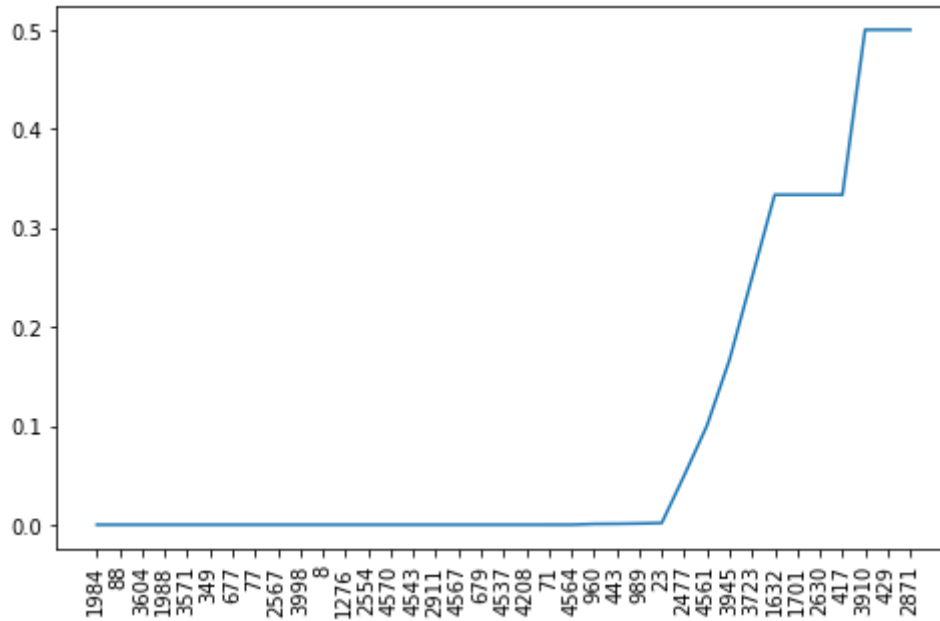


Figure 4.24: Bayesian seed set's child response probability

In Fig 4.25 it is shown what percentage of these 3 models' 1st level children have a zero probability of response, i.e. replies, retweets or mention and therefore sums up the result which shows why PageRank with Bayesian will outperform other models performing solely to find influential users in twitter network.

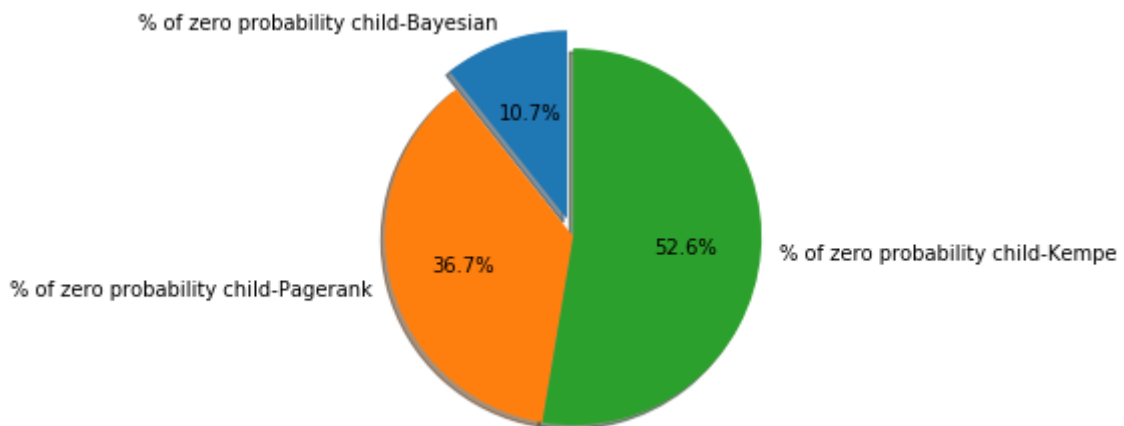


Figure 4.25: % of zero response probability child-among all 3 models

Thus, it proves the Bayesian model will maximize the chances of getting influential users. Clearly the comparison proved that the Bayesian model minimizes getting a seed set that has any child with less responding chances than other two models. Moreover, it enables numerous ways for future work to perform Bayesian models over PageRank that will make sure both the connectivity of the overall network and the response of the child gets prioritized and thus make even better performance for user mining. It may also use Markov blankets for further inferences.



# Chapter 5

## Conclusion and Future Work

Our research mainly focused on how we can find influential users for target marketing in social networks. In this era of technological advancements, like all other daily chores, marketing is also going online. Besides several online markets, social media is the level playing field for everyone. So if we can find the most influential persons and/or groups, then we can easily spread information around us. With a view to finding the most influential users in a social network, we selected "Twitter" as our preferred one. We considered every user as a node. Data were collected from Stanford University. But the data sets were quite huge in number. So we had to take a portion from it. Also, the data were not in the form as we wanted. So we had to pre-process the data; and then make them usable for us. Also we had to use reversed Kruskal Algorithm to remove the cycles and make the graph acyclic. After that three algorithms were separately used to get to the exact result. Of them all, Kempe's Greedy approach and Google's PageRank algorithms are well established. But these two algorithms only used the degrees of a node and the activated neighbouring nodes to calculate their influence over the social network. We do not basically need any followers who don't interact with other users. So, we proposed a model using Bayesian Inference and Modified Markov Blanket algorithm to thoroughly find out the exact seed set who interact with other users. In this research paper, we have used data set of Twitter. The special thing about twitter data set is the connections or edges amongst the nodes can be both unidirectional and bidirectional; and we had to make them in only directional to proceed further.

**Future Work** For future we may consider doing the following things as there is many more scope to improve this work:

- We may use data from other social sites such as Instagram, Tumblr to check our model on them.
- We may improvise our model to work on graphs that has cycles and bidirectional edges amongst the nodes.

# References

- [1] D. Kempe, J. Kleinberg, and É. Tardos, “Maximizing the spread of influence through a social network”, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 137-146, Jul. 2003. DOI: 10.1145/956750.956769.
- [2] W. Chen, C. Wang, and Y. Wang, “Scalable influence maximization for prevalent viral marketing in large-scale social networks”, in *KDD '10*, 2010.
- [3] W. Chen, Y. Yuan, and L. Zhang, “Scalable influence maximization in social networks under the linear threshold model”, Dec. 2010, pp. 88–97. DOI: 10.1109/ICDM.2010.118.
- [4] Y. Zhang, Z. Wang, and C. Xia, “Identifying key users for targeted marketing by mining online social network”, Jan. 2010, pp. 644–649. DOI: 10.1109/WAINA.2010.137.
- [5] A. Goyal, W. Lu, and L. Lakshmanan, “Simpath: An efficient algorithm for influence maximization under the linear threshold model”, Dec. 2011, pp. 211–220. DOI: 10.1109/ICDM.2011.132.
- [6] F.-H. Li, C.-T. Li, and M.-K. Shan, “Labeled influence maximization in social networks for target marketing”, Oct. 2011, pp. 560–563. DOI: 10.1109/PASSAT/SocialCom.2011.152.
- [7] C. Borgs, M. Brautbar, J. Chayes, and S.-H. Teng, “A sublinear time algorithm for pagerank computations”, vol. 7323, Jun. 2012, pp. 41–53. DOI: 10.1007/978-3-642-30541-2\_4.
- [8] S. Naik and Q. Yu, “Maximizing influence of viral marketing via evolutionary user selection”, Aug. 2013, pp. 1435–1436. DOI: 10.1145/2492517.2492580.
- [9] N. Abadi and M. Khayyambashi, “Influence maximization in viral marketing with expert and influential leader discovery approach”, Apr. 2014, pp. 1–8, ISBN: 978-1-4799-4633-4. DOI: 10.1109/ECDC.2014.6836763.
- [10] Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. Yu, “Influence maximization over large-scale social networks”, Nov. 2014, pp. 171–180. DOI: 10.1145/2661829.2662009.
- [11] S. Chen and K. He, “Influence maximization on signed social networks with integrated pagerank”, Dec. 2015, pp. 289–292. DOI: 10.1109/SmartCity.2015.86.
- [12] D. Kempe, J. M. Kleinberg, and É. Tardos, “Maximizing the spread of influence through a social network”, *Theory of Computing*, vol. 11, pp. 105–147, 2015.

- [13] J.-R. Lee and C.-W. Chung, “A query approach for influence maximization on specific users in social networks”, *Knowledge and Data Engineering, IEEE Transactions on*, vol. 27, pp. 340–353, Feb. 2015. DOI: 10.1109/TKDE.2014.2330833.
- [14] S. Jendoubi and A. Martin, “A reliability-based approach for influence maximization using the evidence theory”, Jun. 2017, pp. 313–326, ISBN: 978-3-319-64282-6. DOI: 10.1007/978-3-319-64283-3\_23.
- [15] A. Talukder, M. G. R. Alam, A. Bairagi, S. Abedin, A. Layek, N. Tran, and C. S. Hong, “An approach of cost optimized influence maximization in social networks”, Sep. 2017, pp. 354–357. DOI: 10.1109/APNOMS.2017.8094146.
- [16] A.-s. Olanrewaju, R. Ahmad, and M. Mahmudin, “Influence maximization towards target users on social networks for information diffusion”, May 2018, pp. 842–850, ISBN: 978-3-319-59426-2. DOI: 10.1007/978-3-319-59427-9\_87.
- [17] A. Talukder, M. G. R. Alam, N. H. Tran, and C. S. Hong, “A cost optimized reverse influence maximization in social networks”, in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.
- [18] H. Huang, H. Shen, Z. Meng, H. Chang, and H. He, “Community-based influence maximization for viral marketing”, *Applied Intelligence*, vol. 49, Jan. 2019. DOI: 10.1007/s10489-018-1387-8.
- [19] A. Talukder, M. G. R. Alam, N. Tran, D. Niyato, and C. S. Hong, “Knapsack-based reverse influence maximization for target marketing in social networks”, *IEEE Access*, vol. PP, pp. 1–1, Apr. 2019. DOI: 10.1109/ACCESS.2019.2908412.