

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

Fall 2021

Active Community Opinion Network Mining and Maximization through Social Networks Posts

Mayank Semwal
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>



Part of the [Artificial Intelligence and Robotics Commons](#)

Recommended Citation

Semwal, Mayank, "Active Community Opinion Network Mining and Maximization through Social Networks Posts" (2021). *Electronic Theses and Dissertations*. 8883.

<https://scholar.uwindsor.ca/etd/8883>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Active Community Opinion Network Mining and Maximization through Social Networks Posts

By

Mayank Semwal

A Thesis

Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2021

© 2021 Mayank Semwal

Active Community Opinion Network Mining and Maximization through Social Networks Posts

By

Mayank Semwal

APPROVED BY:

A. Sarkar

Department of Mathematics and Statistics

P. Zadeh

School of Computer Science

C. Ezeife, Advisor

School of Computer Science

August 30, 2021

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Opinion Mining plays a vital role in social networking sites like Twitter, where users share their opinions, and by analyzing the responses (likes, comments, and shares), we can obtain product popularity. Opinion Maximization (OM) is a variant of influence maximization (IM). Unlike IM, only a small set of potential users share positive opinions about the product (e.g., iPhone 12) to maximize the overall opinion spread as early adopters toward a target product while imposing a budget constraint. In contrast, negative opinions can discredit a product's overall popularity.

Existing OM systems like CONE take a partial historical rating of users on multiple products and perform opinion estimation to maximize overall positive opinions using OM. However, CONE does not consider actual user opinions from social posts where users provide opinions through comments, likes and sharing about a product. OBIN mines users' low-frequency features from comments to create a community preference influence network utilizing user response on posts and relationships between them. However, OBIN only performs feature-level opinion mining and does not consider a joint approach that combines sentence-level and feature-level to remove subjective reviews and includes slang words and emoticons, which users often use over the internet. Also, OBIN discovers community preference but does not perform OM, which is more profitable to the seller when introducing a product in the market. The limitation of CONE and OBIN is that they consider opinion mining and maximization as separate subtasks that require more training time and do not consider community opinion, influence among the community users, nor use opinion maximization on their network to minimize viral marketing budget for selecting most influential nodes.

This thesis proposes Active Community Opinion Network Mining and Maximization (ACOMax), an extension of the OBIN system that adds active OM and joint opinion mining for solving two tasks (feature and sentence opinion mining) to enhance the model's accuracy by reducing training time. ACOMax first performs mining of multiple posts related to the product using TwitterAPI while considering relationships between users. Second, opinion mining (positive and neutral) from user reviews on selected posts to perform (i) Sentence-level mining to determine the overall positive sentiment of subjective opinions using VADER. (ii) feature-level opinion mining to extract frequent features with a favourable opinion about the product using the Apriori algorithm. Third, construct an opinion network graph of users who share positive opinions from (ii) to be utilized by the seller to actively select top k seed users with maximum opinion spread under Multiple Linear Threshold (MLT) for opinion maximization. To evaluate our model's performance, we extracted real-time user data using TwitterAPI. Our proposed model (ACOMax) outperforms previous models for total opinion spread in terms of F1 and Accuracy with the help of joint opinion mining and solves the cold start problem of CONE, and improves the total opinion spread in a social network.

Keywords: Opinion Maximization, Social Network, Sentiment Classification, Opinion Mining, Community Detection, Collaborative filtering, Influence maximization, Frequent pattern mining

DEDICATION

I would like to dedicate this thesis to my parents, supervisor Dr. Ezeife, internal and external readers, and my friends who have helped and supported to complete my graduate study at the University of Windsor.

ACKNOWLEDGEMENT

My sincere appreciation goes to my parents Mr. Vijay Prakash Semwal and Mrs. Chandrakala Semwal. Your love, faith and words of encouragement gave me the extra energy to see this work through.

I would like to express my sincere gratitude to my advisor Prof. Dr. Christie Ezeife for her continuous support throughout my graduate study. She always provided me a chance to grow and further enhance research skills. Your constructive criticism, advice and support always gave me the drive to do the quality research work and successfully complete the work. I also like to thank Dr. C. I. Ezeife for research assistantship positions from NSERC grants.

Besides my advisor, I would like to thank my thesis committee members: Dr. Animesh Sarkar (my external reader), Dr. Pooya Moradian Zadeh (my internal reader) and Dr. Saeed Samet (thesis defense chair) for accepting to be my thesis committee, despite their tight schedules and their insightful comments and encouragement is highly appreciated.

Finally, I would express my appreciation to all my friends and colleagues at the University of Windsor, for their advice, encouragement, and support throughout the duration of this work.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	iii
ABSTRACT.....	iv
DEDICATION.....	v
ACKNOWLEDGEMENT.....	vi
LIST OF TABLES	ix
LIST OF FIGURES.....	x
LIST OF EQUATIONS.....	xi
CHAPTER 1: INTRODUCTION.....	- 1 -
1.1 What is a Social Network –.....	- 1 -
1.2 Social Network Graph and Properties	- 1 -
1.3 Social Network Analysis (SNA) –.....	- 3 -
1.4. Influence Maximization (IM) in Social Networks -.....	- 6 -
1.4.1 Diffusion Models.....	- 6 -
1.5. Opinion Mining and Sentimental Analysis	- 9 -
1.6 Active Opinion Mining and Maximization	- 11 -
1.6.1 Joint Opinion Mining.....	- 12 -
1.6.2 Community Opinion Network	- 13 -
1.6.3 Opinion Maximization (OM).....	- 13 -
1.7 Data Mining for Community detection in Social Networks Analysis.....	- 14 -
1.7.1 Clustering.....	- 14 -
1.7.2 Classification	- 15 -
1.7.3 Association Rule Mining	- 16 -
1.7.4 Apriori algorithm (used for identifying frequent feature sets)	- 17 -
1.8 Problem Definition	- 18 -
1.9 Thesis Contribution.....	- 18 -
1.9.1 Thesis Feature Contribution	- 19 -
1.9.2 Thesis Procedural Contributions	- 20 -
1.10 Outline of Thesis.....	- 21 -
CHAPTER 2: RELATED WORK	- 22 -
2.1 Feature-based opinion summarization mining or FBS (Hu & Lu, 2004)	- 22 -
2.2 Greedy algorithm for Influence Maximization (Kempe, Kleinberg & Tardos, 2003)....	- 24 -
2.3. ‘Lazy Forward’ or CELF Optimization (Leskovec et al., 2007).....	- 27 -

2.4. Discovering Influential Nodes from Social Trust Network by Ahmed & Ezeife, 2013 (Trust-General Threshold, T-GT)	28 -
2.5 Social Network Opinion and Posts Mining for Community Preference Discovery by Mumu & Ezeife, 2013 (OBIN)	30 -
2.6. Multi-Round Influence Maximization by Sun, Huang, Yu & Chen, 2018 (MRIM) -	36 -
2.7. Active Opinion Maximization in Social Networks by Liu, Kong & Yu, 2018 (CONE)-	38 -
2.9. Comparison of the existing models	40 -
CHAPTER 3: The Proposed Active Community Opinion Network Mining and Maximization (ACOMax) System Through Social Network Posts	42 -
3.1 Problem Definition	42 -
3.2 Proposed Active Community Opinion Mining and Maximization (ACOMax) System -	43 -
3.3 Social Post Miner (SPOM)	52 -
3.4 Joint Sentiment Analysis of opinions using Opinion Polarity Miner (OPM)	54 -
3.4.1 Preprocessing	55 -
3.4.2 Identification of Opinion Words.....	57 -
3.4.3 Sentence-level Opinion Mining.....	57 -
3.4.3.1 VADER (Valence Aware Dictionary for sEntiment Reasoning)	58 -
3.4.3.2 Slang word replacement & Emoji detection:.....	59 -
3.4.4 Feature Identification using Apriori Algorithm.....	59 -
3.4.5 Extraction of Opinion Words & Semantic Orientation	61 -
3.4.6 Generating Community Opinion Network Graph:	64 -
3.5 Greedy Algorithm for Opinion Maximization under Multi Linear Threshold (MLT) model -	67 -
3.6 A Walkthrough Example with Comparison from OBIN:	69 -
CHAPTER 4: EXPERIMENTAL EVALUATION AND ANALYSIS	76 -
4.1 Dataset Selection.....	76 -
4.2 Evaluation Measures	77 -
4.3 Complexity Analysis.....	82 -
4.4 Implementation and Coding.....	83 -
CHAPTER 5: CONCLUSION AND FUTURE WORK	84 -
REFERENCES.....	85 -
VITA AUCTORIS	94 -

LIST OF TABLES

Table 1: User information from social network.....	2
Table 2: Relationship information of users from Table 1.....	2
Table 3: Transactional data to mine by Apriori algorithm	17
Table 4: Social network data with Influence Probability.....	25
Table 5: Example of Topic Categories of OBIN	32
Table 6: Example of relevant nodes and data for $z = \text{iPhone}$	32
Table 7: Example of Social Post Data	34
Table 8: Facebook User Table	34
Table 9: Social Posts Table.....	34
Table 10: Facebook Comments table.....	34
Table 11: Example of sample dataset for user comments.....	35
Table 12: Example data in Facebook Comment table	36
Table 13: Example data in Facebook Posts table.....	36
Table 14: a) Partial observed rating matrix, b) Filled observed rating matrix.....	39
Table 15: Comparison of Existing Models	41
Table 16: Profile matrix for $p = \text{iPhone12}$	47
Table 17: Updated Profile matrix for $p = \text{iPhone12}$	47
Table 18: Enhanced Profile matrix PM	48
Table 19: Preprocessed user opinions from PM	48
Table 20: Sentence Polarity Score	49
Table 21: Frequent features FFT.....	49
Table 22: Feature features Polarity Score	47
Table 23: Profile matrix of posts for $p = \text{iPhone12}$	53
Table 24: SPOM: Final PM table after extraction	54
Table 25: Data preprocessing of comments.....	56
Table 26: PM containing users with positive opinions about $p=\text{iPhone12}$	64
Table 27: User-User relationship with influence	67
Table 28: OBIN vs proposed ACOMax comparison example	74
Table 29: Comparison of 99% CI for different number of discovered influential nodes	79
Table 30: Comparison of discovering influential nodes by OBIN and ACOMax.....	79
Table 31: Complexity Analysis	82

LIST OF FIGURES

Figure 1: Example of Directed and Undirected Graph.....	2
Figure 2: Graph model of social network data from Table 1 and 2.....	3
Figure 3: An example of Social Network Graph.....	3
Figure 4: Six-degree separation.....	3
Figure 5: Types of Social Network Mining Tasks.....	4
Figure 6: Community detection architecture.....	5
Figure 7: Linear Threshold model (LT) example.....	7
Figure 8: Independent Cascade Model (IC) example.....	9
Figure 9: Opinion Mining vs Sentiment Analysis.....	9
Figure 10: Factors that influence customers buying decision.....	10
Figure 11: Joint Opinion Mining Approach.....	12
Figure 12: Clustering Example.....	15
Figure 13: Classification in Influence Maximization (Hu, Wang & Yu, 2014).....	16
Figure 14: Bipolar adjective structure, synonym, and antonym.....	23
Figure 15: Social Network Graph with influence probability modeled on data in Table 4.....	25
Figure 16: The Greedy k-best influence maximization algorithm.....	25
Figure 17: Running time of exhaustive search, greedy and CELF (Leskovec et al., 2007).....	27
Figure 18: T-GT trust and Influence graph.....	29
Figure 19: Example of CONE for opinion maximization (Liu, Kong & Yu, 2018).....	40
Figure 20: Proposed ACOMax Architecture.....	43
Figure 21: Community Influence Graph without probabilities.....	66
Figure 22: Community opinion network with probabilities.....	67
Figure 23: Graph G for greedy algorithm.....	68
Figure 24: Comparison between OBIN and Proposed ACOMax.....	69
Figure 25: Tweets vs Influence Rank.....	80
Figure 26: Comparison of opinion spread under opinion maximization (CONE vs ACOMax) ..	81
Figure 27: Time complexity (CONE vs ACOMax).....	83

LIST OF EQUATIONS

Equation 1: Calculate Closeness between nodes in a social graph	5
Equation 2: Linear threshold calculation	7
Equation 3: Apriori Equation to compute support of itemset i.....	16
Equation 4: Apriori Equation to compute confidence of itemset i	16
Equation 5: Matrix factorization for learning user and item matrix	38
Equation 6: Inner product to estimate rating vector.....	39
Equation 7: Calculate Compound score of a sentence	59
Equation 8: Calculate overall score of feature opinion.....	64
Equation 9: Calculate Influence probability between users.....	68
Equation 10: Calculate Recall score of relevant nodes.....	77
Equation 11: Calculate Precision score of relevant nodes	78
Equation 12: Calculate F1-Score	78

CHAPTER 1: INTRODUCTION

The online data that led us to the era of Big data is created with the service of social media platforms such as Instagram (<https://www.instagram.com/>), YouTube (www.youtube.com/) and Facebook (<https://www.facebook.com/>) in which a user interacts and shares information with other users on diverse subject matters. Discovering communities in a social network means recognizing a set of nodes/users communicating with each other in the form of comments, likes and shares, which leads to identification of influential posts, influential users, users' opinions analysis based on shared interest and may extend product advertisement in a network (Parthasarathy, Ruan & Satuluri, 2011).

1.1 What is a Social Network – Social Network is a network of interactions or relationships, where the node consists of users, and the edges consist of the relationships or interactions between the users, which are interconnected by various types of relationships, such as friendship, trust etc. (Bonchi et al., 2011). Social Network Analysis (SNA) concentrates on techniques to analyze these relationships and information flows between nodes in a social network and produce models that facilitate understanding the network structure. Researchers like (Kempe, Kleinberg & Tardos, 2003) and (Leskovec et al., 2007) are actively studying and understanding social networks' properties and their structures and their challenges by applying various data mining and machine learning methods to these data. The best example is the task of influence maximization and opinion mining, which is the problem of detecting a small subset of social network graph that could maximize the spread of influence (Kempe, Kleinberg & Tardos, 2003).

1.2 Social Network Graph and Properties - The social network graph G is a pair (V, E) , where V is a set of vertices (or nodes), and E is a set of edges between the vertices $E \subseteq \{(u, v) \mid u, v \in V\}$. Social networks are modelled as either directed or undirected graph. Figure 1(a), a node can point to another node but not vice versa, and an edge connects them; such relationships in a social network are modelled as an undirected graph. Figure 1(b), with edges represented by arrows, consists of a set of nodes with a set of directed edges; that is, each directed edge is a link from one node to another.

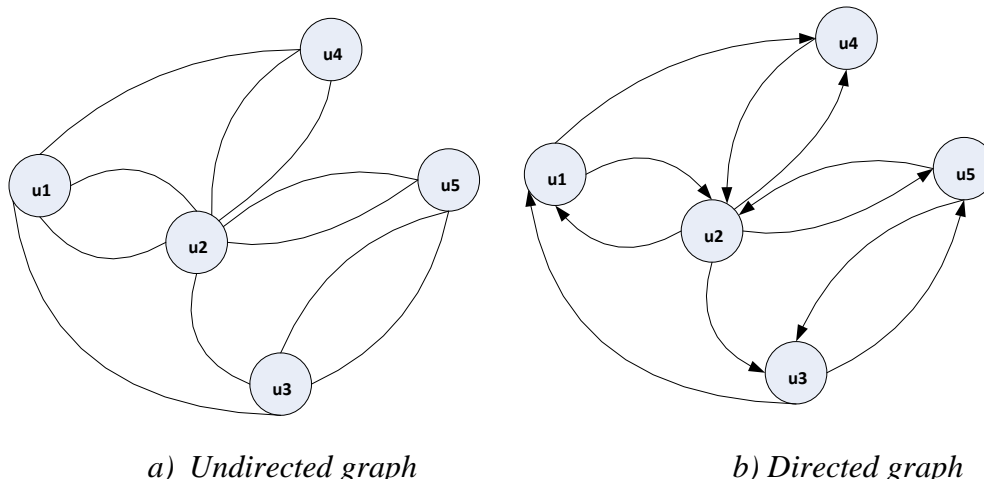


Figure 1: Example of Directed and Undirected Graph.

The social interactions/links between nodes exist in two ways:

- 1) **Explicit**, e.g., user's declaring their friends or connections directly through actions like joining a group, liking a page, following a user, or accepting a friendship request.
- 2) **Implicit**, e.g., links identified from users' activities by analyzing broad and repeated interactions between users such as voting, sharing, tagging, or commenting (Bonchi, Castillo, Gionis & Jaimes, 2011).

Example of a scenario of social network: Let us consider a social network data table, table 1 consist of a list of individuals in a social network and table 2 reports friendship relationship among these individuals.

Table 1: User information from social network

User_id	Name	Age	Sex	Location
101	Mike	28	M	California
102	Komal	26	F	Vancouver
201	Ishita	22	F	Toronto
301	Anurag	29	F	New York

Table 2: Relationship information of users in Table 1

User_id	Friend_Of	DateCreated
101	301	12-Mar-2013
301	201	22-Apr-2015
101	102	14-Nov-2016
102	301	02-Dec-2017

Based on data above we can model a simple social network graph as shown in Figure 2.

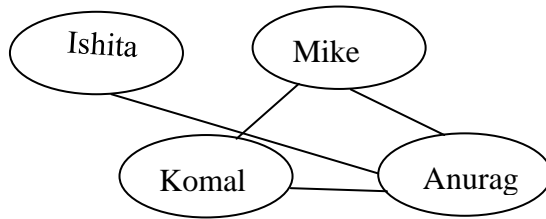


Figure 2 - Graph model of social network data in Table 1 and 2.

In the above graph, $G(V, E)$, V is the set of individuals (or vertices) in the social network, and E is the set of all friendship links (or edges).

$V = \{Mike, Anurag, Komal, Ishita\}$

$E = \{(Ishita, Anurag), (Mike, Komal), (Komal, Anurag), (Mike, Anurag)\}$.

1.3 Social Network Analysis (SNA) – Social network analysis is to extract a network of interactions or relationships from different communication resources to analyze structures and influencing users of social networks. Social network mining tasks use various graph-based proximity measures for mining and analyzing social network data. SNA can be classified into two major categories: descriptive or predictive (Figure 4), below we will analyze types in descriptive and predictive social network mining.

Why social influence under social network analysis is important?

- Social six-degrees of separation is the idea that all people on average are six, or fewer, social connections away from each other. As a result, "friend of a friend" statements can be made to connect any two people in a maximum of six steps.

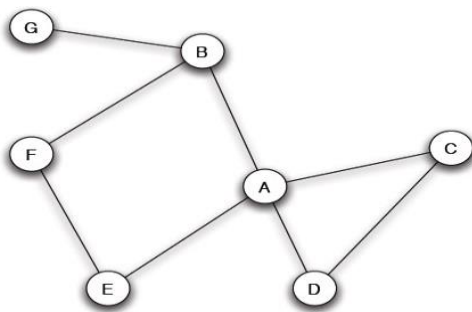


Figure 3 - An example of Social Network Graph.

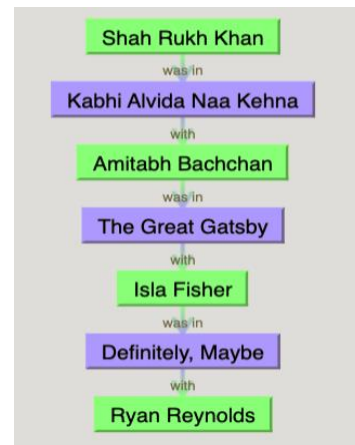


Figure 4: Six-degree separation

Figure 4 shows two movie actors from India (Shah Rukh Khan) and Canada (Ryan Reynolds) are separated by just 3 degrees. The graph G in figure 3 have 7 vertices as follows:

$V = \{A, B, C, D, E, F, G\}$ and 8 edges as follows,

$E = \{(A, B), (A, D), (A, C), (A, E), (F, E), (F, B), (G, B)\}$.

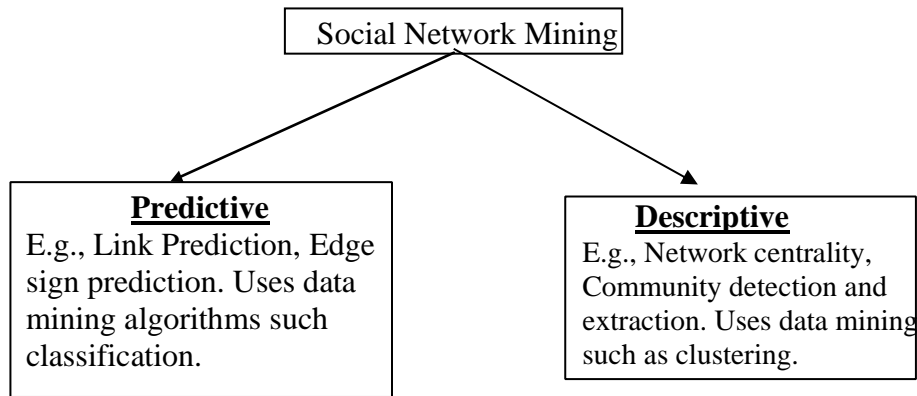


Figure 5 – Types of Social Network Mining Tasks

1). A predictive model predicts unknown values. For example, predicting whether an individual will be a friend of another individual. Some predictive mining of social networks is listed below:

- **Link Prediction:** The link prediction task in a social network is to predict an edge between two nodes. More formally, given a snapshot of a social network at time t , the link prediction task seeks to accurately predict which edges will be added to the network at time t' . Link Prediction is a crucial component of the friend recommendation system in many social network sites such as Facebook.
- **Node Classification:** In large social network graphs, such as online social networks like Facebook, a subset of users or nodes may be labelled with information that indicates demographic values, interest, beliefs, or other characteristics of the users.

2). A descriptive model identifies patterns or relationships, such as trends, clusters, and anomalies in data. Some descriptive mining tasks in a social network are listed below:

- **Degree:** The degree of a vertex in a graph can be denoted as $D(v)$, which is the number of edges on that vertex. Let us consider the social network graph in figure 3. In this graph, the degree of node A is 4 and node B is 3. The notion of a degree in a directed graph is a bit different. For example, in the directed graph in figure 1 b, there are 2 types of degrees for node B, namely in-degree, which is 2, because edges from nodes A and D are directed towards node B. Similarly, node B's out-degree is 1 as it has 1 edge, to node C, away from itself.

- **Community Detection:** The goal of the community detection task in social network mining is to detect groups or communities in the social network graph with more (or dense) edges among nodes in the same group than that of among nodes outside the group.

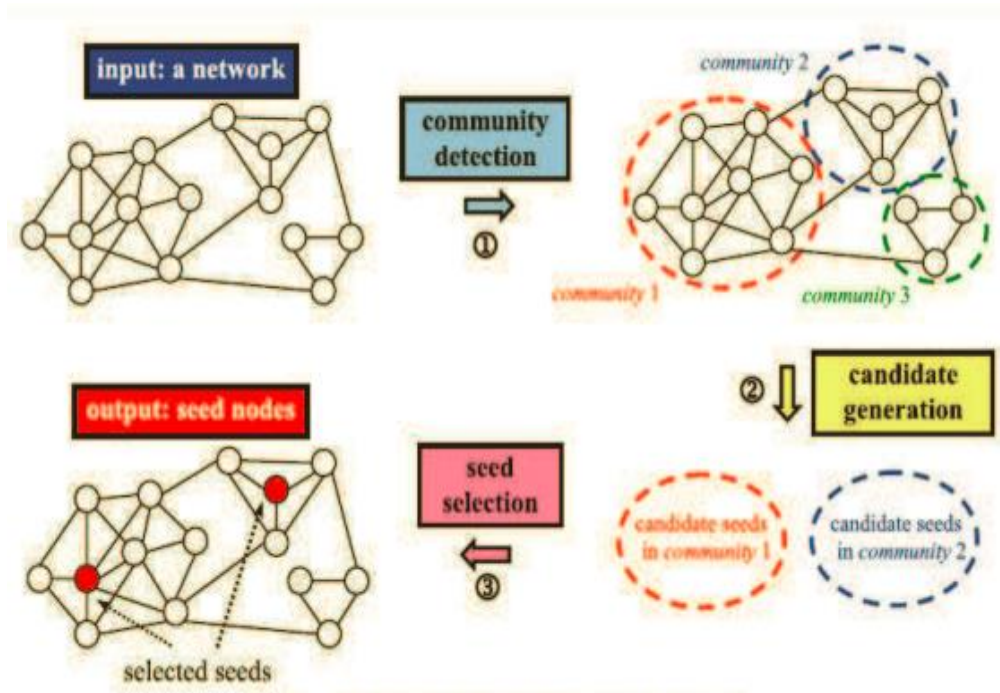


Figure 6 - Community Detection Architecture

- **Influence Maximization (IM):** The influence maximization task attempts to help companies determine potential customers in the market so that by mouth-to-mouth word spread about the product, these customers can influence a higher number of customers. Example, when Hotmail launched, it grew from 0 users to 12 million users in just 18 months on a tiny advertising budget.
- **Closeness:** Closeness centrality is a way of detecting nodes that can spread information very efficiently through a graph. The closeness centrality of a node measures its average farness (distance) to all other nodes. Nodes with a high closeness score have the shortest distances to all other nodes. The closeness score of node x is calculated using equation 1, where N is the number of nodes in $G (V, E)$:

$$(\sum_{y \in V} d_{xy}) \div (N - 1)$$

Equation 1: Closeness of node x calculated

1.4. Influence Maximization (IM) in Social Networks - Analyzing information diffusion and social influence in a social network has many applications to the real-world. Influence maximization (IM) for viral marketing is an example of such a critical application. In IM, a small set of users are selected, and these users can start a chain reaction of influence with a small marketing budget to influence a large population of a social network (Ahmed & Ezeife, 2013). Existing IM, such as Linear Threshold Model (LT) and Independent Cascade Model (IC) by (Kempe, Kleinberg & Tardos, 2003), formulated the influence maximization as a discrete optimization problem and solved it using a greedy algorithm. 'Lazy forward' optimization (Leskovec et al., 2007) is about 700 times faster than greedy (Kempe, Kleinberg & Tardos, 2003). These models, or their variations, are the most used diffusion models in Influence Maximization.

1.4.1 Diffusion Models - A diffusion model, also known as propagation model, determines how the influence propagates through the network and is used in discovering communities and performing influence maximization. This diffusion model's role is to replicate or simulate a real-life diffusion process and determine which nodes and how many nodes will be activated by any given set of nodes (called seed nodes) after the diffusion process is over. In a social graph $G(V, E)$, a vertex $v \in V$ is active if the information has reached the vertex and was accepted by it. Similarly, a vertex where information has not been reached or got convinced so far is called inactive. It is assumed that an inactive vertex can become active during the process of diffusion of the information but not vice-versa. For the diffusion process to start, there must be some initial set of active nodes called seed nodes, which are initially activated. In this section, the two of the most well-known models, namely the LT and IC by (Kempe, Kleinberg & Tardos, 2003), are explained.

Definition of Influence Spread during diffusion process by (Kempe, Kleinberg & Tardos, 2003): Given an initial active set S_0 , the "influence spread", "influence", or just "spread" of S_0 , denoted as $\sigma(S_0)$, is defined to be the expected number of active nodes influence by S_0 at the end of the diffusion process when no more adoptions are possible. Here, $\sigma(\cdot)$ is a function, defined as $\sigma : 2^V \rightarrow \mathfrak{R}$, mapping a seed set to a real number (the expected number of nodes influence by the seed set to adopt an innovation at the end of the diffusion). On the other hand, the verb "influence" (as in node v influences node u) means " v activates node u ".

1.4.1.1. Linear Threshold (LT) Model – Given a directed social network graph $G(V, E)$, where nodes V representing users of the network and E represents the relationship/interaction (e.g., comments, likes, shares). A node u is influenced by each neighbor v according to a weight (Kempe et al., 2003).

$$b_{u,v} = \sum_{(u \text{ neighbors of } v)} b_{u,v} \leq 1$$

Each node v chooses a threshold uniformly at random $\theta_v \sim U[0,1]$. The threshold of a node v is defined as the minimum proportion of its neighbors who will already influence the v to adopt the behavior. For example, suppose v 's threshold is 25%, v has 100 neighbors, and 26 of them have liked the product, since $26/100 = 26\% > 25\%$, v will adopt the product too. Each edge (u, v) in E is also assigned with a weight value $b_{u,v}$. The vertex u is influenced by each of its neighbours v according to the weight $b_{u,v}$ such that sum of all the weights $b_{u,v}$ for all $u \in N(v)$ is ≤ 1 where Nv is set of all active neighbours of v . The diffusion process happens in discrete steps, i.e., $t = 0, 1, 2, \dots, n - 1$. At any time t , each node $v \in V$ is either active or inactive. One v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior. At time $t > 0$, all nodes that were active at time $t-1$ will remain active, any inactive node u is activated if the total weight of its active neighbors is no less than its threshold:

$$\sum_{(u \text{ neighbors of } v)} b_{u,v} \geq \theta_v$$

Equation 2: Linear threshold calculation

Example: Figure 7 to illustrate how the Linear Threshold Model works. Let S_t denote the set of active nodes at time t , $t = 0, 1, 2, \dots, n-1$. Then S_{t-1} denotes the set of inactive nodes at time t .

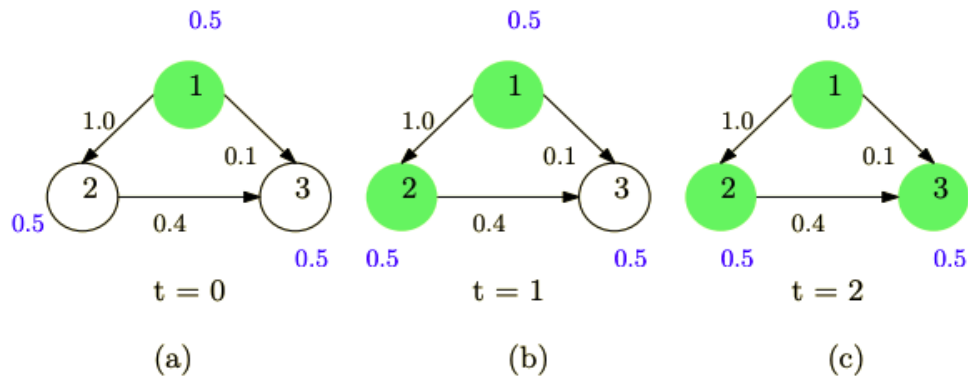


Figure 7– Linear Threshold Model Example

Step 1: At time $t=0$ (Figure 7 (a)), there is a social network $G = (V, E)$, along with an initial set of active nodes, i.e., $S_0 = \{1\}$.

Step 2: At time $t=1$ in (Figure 7 (b)), node 1 activates node 2 with influence probability $p_{1,2} = 1.0$ on node 2 and threshold of node 2 is $\theta_2 = 0.5$. So, node 2 gets activated because influence probability ($b_{u,v}$) $> \theta$ (equation 2) of node 1 is higher than the threshold of node 2, but fails to activate node 3 since $p_{1,3} = 0.1$ and $\theta_3 = 0.5$.

Step 3: At time $t=2$ (Figure 7(c)), node 1 and 2 jointly activate node 3 since $p_{1,3} + p_{2,3} = 0.1 + 0.4 = 0.5$, and $\theta_3 = 0.5$. At this point, the diffusion stops since no more activations are possible. From Figure 6(c), we can see the influence spread of $\{1\}$ is 3, the number of active nodes at the end of the diffusion.

1.4.1.2. Independent Cascade (IC) Model – The Independent Cascade model represents a social network as a weighted, directed graph $G = (V, E)$. Each edge $(v, u) \in E$ is assigned a non-negative probability $p_{u,v}$ indicating the influence that node v exerts on node u , that is if v is active, it succeeds in activating u with the probability of $p_{u,v}$. Unlike LT which assigns the threshold to propagate the influence, IC assigns influence probability to each node. The diffusion process happens in discrete steps, i.e., $t = 0, 1, \dots, n-1$. At any time t , each node $v \in V$ is either active or inactive. Once v is activated, it remains active and cannot switch back to inactive. At time 0, there is an initial set S_0 that adopts a new behavior, and the diffusion process unfolds as follows. If a node v is active, it is given one single chance to activate each of its inactive neighbors u with probability of $p_{u,v}$. The diffusion process will stop when no more activations are possible (Kempe et al., 2003).

Example. We use Figure 8 to illustrate how the Independent Cascade model works. Let S_t denote the set of active nodes at time t , $t = 0, 1, 2, \dots, n - 1$, with $S_{-1} = \emptyset$. Then S_{t-1} denotes the set of inactive nodes at time t . A node u gets activated if: **a)** $p_{u,v} > \text{initially set threshold (suppose 0.3)}$
b) head/tail before activating

Step 1: At time 0 (Figure 8 (a)), there is a social network $G = (V, E)$, along with an initial set of active nodes, i.e., $S_0 = \{1\}$.

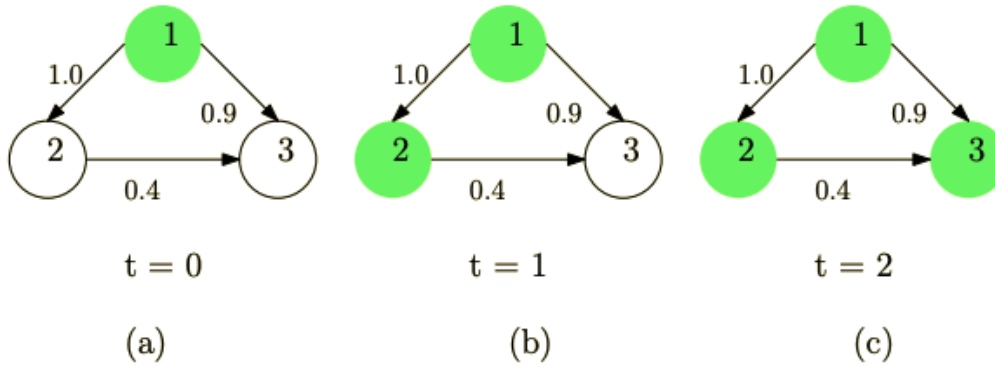


Figure 8– Independent Cascade Model example.

Step 2: At time 1, node 1 activates node 2 with the probability $p_{1,2} = 1.0$ for the influence propagation from node 1 to node 2 but fails to activate node 3 with the probability $p_{1,3} = 0.9$ for the influence propagation from node 1 to node 3 (Figure 8 (b)).

Step 3: At time 2, nodes 2 activates node 3 with the probability $p_{2,3} = 0.4$ for the influence propagation from node 2 to node 3 (Figure 8 (c)). At this point, the diffusion stops since there are no more activations possible. From Figure 8 (c), we can see the influence spread of $\{1\}$ is 3, the number of active nodes at the end of the diffusion.

1.5. Opinion Mining and Sentimental Analysis - The fields of opinion mining and sentiment analysis are distinct but deeply related. Opinion Mining extracts and analyzes people’s opinion about an entity and identifies features while sentiment analysis (SA) identifies the sentiment expressed in a text. Therefore, the target is to find opinions, identify the sentiments they express, and then classify their polarity.

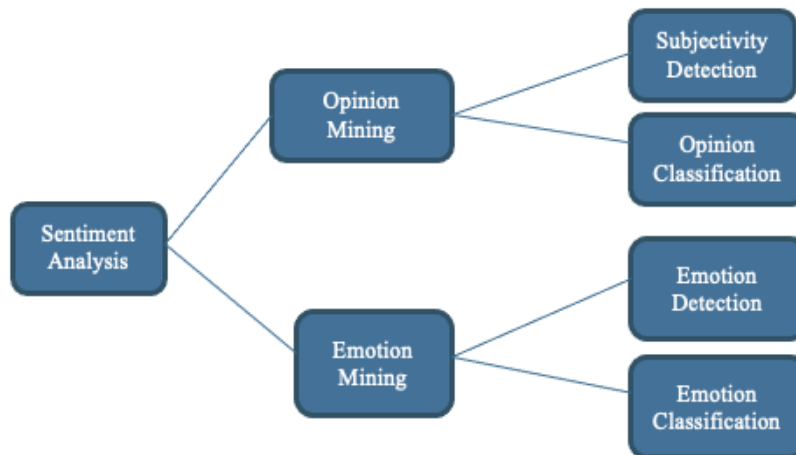


Figure 9- Opinion Mining vs Sentiment Analysis

Opinion mining and its analysis have been studied extensively, and generally, three research directions are explored, i.e., document-level, sentence-level, and feature-level opinion mining:

a). Document-level (Turney, 2002) presented an approach of determining document's polarity by calculating the average semantic orientation (SO) of extracted phrases. SO was computed by using pointwise mutual information (PMI) to measure the dependence between extracted phrases and the reference words "excellent" and "poor" by using web search hit counts. The limitation of document-level opinion mining is it determines the documents' sentiment without performing extraction and classification on entities. Another is they are not focused on features being commented on the posts.

b). Sentence-level (Liu, Yu, Liu & Chen, 2014), The aim is to determine whether opinions expressed in sentences with modality are positive, negative, or neutral. Modality is commonly used in text. For example, in the sentence, “this cellphone would be perfect if it has a bigger screen”, the speaker is negative about this phone although there is a typically positive opinion word “perfect” in this sentence.

c). Feature-level, models classify and summarize reviews by extracting high-frequency feature keywords and high-frequency opinion keywords. Feature-opinion pairs were identified by using a fixed list of keywords to recognize high-frequency feature words. Research by (Hu & Liu, 2004) extended feature-level into a statistical approach capturing high-frequency feature words using association rule mining. Infrequent feature words are captured by extracting known opinion words' adjacent noun phrases.

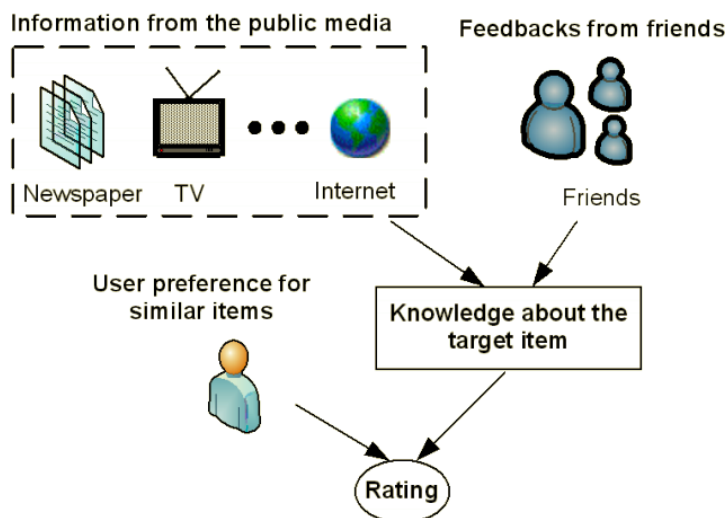


Figure 10: The three factors that influence a customer’s buying decision: a) user preference for similar items, b) information regarding the target item from the public media, c) feedbacks from friends

However, (Hu & Liu, 2004) approach could not address in-frequent features effectively. To solve this problem, OBIN (Mumu & Ezeife, 2014) uses the OpinionMiner (Jin, Ho & Srihari, 2009) framework. OBIN solved the issue of in-frequent features and solved issues to 1) Automatically extract potential product entities and opinion entities from the reviews, 2) Identify opinion sentences that describe each extracted product entity, 3) Sentiment analysis to determine opinion orientation (positive or negative) given each recognized product entity. OpinionMiner effectively identifies complex low-frequency phrases in the product-specific reviews, new potential product and opinion entities are discovered based on the patterns the classifier has seen from the training data, also included Part-of-Speech (POS, is the process of marking up the words in a text) and Tagging and Named Entity Recognition (NER, is the process of identifying and classifying person names and organization names) in their algorithm for opinion mining.

Sentiment analysis of Negative Opinions: In general, IM methods assume that most of the users are potential customers. However, in the promotion of many products, such as new iPhone, many of the opinions can be detrimental due to the different preference of these users. The spread of negative opinions harms the product's reputation and washes out the positive opinions. In this case, simply maximizing the spread over a social network can no longer achieve an optimal outcome. The goal is to divide the customers in the market into several portions, and a specific marketing strategy should be investigated and designed carefully for each portion of users to maximize the profit.

1.6 Active Opinion Mining and Maximization - Conventional opinion mining approaches assume that the users' opinions can be either estimated or generated based upon their neighbour's opinions (Gionis, Terzi & Tsaparas, 2013). Users' opinions could be estimated from the user's history, or their opinions could be observed once they have been exposed to the product during the propagation process. The opinion estimation process of unknown ratings and influence propagation intertwines together, requiring the model to consider the two components collectively. CONE (Liu, Kong & Yu, 2018) proposed the active learning framework to address the above challenges. In the active learning framework, the selection of seed nodes in information diffusion is interrelated with the estimation of user opinions.

However, the estimated opinion does not reflect users' actual preferences (existing preferences expressed via comments and likes on posts) on the target product. Therefore, opinion

mining and maximization frameworks like CONE and OC (Zhang, Dinh & Thai, 2013) can no longer work. Instead of opinion estimation, we can extract user opinions to reflect their actual preferences from user comments about the product through social network posts using opinion mining. Therefore, the proposed ACOMax uses the active learning framework to consider opinion mining and opinion maximization collectively.

1.6.1 Joint Opinion Mining – The joint opinion mining approach proposed by (Mai & Le, 2020) performs joint sentence and aspect-level sentiment analysis of product comments resulting in (1) capture the mutual benefits between these two tasks, and (2) leverage knowledge learned from solving one task to solve another. Joint approach by (Mai & Le, 2020) shows that the joint model achieves better performance and outperforms separately considered sentence and aspect opinion mining. The joint opinion mining approach by (Belisário, Ferreira & Pardo, 2020) suggests that the combination of subjectivity classification, lexical analysis and feature mining produce better influential users than any of the feature sets individually. The more subjective reviews (inclusion of slang, emoticons) are used for feature mining, the more they will be associated with positive sentiments higher the probability that the overall score is positive.

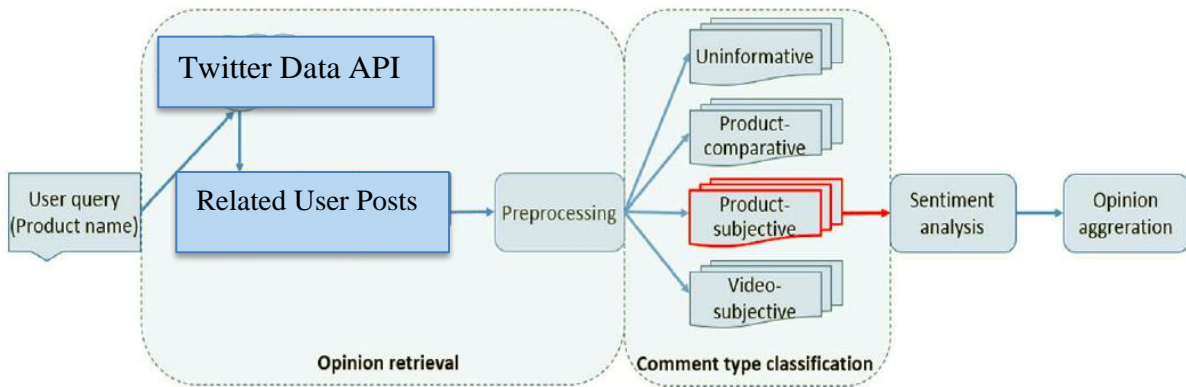


Figure 11: Joint Opinion Mining Approach

Thus, in this thesis, the joint opinion does both sentence subjectivity analysis in conjunction with feature sentiment analysis for opinion classification. For every sentence/review in the dataset, we need to identify the subjective reviews, remove named entities (Apple, iPhone), calculate sentence polarity and only pass opinions with positive polarity for feature-level mining. It aims to predict the overall rating based on users' opinions about the target products through posts.

1.6.2 Community Opinion Network – The idea is to build a trust network of community detected through opinion mining through social network posts. Several researches by (Pang et al. 2002, Turney 2002, Agrawal et al. 2003, Dave et al. 2003, Hu and Liu 2004, Mishne and Glance 2006, Ding et al. 2008, Gomez et al. 2008, Li et al. 2008, Tang et al. 2009) have been done for analyzing user's responses on interest networks (i.e., user-service interaction), but there has been no previous work studying user responses in friendship networks (i.e., user-user connections). Such user-service connections are domain specific and product-feature oriented. For example, these networks may be product review domains. Due to the emerging popularity of friendship networks, discovering common interests shared by users is a fundamental problem in such friendship networks since it is the bread-and-butter function of building user communities of the same interests, finding the topic experts in different subjects, identifying hot social topics, and recommending personalized relevant contents. An efficient and scalable solution is crucial to the growth of social communities. Main tasks in generating the community opinion network after successful mining of user opinions are:

1. Selecting users with positive opinion about the target product on social posts by aggregating users' opinions (e.g., likes, retweets, comments and quotes). Discovering user-user relationships through interactions among selected users on social posts.
2. Calculating user influence probabilities i.e., social influences of users. For example, *user1* may have high influence on *user2* on product *p*. In general, popularity of a product somehow depends on how fast the posted influence spreads in a community (a group of users with similar interest).
3. The crucial characteristic of community opinion network is the overall opinion spread of users towards the target product, for example, whether a product opinion is positive or negative.

1.6.3 Opinion Maximization (OM) – OM is a variant of influence maximization (IM) where the main task is to select a small set of users whose positive opinions about an item can maximize the item's overall positive opinion spread in the social network. OM takes a user's opinions on the target product before the campaign for seed user selection until the budget is used up. Example, users who share positive opinion about the product (e.g., iPhone 12) as early adopters of a target product to maximize the overall opinion spread. In contrast, negative opinions can discredit a product's overall popularity.

In this thesis, we apply the Multi Linear Threshold model for opinion maximization to select the small seed set S to maximize the total opinion spread by users in G (community opinion network graph from step 1.6.2) towards the target product p .

1.7 Data Mining for Community detection in Social Networks Analysis - Data mining is a process of knowledge discovery (KDD). KDD process include a) data selection, b) data pre-processing (integrates target data from various sources and cleans target data by removing noise and inconsistent data), c) data transformation (which summarizes or aggregates the pre-processed data into appropriate forms), d) pattern evaluation and knowledge interpretation (representation or visualization of these interesting patterns discovered). Data mining is closely related to the subareas of machine learning but handles much larger data in an automated fashion with more

1.7.1 Clustering

It is process of grouping a set of related objects in such a way that objects in the same group are like each other (Jain & Dubes, 1998). The process is a measure of similarity between objects and combine similar objects into the same cluster while keeping dissimilar objects in different clusters according to algorithm. The process of clustering decomposes a large-scale system into smaller components. Some clustering techniques include:

1. Partitioning methods such as K-means algorithms (MacQueen et al., 1967). The algorithm consists of simply starting with k groups each of which consists of a single random point, and thereafter adding each new point to the group whose mean the new point is nearest. After a point is added to a group, the mean of that group is adjusted to take account of the new point. In (Soni & Ezeife, 2013), the authors improve the K-means algorithm and propose a novel approach named Semantic non-parametric K-Means++ to automatically move emails from inbox to appropriate folders and sub-folders.

2. Hierarchical methods such as agglomerative approach where each object is placed in its own cluster and then merges these atomic clusters into larger clusters until all the objects are in a single cluster, or divisive approach where all objects are placed in one cluster and then subdivides the cluster into smaller pieces until each object forms a cluster on its own (Hastie et al., 2001). In (Chen et al. 2014), the authors exploit the hierarchical clustering algorithm to improve the

efficiency of mining influence maximization by discovering the community structure of the network to reduce the search space for influential nodes

3. Grid-based methods, cluster data elements of a data stream. Initially, the multidimensional data space of a data stream is partitioned into a set of mutually exclusive equal-size initial cells (Park & Lee, 2004). Example, using clustering technique in web mining Figure 12 shows “automatic storage of emails falling within a certain cluster based on email contents and senders”.

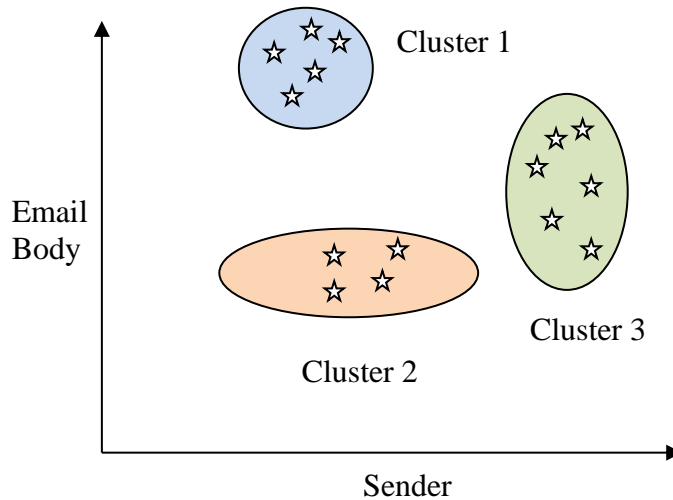


Figure 12 Clustering Example

1.7.2 Classification

Classification is used to classify an item in a set of predefined set of classes or groups. The classification process involves the training set and testing set. The training dataset is used to train model, by pairing the input with expected output. Then, the same classification model is applied to the test data having unknown target class values, to check for its prediction accuracy. Some classification algorithms include nearest neighbours (K-NN) (Cover & Hart, 1967), Naïve Bayes classifier (McCallum et al., 1998), Support Vector Machine (SVM) (Cortes and Vapnik, 1995) and decision trees (Quinlan,1986). In (Hu, Wang & Yu, 2014), the authors propose an algorithm that exploits classification algorithms to tackle the Influence Maximization problem and uses the result of a greedy algorithm to train classifiers to directly select influential nodes based on their features (Figure 13).

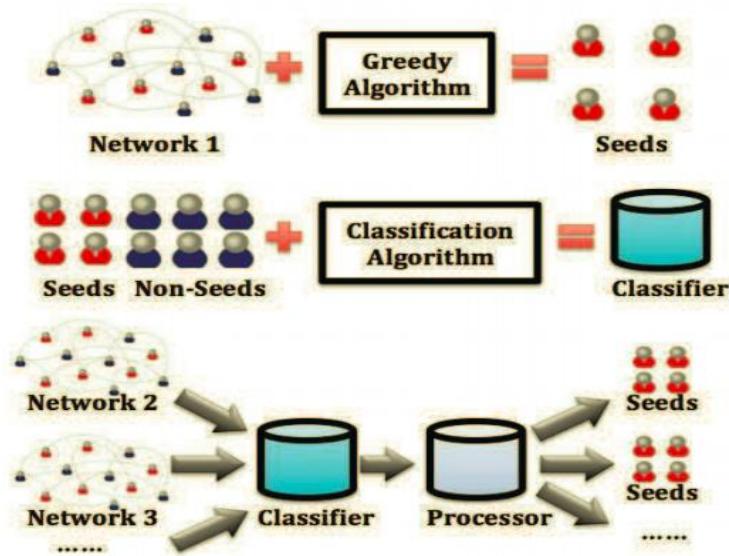


Figure 13 Classification in Influence Maximization (Hu, Wang & Yu, 2014)

1.7.3 Association Rule Mining

Association rules analysis is an unsupervised technique to discover how items are associated with each other (Ma, Hsu & Liu, 2000). The association rule consists of two parts the left-hand side is called antecedent, and the righthand side is called consequent. Association rule is represented in the form $X \rightarrow Y$, where X and Y belong to a candidate set $I = \{i_1, i_2, \dots, i_n\}$ of n items. Association rule is performed in two stages i) finding all frequent patterns (itemsets) having support greater than or equal to minimum support ii) finding all rules from frequent patterns with confidence greater than or equal to minimum confidence. Association rule finds the relationship between the items in the rule. In Association rule, confidence and support are two major factors, which can be computed by Equation and Equation .

$$\text{Support of item } i = \frac{(\text{number of occurrences of } i)}{(\text{number of database transactions})}$$

Equation 3: Equation to compute support of itemset i

$$\text{Confidence of item } i = \frac{(\text{number of occurrences of } i)}{(\text{number of occurrences of the antecedent of the item } i)}$$

Equation 4: Equation to compute confidence of itemset i

Rules that satisfy user-specified minimum support are called frequent items, and if the confidence is greater than a user-specified minimum confidence then we say the rule is accurate. An example

of frequent pattern mining algorithm is Apriori Algorithm (Agrawal & Srikant, 1994), as described below:

1.7.4 Apriori algorithm (used for identifying frequent feature sets): The Apriori algorithm (Agrawal & Srikant, 1994) is a popular algorithm for association rule mining, and it works in two steps i) generate frequent itemsets ii) pruning the itemsets based on the user-defined support. Apriori algorithm takes a transactional database and output is frequent itemsets that satisfied minimum support. So, in the first step, support count of each item in the candidate set (C_1) is calculated, and those items that don't satisfy the minimum support are pruned and produced frequent set (L_1). In the next step, the candidate set (C_2) is produced by Apriori join method by L_1 **App-join** L_1 . This process is iterative until can't produce more candidates set.

Example - Let us consider transactional data as shown in Table 3 as input, where candidate set (C_1) = {A, B, C, D}, minimum support=3, and goal is to find frequent items to create possible association rules.

Step 1: Find frequent item (L_1) from candidate set (C_1): The principal step in Apriori process is to find frequent item by the counting occurrence of each item. The items that don't satisfy the minimum support count are pruned and produced frequent item (L_1). In our case, frequent item (L_1) = {A:3, B:6, C:4, D:5}.

Transaction Id (TID)	Items
T1	A,B,C,D
T2	A,B,D
T3	A,B
T4	B,C,D
T5	B,C
T6	C,D
T7	B,D

Table 3: Transactional data to mine by Apriori algorithm

Step 2: Generate candidate set (C_2) from frequent item (L_1) by Apriori join (L_1 App-join L_1): We can generate a candidate set (C_2) by L_1 App-join L_1 . Frequent item (L_1) can be joined only with an item that comes after it in frequent item (L_1). Which will give candidate set (C_2) = {AB, AC, AD, BC, BD, CD}.

Step 3: Find frequent item (L_2) from candidate set (C_2): Frequent item (L_2) is obtained by following the same procedure as in step 1. We can count the occurrence of each item in candidate

set (C2), and infrequent items are removed to create frequent itemset (L2) = {AB: 3, BC: 3, BD: 4, CD: 3}.

Step 4: Generate candidate set (C3) from frequent item (L2) by Apriori join (L2 App-join L2)

We can apply the same process as in step 2 to generate candidate set (C3) by joining L2 with L2 using Apriori join and it produces candidate set (C3) = {ABC, ABD, BCD}.

Step 5: Find frequent item (L3) from candidate set (C3): None of the item in candidate set (C3) satisfied minimum support. So, we need to stop here and join frequent item to get the final frequent item (L) =L1 U L2= {A, B, C, D, AB, BC, BD, CD}.

1.8 Problem Definition

Given a social network graph G (from a social network such as Twitter), an existing product p (iPhone, Samsung), the goal is to find a community of influential nodes (seed users) who have a positive opinion about the product (based on mining their tweets), for opinion maximization process based on mining of users' opinions (positive) on product relevant tweets and relationships from a friendship network graph $G(V, E)$ where every edge $e_{ij} \in E$ connects nodes v_i and v_j ($v_i, v_j \in V$ and $i, j = 1, 2, \dots, N$) and indicates v_i and v_j have relationships on a selected product. The result of the process is (1) select small set of seed users who have positive opinion by finding all users with positive opinion about the product and only these users will be used to form the network graph, (2) applying the Multi Linear Threshold model for opinion maximization to select the small seed with the maximum influence from the graph from (1).

1.9 Thesis Contribution

The main limitation of existing related systems such as CONE is that they do not consider users' opinions from social posts where users interact and express the most. CONE considers users' historical ratings and then performs influence maximization, whereas, in historical ratings, they have only considered ratings and not comments posted by users on posts. OBIN identifies the relationships among nodes and creates an influence network but does not perform Influence or opinion maximization, which is more profitable to the seller when introducing a new product in the market for opinion estimation when opinion is not present about a product.

Thus, in this thesis, we propose a system to first create the historical ratings of a large group of users on similar products through opinion mining by extracting users' opinions and creating a

community opinion network using selected users who have expressed positive opinions. In the next round, we perform opinion maximization to maximize the overall opinion spread of top-k users. The process of opinion mining and opinion maximization intertwine with each other, which requires the model to consider the two components collectively.

We see the necessity of combining these two approaches, opinion mining (sentiment analysis) and opinion maximization, together to discover communities based on their opinions in social networks. We create a community opinion network of closely related groups of users with the same sentiments or opinions towards a product. Identifying influential users in social networks can help companies with market segmentation and design strategies to better understand people's opinions about a new product without wasting many resources again to create a new opinion network. The proposed ACOMax system will use users' social posts acquired via Twitter for opinion mining, which tell us about user opinions of the item and to discover influential users and their communities that can contribute to do a profitable business. Motivated by the above real-life scenario and viral marketing, we propose a system ACOMax for mining opinions from Twitter using Twitter API and discovering communities for opinion maximization. Our system considers both implicit and explicit opinions, posts on selected product.

1.9.1 Thesis Feature Contribution

In the proposed ACOMax, we perform opinion mining of a selected product on multiple posts to perform opinion mining (feature-level and sentence-level) for identifying nodes with positive opinions about the selected product and then perform opinion maximization to select final seed set of users. In feature-level opinion mining, only feature-containing reviews are selected (as in the OBIN system), neglecting numerous opinions who do not have features mentioned but offer positive feedbacks. Selecting all reviews will add more users and expose their network and contribute to the overall profits or revenue generations. To solve the above problem, we make the following feature contributions in this thesis:

- 1. Reducing opinion spamming by extracting **Time-based** posts and users (select posts and users which are part of Twitter for at least a year and location is Canada) for a targeted product p .**

Reason: Fraction of users and their likes are flagged and removed by automated systems by maximum of 10 months (Viswanath, et al., 2014).

2. Joint Opinion Mining (JOMiner) approach computes the overall opinion score about the product by performing a combined sentence and feature opinion mining. Combination of subjectivity classification, lexical and feature mining produce better influential users than any of the feature sets individually (Belisário, Ferreira & Pardo, 2020). More the subjective reviews (inclusion of slang, emoticons) are used for feature mining more they will be associated with positive sentiments and higher the probability that the overall score is positive.

3. **Community Opinion Network** of users with overall positive opinion score toward the product and assigning influence probability weight among nodes using real world opinions such as likes, retweets and comments over the post.

4. Disregarding users with negative opinion before diffusion process in OM, solves the problem of cold start. We leverage community opinion network graph G to maximize the total opinion spread about the product selected by seller.

1.9.2 Thesis Procedural Contributions

This paper's main problem is extracting a community of influential nodes from a social network graph who offer positive opinion about the product using opinion mining and then perform opinion maximization. This thesis proposes a system (ACOMax), which consists of the following steps:

To make the specified feature contributions, this thesis proposes Active Community Opinion Network Mining and Maximization (ACOMax) system which consists of following major steps:

1. In the OBIN system, the author proposed Topic-Post Distribution (TPD) to extract multiple posts on multiple products. In the proposed ACOMax system, we focus on time-based opinions to consider changing preferences and targeted marketing (multiple posts on a single product). We have proposed Social Post Miner (SPOM), which extracts time-based relevant posts, nodes and considers top nodes based on the number of likes and shares.
2. Instead of only feature-level for opinion mining as used in OBIN we propose joint approach JOM to consider sentence-level and feature-level opinion mining collectively.
3. Construct influence graph $G = (V, E)$ from Twitter follower network of user stored at the end of step 2 and assign influence probability weight to each edge between nodes by computing the below formula:

$$P_{u,v} = \frac{(\#retweets\ of\ u\ on\ v) + (\#replies\ of\ u\ on\ v) + (\#quotes\ of\ u\ on\ v)}{\#tweet\ of\ v}$$

Finally, select the initial set of the users before the opinion maximization process for influence spreading on neighbours.

4. Joint Opinion Mining (JOM) solves the problem of cold start by providing initial set of influential users before opinion estimation and maximization of a new target product such as iPhone12 by the promoter using Multiple Linear Threshold (MLT) an extended version of LT by (Kempe et al., 2003).

1.10 Outline of Thesis

CHAPTER 2: Discuss related Community detection and opinion mining systems, different IM and OM algorithms.

CHAPTER 3: Discusses the proposed Active Community Opinion Mining and Maximization (ACOMax) from social network.

CHAPTER 4: Discusses the experimental implementation for proposed ACOMax system, required tools and technologies.

0 Discusses about the future work and conclusion.

CHAPTER 2: RELATED WORK

2.1 Feature-based opinion summarization mining or FBS (Hu & Lu, 2004) The proposed feature-based summarization FBS method that mine product features from customers’ reviews, identifies sentiment opinion, and summarize the results. The inputs to FBS are a product name and an entry web page for all the reviews of the product. FBS method has the following task:

1) Part-of-Speech Tagging (POS) – NLProcessor linguistic parser is used to parse each review to split text into sentences and to produce the POS tag for each word. Output of the NLProcessor is XML (<http://www.infogistics.com/textanalysis.html>). Example <W C='NN'> means a noun and <NG> means a noun group or noun phrase. Each tagged sentence is saved in the review database. **Example** – suppose a sentence “I am absolutely in awe of this camera”. Output of POS steps is: <S><NG><W C = 'PRP' L = 'SS' T = 'w' S = 'Y'>**I**</W></NG> <VG><W C = 'VBP'>**am**</W><W C = 'RB'>**absolutely**</W></VG> <W C = 'IN'>**in**</W><NG><W C = 'NN'>**awe**</W></NG> <W C = 'IN'>**of**</W><NG><W C = 'DT'>**this**</W><W C = 'NN'>**camera**</W></NG><W C = '.'>.</W></S>

2) Frequent Feature Identification: Association miner CBA (Liu et al., 1998), which is based on the Apriori algorithm (Agarwal & Srikant, 1994) with minimum support of 1%, is applied to obtain the frequently occurring nouns or noun phrases that are explicitly mentioned in the reviews. The generated frequent itemsets are also called candidate aspects.

Example: Assuming there are three sentences in the review, the frequently occurring nouns are shown below:

Sentence #	Noun/Noun Phrase
1	camera, the focus, manual, a broad strap
2	the memory card, lens,
3	bright pictures, camera, zoom

3) Opinion Words Extraction– If a sentence has one or more than one product features and one or more opinion words, then it is called opinion sentence. The opinion words are identified by the following method: **Example** – “The auto-flash is disgusting and makes the face blur”, here disgusting is the effective opinion of auto-flash. “The picture quality is awesome” and “The application that is used in it is awesome” share the same opinion word awesome and suppose there are no sentences to talk about picture quality or application. That means these two

features are infrequent. In this case, the nearest noun phrases around the opinion word awesome are picture quality and application.

4) Opinion Words Orientation—Words that encode a desirable state (e.g., beautiful, amazing) have a positive orientation, while undesirable state (e.g., ridiculous) have negative orientation.

This task has following steps:

- a. Select adjective list from WordNet store them to seed list. For example, great, cool, nice, fantastic are positive adjectives; and bad, dull, dumb are negative adjectives.
- b. In WordNet, adjectives are organized into bipolar cluster. Example the Figure 14 shows bipolar adjective structure for the word ‘tiny’

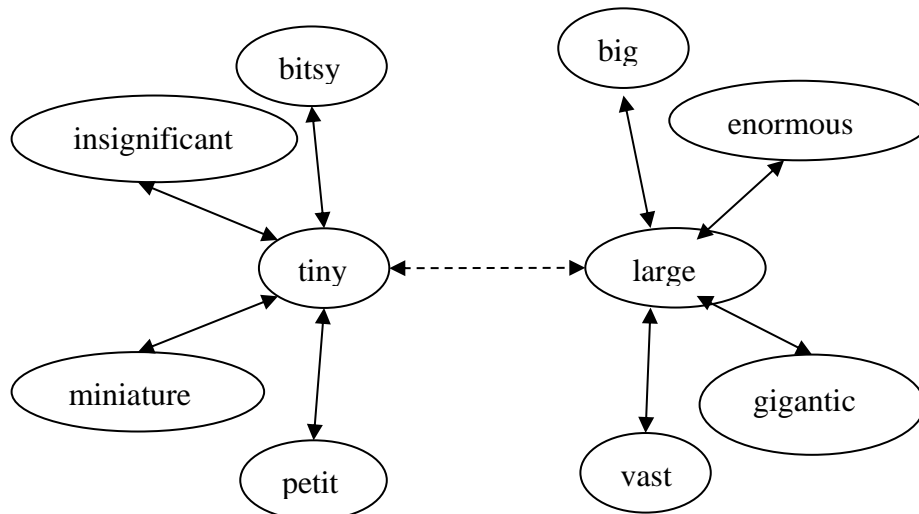


Figure 14 Bipolar adjective structure, synonym, and antonym

Example, Identification method of positive or negative sentences has following steps for the feature “picture” let us take example sentences

- “Overall, this is a good camera with a really good picture clearly”. This sentence is determined as s_i positive by fulfilling first if statement.
- “The auto and manual along with movie modes are very easy to use, but the software is not intuitive”. The orientation of this sentence is determined by the last else statement and average orientation of effective features are used, and the average orientation is positive.

2.2 Greedy algorithm for Influence Maximization (Kempe, Kleinberg & Tardos, 2003)

The goal is to solve the viral marketing problem by choosing a good initial set of nodes (customers) to target, as optimization problem in the context of these models. First, they introduced diffusion models namely LT and IC then they defined influence spread function, $\sigma(\cdot)$ as follows, given a network graph $G(V,E)$ which is directed with influence probability or weight for each edge, and a diffusion model M , the influence of set of vertices $A \subseteq V$, denoted $\sigma^M(A)$ is the expected number of active vertices once the diffusion process is over. Using these notations, we can formally define the k-best maximization problem as follows:

Problem 1 (Influence Maximization) Given a social network graph $G(V,E)$ along with influence probabilities of all edge in E , a diffusion model M and a number k find a set $A \subseteq V$, $|A| \leq k$ such that influence spread, that is $\sigma^M(A)$, is maximum. Kempe, Kleinberg & Tardos, prove that the optimization problem is NP-hard for both LT and IC Models and influence maximization problem cannot be solved in polynomial time. However, that $\sigma(M(\cdot))$ function is sub-modular and monotone.

Theorem 1 (Kempe, Kleinberg & Tardos, 2003): For an arbitrary instance of the Independent Cascade Model or Linear Threshold Model, the resulting influence function is submodular and monotone.

According to (Nemhauser et al. 1978) any submodular monotone function can be solved using natural greedy algorithm with a $(1-1/e)$ approximation guarantee (Theorem 1).

Theorem 2 (Nemhauser, Wolsey & Fisher, 1978): The greedy algorithm gives a $(1 - 1/e)$ approximation for the problem $\max \{f(S) : |S| < k\}$ where f is a monotone submodular function.

That is due the submodular and monotone property of $\sigma^M(\cdot)$ function, the greedy solution will produce result which is at least 63% of the optimal. They presented the greedy algorithm (figure 11) which takes social network graph $G(V, E)$, k and Model m . It begins by initializing seed set S to Null (line 1). Vertex w which maximizes the marginal gain $\sigma^M(S \cup \{w\}) - \sigma^M(S)$ is added to S at each iteration (line 3), until the $|S|=k$.

From Node	To Node	Influence Probability
A	E	0.1
A	F	0.9
B	A	0.3
B	D	0.4
B	C	0.3
C	D	0.6
C	E	0.4
D	E	1
E	A	0.55
E	F	0.45

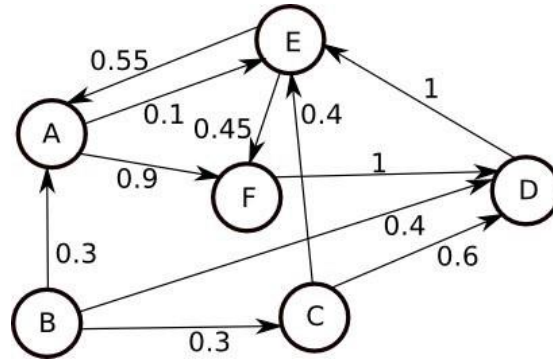


Table 4: Social network data with Influence Probability

Figure 15- Social Network Graph with influence probability modeled on data in table 4.

Algorithm 1: The Greedy k-best influence maximization algorithm

Input: G, k, σ_m /* G is the social network graph, k the desired size of the seed set and σ_m is the influence model*/

Output: Seed set S

Begin

1. Set $S \leftarrow \emptyset$
2. **for** $i = 1$ to k **do** /*Look for seeds until k seeds are found /
3. $u \leftarrow \operatorname{argmax}_{w \in V-S} (\sigma_m(S \cup \{w\}) - \sigma_m(S))$; /*Pick node u which have maximum marginal gain/
4. $S \leftarrow S + u$
5. **end for**

Figure 16- The Greedy k-best influence maximization algorithm

In step 3 of the greedy algorithm, the conventional method for estimating all the marginal influence gain of any node w in V , $\sigma_m(S \cup \{w\}) - \sigma_m(S)$, with respect to current seed set S is described as follows: First, a sufficiently large positive integer M is specified. For any node $w \in V - S$, the process of the diffusion model (IC or LT model) is run for the initial active set S and also for $S \cup \{w\}$, and the number of final active nodes activated by S (or $S \cup \{w\}$), denoted as $\phi(S)$ (or $\phi(S \cup \{w\})$), is counted. Each $\sigma(S)$ and $\sigma(S \cup \{w\})$ is then estimated as the empirical mean obtained from M such simulations.

Example: Let us consider the social network graph with influence probability given in figure 15 above. Using this will demonstrate the greedy algorithm under independent cascade model. For simplicity we will demonstrate the algorithm by setting M to 1. Meaning we are going to estimate

the influence spread by running the diffusion random process only once. Also let us consider $k=2$ i.e., we are looking for a set of influential nodes, S of size 2 from the social graph in figure 15.

Step 1: First the algorithm will first initialize $S = \emptyset$ (Line 1). Then the algorithm will enter a for loop. Since, $k=2$ this loop will run twice. In the first iteration the algorithm will look for node $w \in V \setminus S$ which maximize the marginal gain of influence spread relative to the set S (is null at this point).

Step 2: To get this algorithm will compute the number of nodes activated by set $S \cup \{w\}$ for each $w \in V \setminus S$ under the independent cascade model. Following is the list of all nodes $w \in V \setminus S$ and number of nodes that gets activated by each of these:

- {A– 4 as it activates nodes F, D and E},
- {B - 3 as it activates nodes D and E},
- {C– 3 as it activates nodes D and E},
- {D– 2 as it activates node E},
- {E– 1 as it does not activate any more nodes},
- {F– 3 as it activates nodes D and E}.

Based on the above, the algorithm will choose node A in the first iteration and put it into set S . Now the seed set $S = \{A\}$ and we still need to find one more node so that $|S| = 2$.

Step 3: In the second iteration the algorithm will compute marginal gain, $\sigma M(S \cup \{w\}) - \sigma M(S)$ for each $w \in V \setminus S$, of each of the remaining nodes in $V \setminus \{A\}$. Following is the list of nodes and its corresponding number of nodes activated by adding it to set S :

- B – 2 as it activates node C
- C – 1 as it does not activate any additional nodes.
- D – 0 as it does not activate any additional nodes and D is already activated by set S
- E - 0 as it does not activate any additional nodes and E is already activated by set S
- F - 0 as it does not activate any additional nodes and F is already activated by set S

From the above numbers we can see that **B has the highest marginal gain, i.e., activates most nodes**. So, node B is now added to set S which now has 2 nodes $\{A, B\}$.

Output: Since $|S|=2$ which was our required number of influential nodes the algorithm stops and here and **return $S = \{A, B\}$** .

They compared their algorithm in three different models of influence – independent cascade model, the weight cascade model, and the linear threshold model. Also, their greedy algorithm

with heuristics based on node's degrees and centrality within the network, as well as choosing random nodes to target. In paper by (Kempe, Kleinberg & Tardos, 2003) shown through experiments that their greedy algorithm significantly outperforms, in terms of influence spread, the degree and centrality-based heuristics in influence spread. One of the main limitations of the above greedy approach is efficiency and scalability. Note that for selecting a node (step 3) that maximize the marginal gain $\sigma(S \cup \{w\}) - \sigma(S)$ is computationally expensive, as it needs to explore all the possible combinations.

2.3. ‘Lazy Forward’ or CELF Optimization (Leskovec et al., 2007) - To improve the scalability of greedy approach of influence maximization is the problem of selection of nodes in a network to detect the spreading of information as quickly as possible. CELF exploited the submodular property of influence function $\sigma_m(\cdot)$ to develop an efficient algorithm called CELF, based on a “lazy-forward” optimization in selecting seeds. Due to the submodular property the marginal gain of a node in the current iteration of the greedy algorithm cannot be better than its marginal gain in the previous iterations. To take advantage of this property CELF algorithm maintains a table of marginal gain, $mg(u, S)$, of each node u in current iteration sorted on $mg(u, S)$ in decreasing order, where S is the current seed set and $mg(u, S)$ is the marginal gain of u with respect to S . Table $mg(u, S)$ is re-evaluated only for the top node in next iteration. If required, the table is resorted. If a node remains at the top after this, it is picked and added to the seed set. They evaluated their methodology extensively on two large-scale real-world scenarios namely: a) detection of contamination in large water distribution network, and b) selection of informative blogs in a network of more than 10 million posts. Using these scenarios, they compared CELF's performance and scalability with natural greedy algorithm as shown in figure 17.

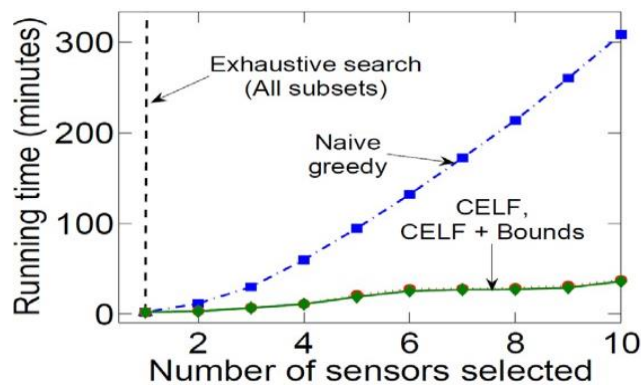


Figure 17: Running time of exhaustive search, greedy and CELF (Leskovec et al., 2007)

Example: Consider the social network graph in figure 15 with given influence probabilities. Again, set $k=2$, we are looking for the seed set of size 2 like the greedy approach CELF optimization will pick node A in the first iteration and will also create a table $mg(u, S)$ as follows: The algorithm will pick A as its marginal gain is the highest and will be removed from the table as follows:

Mg (A, {})	4
Mg (B, {})	3
Mg (C, {})	3
Mg (D, {})	2
Mg (E, {})	1

Mg (B, {})	3
Mg (C, {})	3
Mg (D, {})	2
Mg (E, {})	1

Now in the next iteration the CELF optimization the algorithm will only evaluate the top node, i.e., node B. We saw earlier that marginal gain of node B in respect to $S=\{A\}$ was 3. As there is no change the node B will select as next seed and added to the seed set S. Note that unlike greedy algorithm discussed in the previous section the CELF algorithm avoids computing marginal gain of rest of the nodes (such as C, D and E) and still gets the same result. Thus, CELF is much efficient compared to greedy algorithm. In terms of performance experimental results, CELF generated results that are at most 5% - 15% from optimal. In terms of scalability, CELF also performed a lot better than greedy. For example, for selecting 100 influential blogs, the greedy algorithm requires 4.5h, while CELF takes 23 second (about 700 times faster). Also, memory usage of CELF is about 50 MB compared to 3.5 GB required for greedy algorithm.

2.4. Discovering Influential Nodes from Social Trust Network by Ahmed & Ezeife, 2013 (Trust-General Threshold, T-GT) - This paper state that existing influence diffusion models such as the Linear Threshold model and the Independent Cascade model (Kempe, Kleinberg & Tardos, 2003) consider only positive influence propagation in a social network. However, two opposite relationships (such as like vs. dislike, love vs. hate, trust vs. distrust, friend vs. foe, and so on) may coexist in a social network. For example, users on Wikipedia can vote in favor or against the nomination of others, users on Epinions an express trust or distrust of other people's product reviews by rating, and participants on Slashdot an declare others to be either "friends" or "foes", users on YouTube can express like or dislike of other people's comments. The

authors claim that we need to consider both positive influences exerted by people we trust or like and negative influence exerted by people we do not trust or dislike while studying influence diffusion process. Existing diffusion models for Influence Maximization are modeled such that a node's probability of performing an action (or adopting a product) will increase as the number of his/her friends performing the same action increases. However, the authors argue that, a node's probability of performing an action (e.g., buy a new iPhone) should also decrease if its distrusted users, also buy an iPhone.

Example: In TGT model, a node u trusts node v but distrusts node w . In the corresponding influence graph, if node u trusts node v , then node v positively influences node u with the probability of $p^{+v,u}$ with $p^{-v,u} = 0$. If node u distrusts node w , then node w negatively influences node u with the probability of $p^{+w,u}$ with $p^{-w,u} = 0$. The authors define the positive influence probability $p^{+v,u} = A_{v,u} / A_v$, where A_v denotes the number of actions performed by node v and $A_{v,u}$ denotes the number of actions propagated from node v to node u (i.e., the number of v 's actions imitated by node u). For example, the action log shows that node v (trusted by node u) performs 3 actions in total. Among v 's 3 actions, 2 actions are imitated by u .

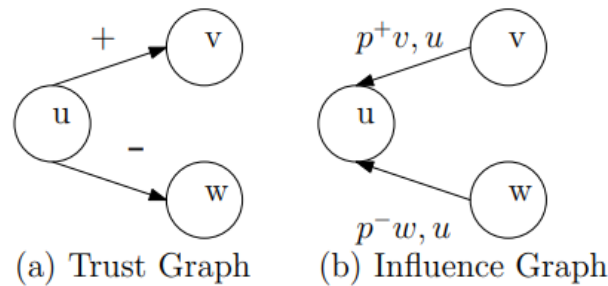


Figure 18: Trust Graph and Influence Graph

Hence, the probability of node u performing a task after node v performs the same action is $2/3 = 0.66$, which is the positive influence probability of node v on node u . Then the authors define the negative influence probability $p^{-v,u} = A'_{v,u} / A_v$ where A_v denotes the number of actions performed by node v and $A'_{v,u}$ denotes the number of actions not propagated from node v to node u (i.e., the number of v 's actions not imitated by node u). For example, the actions log shows node w (distrusted by node u , in Figure 18 (a)) performs 4 actions in total. Among w 's 4 actions, only 1 action is imitated by node u , the remaining 3 actions are not imitated by node u . That is u does not perform 3 out of 4 tasks performed by w . Hence, the probability of node u not performing a task after node w performs the same actions is $3/4 = 0.75$, which is the negative influence probability

of node “w” on node u. The authors propose an effective algorithm named MineSeedLS to discover influential nodes from trust network. T-IM takes a social network graph $G(V, E)$ and a budget k meaning to find at most k influential nodes. The algorithm returns a set of influential nodes of size at most k , also known as seed set, $S \subseteq V$. The algorithm starts by initializing seed set S to \emptyset . Then the algorithm computes influence spread of each node $v \in V$. The node with highest influence spread is picked and added to S . MineSeedLS then performs the following local search operations:

- (1) Delete, if by removing any node v in S increases the influence spread under the T-IM model, then the node v is removed from S .
- (2) Add, if by adding any node v in $V - S$ increases the influence spread under the T-IM model, then the node v is added to the set S .
- (3) Swap, if by swapping any node v in S with any node u in $V - S$ increases the spread under T-IM model the node v is removed from S and node u is added to S .

2.5 Social Network Opinion and Posts Mining for Community Preference Discovery

by Mumu & Ezeife, 2013 (OBIN) - This paper proposes a new influence network (IN) generation algorithm (Opinion Based IN:OBIN) through opinion mining of friendship networks (like Facebook.com). OBIN mines opinions using extended OpinionMiner that considers multiple posts and relationships (influences) between users. Approach used includes frequent pattern mining algorithm for determining community (positive or negative) preferences for a given product as input to standard influence maximization algorithms like CELF for target marketing.

Step 1: At first OBIN calls Topic-Post Distribution algorithm (TPD, it keeps track of $V \times D$ (user by profile) matrix, $D \times W$ (profile by posts) matrix, and $W \times C_m$ (post by comments) matrix to extract relevant nodes $v \in V$ for a topic z and filter them according to higher influential score determined by Approve A and Simple Response SR . The resultant data are stored into our transactional database for next steps. Let us take a topic $z = \text{“iPhone 5”}$. In our pre-processing model, $user_name = \text{“iPhone”}$ will result the following data: Message: “Perfect Fit Tech wants to know which is better? iPad Mini vs iPad gen?”, picture: “<http://fbcdn-photos-a.akamaihd.net/3842.jpg>”, shares: 91, likes: 6171, comments: 47.

In this step, we investigate four parameters: “message”, “shares”, “likes”, and “comments”. According to problem definition,

$$Approves(A) = likes = 6171$$

Simple Response (SR) = shares + comments = 91 + 47 = 138

Then, apply a term matching process to find whether “*message*” contains the topic-term or not.

Resultant data have the following tabular format:

Posts	Term	<i>A</i>	<i>SR</i>
Post1	Yes
Post2	Yes
:
Postn	No

A profile d is a vector w_d of N_d posts; a vector v_s of V_s nodes chosen from a set of nodes of size V . A collection of D profiles on topicz is defined as: $D = \{D_1, D_2, \dots, D_i\}$; $D_i = \{(w_1, v_i), (w_2, v_i), \dots, (w_N, v_i)\}$ where $w = \text{topic} - \text{post}$ and $N = \text{number of topic} - \text{posts}$.

Step 2: In this second step, OBIN calls PCP-Miner to fetch all the opinions for each relevant post w of each relevant node v , and apply sentence and word segmentation and some cleaning such as stemming, string matching etc. PCP-Miner apply POS-tagging (Brill, 1994) to identify adjective, adverb as opinion words and noun, noun phrase as features. Then identify the polarity of the comment i.e., the comment expressing positive or negative opinion. And finally compute the popularity of the relevant post w . In this algorithm it identifies opinion comments across all the comments on that post w , identifies the semantic orientation (SO) of the comments, and measure the polarity of the comments as well as the popularity of the post. Our proposed PCP-Miner model considers four major features on users’ comments: White Responses (R_W), Black Responses (R_B), Raising Discussion (RD), and Controversiality (C). The positive, negative, or neutral polarity is determined as follows:

$$R_W \text{ if } (Positive \text{ comments} > (Negative \text{ comments} + Neutral \text{ comments}))$$

$$R_B \text{ if } (Negative \text{ comments} > (Neutral \text{ comments} + Positive \text{ comments}))$$

$$RD = (nC_L - nC_T) \times nC_U, C = Y/X$$

Where nC_L = number of comments that are replies to other comments, nC_T = total number of comments, nC_U = total number of unique comments, Y = total number of negative comments and X = total number of positive comments. They consider $0.5 < C < 1.5$. If $C = 0$, then total

agreement i.e., the post is either positive or negative. If $C = 1$, then highest controversiality, i.e., the post opinions split exactly into two sides.

Step 3: In this step, solution framework stores all the extracted and computed data into data warehouse for further mining purpose.

Step 4: In this step, paper created ranked list of mined relevant nodes $v \in V$, their corresponding popular topic-posts $w \in W$, and aggregated opinions on each post along with their polarity (positive impact or negative impact). Proposed solution framework OBIN calls PoPGen model to identify the relationships among nodes v_i and v_j ($v_i, v_j \in V$ and $i, j = 1, 2, \dots, N$ on a topic z and how they influence to each other. This solution also identifies a global relation between nodes v_i , and v_j for similar topic, hence discover the community preference.

Example: To demonstrate the OBIN framework, they use a small sample real-time dataset extracted from Facebook and integrate Graph API with FQL and conduct a local search in Facebook to collect all the relevant nodes $v \in V$ for a given topic z . To collect a complete list of topic categories, they use Facebook and run jQuery and collected 146 categories. Table 5 shows a sample list of categories collected from Facebook, where Cat_id represents the category id and Cat_name represents the title of the category.

Step 1: Let us suppose, $z = \text{iphone}$, input to Graph API: {"https://www.facebook.com/search/results.php?"}, FQL = {SELECT id, name, category, likes, links FROM search WHERE q = 'iPhone' AND (type = 'page' OR type = 'group')}. The results executed from Graph API and FQL are shown in Table 2. It denotes the schema of relation as $U_R = \langle \text{Node}(v), \text{Term}, A, \text{Link} \rangle$

Cat_id	Cat_name
1103	Actor/Director
1105	Movie
1109	Writer
1202	Musician/Band
1300	Book
1602	Public Figure
1700	Politician
2214	Health/Beauty

Table 5 Example of topic categories

Node id v	Term	A	Link
130489060322069	iPhone	3116728	iPhone. Page
110018862354999	iPhone 4	1435239	Iphone-4
214456561919831	iPhone Fans	261210	theappleclan
101936296565340	iPhone 4S	262165	IPhone-4S/101936296565340

144971705536847	iPhone 3G	234676	IPhone-3G/144971705536847
-----------------	-----------	--------	---------------------------

Table 6 Example of relevant nodes and data for $z = \text{iPhone}$

For example, in table 6, the first row shows a node with unique id “130489060322069” and name “iPhone” (in Term column) that has 3116728 friends, and we can visit his profile by “iPhone. Page” link. They are analyzing data set with language in English. So, although $v=159984244020234$ has a good A value, we ignore it, and we index the data set according to A in descending order.

Step 2: Generate Topic-Post Matrix for each relevant node. From the step 1, we have a set of 130 nodes with their corresponding Approve (A) and links to their profile. Now let us set a threshold Approve (A) as 1000, meaning that we are looking for nodes having $A \geq 1000$ from this dataset. Now preprocessing step takes each node from table 2 and crawl its profile page to search relevant posts on topic z . In table 2 we have a set of 7 users V :

$V = \{130489060322069, 110018862354999, \underline{101936296565340}, 214456561919831, \underline{144971705536847}, \underline{267282993312609}, \underline{146534208714348}\}$

$A = \{3116728, 1435239, 262165, 261210, 234676, 189483, 118674\}$.

Let’s take node $v = 130489060322069$ and execute query as: FQL = {SELECT post_id, message, likes.count, comments.count, share.count, created_time, (*comments.count* + *share_count*) FROM stream WHERE source_id=’ 130489060322069’ AND *message* != "" AND *created_time* = *month*(‘2013 – 03 – 06’) ORDER BY *likes.count* desc LIMIT 100}.

This results a set of first 100 posts with total number of likes, comments, and shares. For each post denote the schema of the relation as $P_R = \langle Post(w), Term, A, SR \rangle$ where $SR = \text{comments} + \text{shares}$. For example, first row in Table 7 shows a post with id “469219579782347” posted by node “130489060322069”, that has the post title “black- like, white-comment, and the winner is ?” and has got 61153 likes in the post, and total number of re-shares and unique comments are 11325.

Post idw	Term	A	SR
469219579782347	black- like, white-comment, and the winner is ?	61153	11325
468646856506286	pretty amazing	33899	2213

469758623061776	Apple 5th Avenue	33041	2198
467263769977928	white or black?	31359	10364
465792903458348	Take it	28028	2622
180356388777720	Amazing iPhone!	20147	1880
465731800131125	iPhone 5 - The biggest thing to happen to iPhone since iPhone :)	19685	1420

Table 7 Example of Post Data

Step 3: In this step, we apply $(term + A)$, $(term + SR)$, and $(A + SR)$ features to classify relevant and irrelevant nodes using Support Vector Machine (SVM).

U_id	Name	Link
429326	Alex Brown	http://www.facebook.com/Alex.Brown
223952	Peter Pen	http://www.facebook.com/223952

Table 8 Facebook User Table

P_id	Approves	SR	RD	C	Score (θ_z)	Title	Link
962538	1990	78	NULL	NULL	0	Samsung VS Apple	http://www.facebook.com/223952/posts/962538

Table 9 Posts Table

Cat_id	Tp_id	P_id	Cm_u_id	Polarity	Time_posted	Comment
1	2	962538	6932106	NULL	2012-11-02 19:04:08	I have an iPhone 5, i upgraded from a 4. The overall application of the phone is awesome.
1	2	962538	40527930	NULL	2012-11-02 19:10:02	iPhone 4 was much better than this.

Table 10 Comments Table

TPD keeps track of $V \times D$ (user by profile) matrix, $D \times W$ (profile by posts) matrix, and $W \times C_m$ (post by comments) matrix. Let us suppose $term = \{iphone, Apple, cell, mobile, handset\}$, $A \geq 100$, and $SR \geq 20$, which extracts most relevant posts on the topic $z = iphone$. Then store the relevant nodes data, posts data, and corresponding users' comments data in our transactional database called OBIN_transaction. We denote the schema of relation as $D = \langle$

$D_1, D_2, \dots, D_i >, D_i = < (w_1, v_i), (w_2, v_2), \dots, (w_N, v_i) >$. For example, $D = \{\text{iphone, iphone 4, iphone Fans}\}$,

$D_1 = \{(469219579782347, 130489060322069), (468646856506286, 130489060322069), (469758623061776, 130489060322069), (467263769977928, 130489060322069), (465792903458348, 130489060322069), (472223806148591, 130489060322069), (466379303399708, 130489060322069), (180356388777720, 130489060322069)\}$

Step 4: From TPD, a list of all comments for each post on topic are extracted in this step of PCP-Miner. PCP-Miner takes all the comments and data preprocessing is done by sentence segmentation and cleaning. In this step, table 11 shows a sample comment data for $w = 180356388777720$ after applying cleansing method. Then for each comment (c), the PCP-Miner algorithm performs the above-mentioned process as following steps:

Post id w	User id v_t	Time	Comment c
180356388777720	100002395810151	2013-01-06T05:57:57+0000	i want
180356388777720	100003290108936	2013-01-06T10:18:16+0000	this is really cool
180356388777720	100004582655605	2013-01-06T11:35:48+0000	Cool
180356388777720	100002090841333	2013-01-07T12:19:56+0000	i want to have one lyk that

Table 11: Example of sample dataset for user comments

1). Apply Tokenization. Example, we take the row (i want to have one like that), the algorithm tokenizes to words according to punctuations $\{',', ';', ':', '!', '?'\}$ and spaces $\{ ' '\}$. $TK[5] = \{i, want, to, have, one, lyk, that\}$. All the tokenized comments are stored in a temporary hash table called TOK .

2). $NLProcessor(TK, Tag)$ is then take TOK table with a list of predefined $POS - tags$. $PO[5] = \{i_PP, want_VBP, to_TO, have_VB, one_NN, lyk_UH_IN, that_PP\}$. All the POS-tagged comments are stored in a temporary hash table called POS .

3). A list of adjectives, verbs, adverbs, and nouns are extracted from step 2 for all the user comments. For example, from c_5 , a set of features are $PO[1] = \{want_VBP, have_VB, that_PP\}$. Here the feature $\{that\}$ is infrequent feature. To identify the corresponding feature for infrequent feature $\{that\}$, we apply $AssociationRule()$ algorithm and found feature $FFT[5] = \{iphone\}$ i.e., the post $Term$ itself. All the frequent and infrequent features are stored in a temporary hash table called $FeqFT$.

4). OpinionExtractor() algorithm then extract opinion words from the POS table. Opinion words are the adjectives, verb, adverb across the extracted features. Extracted opinion words are stored in a temporary hash table called OE.

5). To compute the polarity measure of a comment, we need to identify the semantic orientation and polarity of the opinion words stored in the table OE . For each opinion word OW_i in the list OE , we search its synonyms or antonyms in WordNet and collect its orientation. For example, $OE[1] = \{\text{want, positive}\}$, $OE[2] = \{\text{cool, positive}\}$, $OE[5] = \{\text{want, positive}\}$. If a negative word comes in front of an opinion word, we consider the semantic orientation of the opinion word is its opposite orientation.

6). According to table OE, we have all the orientation i.e., the polarity of individual comment. To compute the popularity score θ_z of a post, we calculate the differences between all positive oriented comments and negative oriented comments. For example, Table 12 and Table 13 show the resultant popularity matrix for $w = 468646856506286$ where $\theta_z = (5 - 0) = 5$, and (Positive) $>$ (Neutral + Negative) i.e., $5 > (2 + 0)$.

Post id w	User id v_t	Polarity	Time	Comment c
180356388777720	100002395810151	positive	2013-01-06T05:57:57+0000	i want
180356388777720	100003290108936	positive	2013-01-06T10:18:16+0000	this is cool

Table 12 Example data in Facebook Comment table

Post id w	A	SR	θ_z	Term
180356388777720	20147	1880	51	Amazing iPhone!

Table 13 Example data in Facebook Post table

All the data of relevant nodes v_s , nodes who commented v_t , posts w , and comments c based on the popularity score θ_z , Approve A , and Simple Response SR , are then transferred to data warehouse OBIN_dwh.

2.6. Multi-Round Influence Maximization by Sun, Huang, Yu & Chen, 2018 (MRIM):

MRIM models the viral marketing scenarios in which advertisers conduct multiple rounds of viral marketing to promote one product where the advertiser can select seed sets adaptively based on the propagation results in the previous rounds. MRIM focus on multi round triggering model (MRT) which is a variant of basic trigger model as the basic diffusion model. In trigger model, a

social network is modeled as a directed graph $G = (V, E)$, where V is a finite set of vertices or nodes, and $E \subseteq V \times V$ is the set of directed edges connecting pairs of nodes. The diffusion of influence proceeds in discrete time steps $\tau = 0, 1, 2$ at time $\tau = 0$, the seed set S_0 is selected to be active, and each node v independently chooses a random triggering set $T(v)$ according to some distribution over subsets of its in-neighbors. At each time $\tau \geq 1$, an inactive node v becomes active if at least one node in $T(v)$ is active by time $\tau - 1$. The diffusion process ends when there are no more nodes activated in a time step. The MRT model includes T independent rounds of influence diffusions. In each round, the diffusion starts from a separate seed set S_t with up to k nodes. For the Multi-Round Influence Maximization (MRIM) problem, the goal is to select at most k seed nodes of each round, such that the influence spread in T rounds is maximized. At the beginning of each round, to determine the seed set for the current round based on the propagation results observed in previous rounds. If (S_t, t) an item, where S_t is the seed set chosen in round t . For each item (S_t, t) , after the propagation, the nodes and edges participated in the propagation are observed as the feedback. A realization is a function ϕ mapping every possible item (S_t, t) .

Given graph $G = (V, E)$, triggering set distribution, seed set budget k , number of rounds T , simulation number R , set of already activated nodes A_{t-1} as the feedback from the previous rounds return best seed set S_t and activated nodes A_t . For MRIM, in each round t , pick an item (S_t, t) , see its state $\Phi(S_t, t)$, pick the next item $(S_{t+1}, t+1)$, see its state, and so on. After each pick, previous observations can be represented as a partial realization ψ , a function from some subset of E to their states. A policy π is an adaptive strategy for picking items based on partial realizations. In each round, π will pick the next set of seeds $\pi(\psi)$ based on partial realization ψ so far. If partial realization ψ is not in the domain of π , the policy stops picking items. The goal of the MRIM is to find the best policy π such that it maximizes the influence spread according to feedback. MRIM operate at per round base, it uses greedy algorithm which takes feedback from previous rounds and selects the item for the current round, and then obtain new feedback.

Step 1: Initialize S_0 as seed selected. For all nodes v in the seed set of round t (v, t) that belongs to V_t and not yet added to S_0 then estimate the influence spread ρ [$\rho = (S_0 \cup \{(v, t)\})$] in the MRT model by simulating the diffusion process R times.

Step 2: Add maximum influence spread node (v, t) in each round k to seed set S_0 calculated in step 1. Repeat till the budget for round T exhausts or all seed users are activated. i.e. $(v, t) = \arg\max_{(v, t) \in V_t \setminus S_0} \rho^{\wedge}(S_0 \cup \{(v, t)\})$. Observe each propagation of S_t and update activated nodes A_t .

2.7. Active Opinion Maximization in Social Networks by Liu, Kong & Yu, 2018

(CONE): Given a social network G , an item set P , a rating matrix R , a target product pt that does not belong to P , the terminal round number T and the MLT (Multi Linear Threshold) diffusion model. The goal is to select k seeds from V (set of users) to activate at the beginning of round q to maximize the total opinion given by active users in G towards product pt at the end of round T . In MLT each user in a weighted social network G is associated with a threshold θ from $[0, 1]$. Before the first round of MLT, all users are inactive towards the target product. At round t , an inactive user becomes active. At each round, the influence only propagates n neighbors before the next round starts. All active users will stay active and cannot back to inactive status.

Input: Social network graph $G = (V, E)$, target product pt , partial historical rating matrix R , seed user size for each round: $k(1), k(2)$ to $k(T)$, matrix factorization parameter λ, κ (budget), number of rounds T .

Output: Set of best seed users selected in each round ($S^{(1)}$ to $S^{(T)}$).

1. First, from the given partial observed ratings, user profiles and item profiles matrix can be learned through fitting the observed ratings by solving the following optimization problem. Initially randomly initialize two matrices U and V as user profile and item profile matrices and estimate the updated user profile matrix ($U^{(0)}$) by using equation 5 with $q = 0$ and $\lambda > 0$:

$$\underset{U^{(q)}, V^{(q)}}{\text{minimize}} \|P_{\Omega}(R^{(q)} - (U^{(q)})^T V^{(q)})\|_F^2 + \lambda(\|U^{(q)}\|_F^2 + \|V^{(q)}\|_F^2)$$

Equation 5: Matrix factorization for learning user and item matrix

$P_{\Omega}(X)$ is the matrix with only the indices in Ω (set of observed ratings) of X preserved. $\|\cdot\|_F^2$ denotes the Forbenius 2-norm and λ is the regularization parameter for avoiding overfitting while estimating the user and item matrix and uses 10-fold cross validation to select the value of λ and k , in 10-fold cross validation it takes one-fold and compare with nine others and does it iteratively for all the folds.

2. Randomly initialize the target item profile $v^{(0)}$ and observed target item rating vector as $r^{(0)}$ and activated users set $C^{(0)}$. From the equation in step above, $U(q)$ and $V(q)$ is the updated user and item profile matrix at the end of round q . The last row of $V(q)$ is the updated profile vector for target item (pt). Now $U(q)$ and $V(q)$ is used to refine the predictions on users' opinion of pt using equation below. For loop of round from $q=1$ to T , update the estimated rating vector and initialize $S(q)$ which is set of seed users selected in each round q .

$$\hat{\mathbf{r}}(\mathbf{q}) = (\mathbf{U}^{(q-1)})^T \mathbf{v}_t^{(q-1)}$$

Equation 6: Inner product to estimate rating vector

Example: Given the budget k , apply greedy algorithm to select $k(q)$ users to maximize the total estimated opinion based on $\hat{\mathbf{r}}(\mathbf{q})$. At this step greedy algorithm is called to select best seed users. Among all users in the scan, the one with the largest estimated opinion is added to S and repeat the scan until run out of budget or there are no more inactive users in the network. To distinguish positive/negative opinions, convert the ratings in R to opinion as:

$$o_{ij} = \begin{cases} r_h - r_{avg}, & \text{if } \hat{R}_{ij} > r_h \\ r_l - r_{avg}, & \text{if } \hat{R}_{ij} < r_l \\ \hat{R}_{ij} - r_{avg}, & \text{otherwise} \end{cases}$$

where r_h and r_l are the highest rating and lowest rating, and r_{avg} is the average value of all ratings in R . In other words, consider $r = r_{avg}$ as the neutral opinion, $r > r_{avg}$ as positive opinion and $r < r_{avg}$ as negative opinion. Add new observed ratings obtained from first round q from greedy algorithm at step 5 to obtain $\mathbf{r}(q)$ and build the new extended rating matrix $\mathbf{R}(q) = (\mathbf{R}, \mathbf{r}(q))$ and keep adding till the end of rounds $\mathbf{r}(q-1)$. Again, estimate the user profile matrix $\mathbf{U}(q)$ and the item profile matrix $\mathbf{V}(q)$ by solving equation in step 3, last row of item matrix is updated item profile vector and mentioned as $\mathbf{v}(q) = \mathbf{V}(q)(:, n+1)$. Then, end of loop.

Example: Given a partially observed rating matrix on some products,

Item Users	M1	M2	M3	M4	M5
U1	3	-	1	-	1
U2	1	-	4	1	-
U3	3	1	-	3	1
U4	-	3	-	4	4

→

Item Users	M1	M2	M3	M4	M5
U1	3	1	1	3	1
U2	1	2	4	1	3
U3	3	1	1	3	1
U4	4	3	5	4	4

Table 14: a) Partial observed rating matrix b) Filled observed rating matrix

Step 1: First fill the partial rating matrix for user A who has not rated product M2 & M4. for product M2 = $((1*1) + (0*2)) = 1$, for product M4 = $((1*3) + (0*1)) = 3$. Similarly, for U2, U3 and U4. For user B, Prediction for product M2 = $((0*1) + (1*2)) = 2$
 Prediction for product M5 = $((0*1) + (1*3)) = 3$. Similar operations will be executed to fill the other ratings.

Step 2: Seed selection in two steps from figure 16 for the social graph between users

- 1). Replace undirected links and single directed links with two directed links. i.e., A to B and B to A.
- 2). Quantify the weight between nodes using Jaccard Coefficient, where set of users who follow user A and the set of users who are followed by A. Weight for C = $\frac{|A \cap \{A, c1, c2\}|}{|A \cup \{A, c1, c2\}|} \Rightarrow \frac{1}{3} \Rightarrow 0.33$ and $c1 = \frac{1}{3} = 0.33$ and $c2 = 0$. Respectively, $A = 1, B = 1, D = 0.33, d1 = 0, d2 = 0.33$

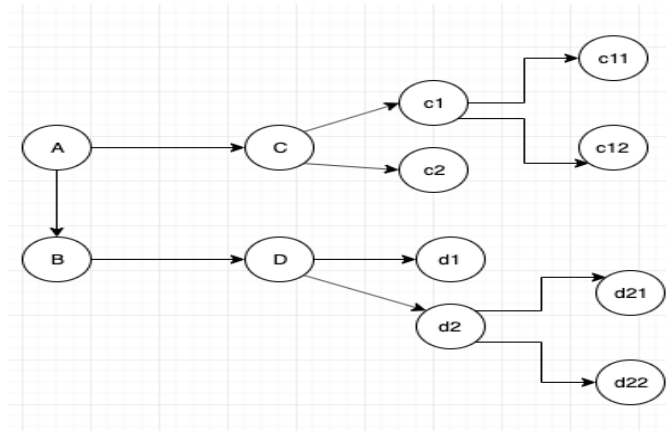


Figure 19 : Example of CONE for opinion maximization (Liu, Kong & Yu, 2018)

Step 3: If seed user size for each round i.e., $k=2$ and product selected is M4. For Round 1, $k(1) = \{A, B\}$, users selected that will maximize the estimated opinions.

Estimate avg rating on product M4 which is 2.75. A's rating is 3 and B's is 1. Selected at this round $S(1) = \{A\}$. For Round 2, $k(2) = \{C\}$, rating $>$ avg, at this round $S(2) = \{C\}$. For Round 3, $k(3) = \{c1\}$, if rating $>$ avg, $S(3) = \{c1\}$, else not selected.

Output: $\{A, C, c1\}$

2.9. Comparison of the existing models:

Name	Description	Limitations
FBS (Hu & Liu, 2004)	Proposed a system that mines and summarizes all the customer reviews of a product. It assumes that frequent nouns are the aspects of a product. Then, an orientation identification algorithm based on a pre-defined seed set and the semantic structure of WordNet are employed to automatically identify the opinion orientation.	This method tends to produce too many redundant features and miss low frequency but important features.

CELF (Leskovec et al., 2006)	Improve the scalability of greedy approach of influence maximization by (Kempe, et al., 2003) which in turn work 700 times faster than Greedy.	This method is a single cascade model and ineffective for real world opinion maximization.
MRIM (Sun, Huang, Yu & Chen, 2018)	MRIM focus on multi-round triggering model which is a variant of basic trigger model as the basic diffusion model where the advertiser can select seed sets adaptively based on the propagation results in the previous rounds.	Does not consider the opinions of the users.
T-GT (Ahmed & Ezeife, 2013)	Proposes a trust based social network where both positive relationships and negative relationships are considered based on trust relationship among users in trust network and proposes an algorithm MineSeedLS (as CELF-like algorithms cannot be applied to TGT) to discover influential nodes.	Considers only one measure of influence whereas Facebook has three measures of influence, i.e., comment, like, and share.
OBIN (Mumu & Ezeife, 2013)	OBIN takes as input a social network graph and a product z and outputs an influence graph for a product z from computed community preference of the entire social network containing only the relevant nodes to a certain product.	It is not viable option when any company is launching a new product because it also does not perform the opinion estimation of the users whose comments are not present about a product. Also, does not consider the active user selection because it compares itself with CELF, which is a single influence diffusion model and does not allow users to change their state once infected.
CONE (Liu, King & Yu, 2018)	Proposed multi round active seed selection called active opinion approach to select users who has largest number of positive opinions. It keeps selecting seeds to activate more users in multiple rounds until budget lasts and incorporates historical ratings of users on similar product and fill unknown ratings using collaborative filtering.	Doesn't consider social posts about a product and users' preferences. It also does not apply to a user-user social network, e.g., Facebook, where users' influence on an item depends on relationships between users, which requires the identification and inclusion of implicit and explicit opinions.

Table 15: Comparison of Existing Model

CHAPTER 3: The Proposed Active Community Opinion Network Mining and Maximization (ACOMax) System Through Social Network Posts

The Active Community Opinion Mining and Maximization (ACOMax) system's main objective is to select influential nodes relationships between nodes to select the best seed users before the start of the campaign for opinion maximization process. Opinion mining is performed to extract users' preferences from social posts from likes, comments, and retweets. Our goal is to identify popular tweets and perform opinion mining to analyze the users' sentiments about the selected product and select top influential users with positive opinion about the product from the friendship network. Usually, the target item is new to the market, and nobody has reviewed or rated it yet. To solve this, we perform opinion mining of users' opinions and estimate user opinions for a new target product using Collaborative Filtering via matrix factorization. In this, the campaigner can keep selecting seed users to activate more users in multiple rounds until using up all the seed budgets.

The proposed ACOMax system will use mined user ratings from multiple posts for a selected product, which tell us about user opinions about the item and will help us to estimate opinions for a new product that can contribute to do a profitable business. Motivated by the above real-life scenario and viral marketing, in this thesis, we propose ACOMax for mining positive opinions and discovering top influential seed users from the community of positive users for opinion maximization using Multiple Linear Threshold model (MLT). Our framework considers both implicit and explicit opinions and mines opinions from Twitter using TwitterAPI, and we use collaborative filtering to fill partial ratings of influential users, enhancing the user-item matrix with high accuracy.

3.1 Problem Definition

Given a social network graph G (from a social network such as Twitter), an existing product p (iPhone, Samsung), the goal is to find a community of influential nodes (seed users) who have a positive opinion about the product (based on mining their tweets), for opinion maximization process based on mining of users' opinions (positive) on product relevant tweets and relationships from a friendship network graph $G(V, E)$ where every edge $e_{ij} \in E$ connects nodes v_i and v_j (

$v_i, v_j \in V$ and $i, j = 1, 2, \dots, N$) and indicates v_i and v_j have relationships on a selected product. The result of the process is (1) select small set of seed users who have positive opinion by finding all users with positive opinion about the product and only these users will be used to form the network graph, (2) applying the Multi Linear Threshold model for opinion maximization to select the small seed with the maximum influence from the graph from (1).

3.2 Proposed Active Community Opinion Mining and Maximization (ACOMax) System

1). Figure of ACOMax System:

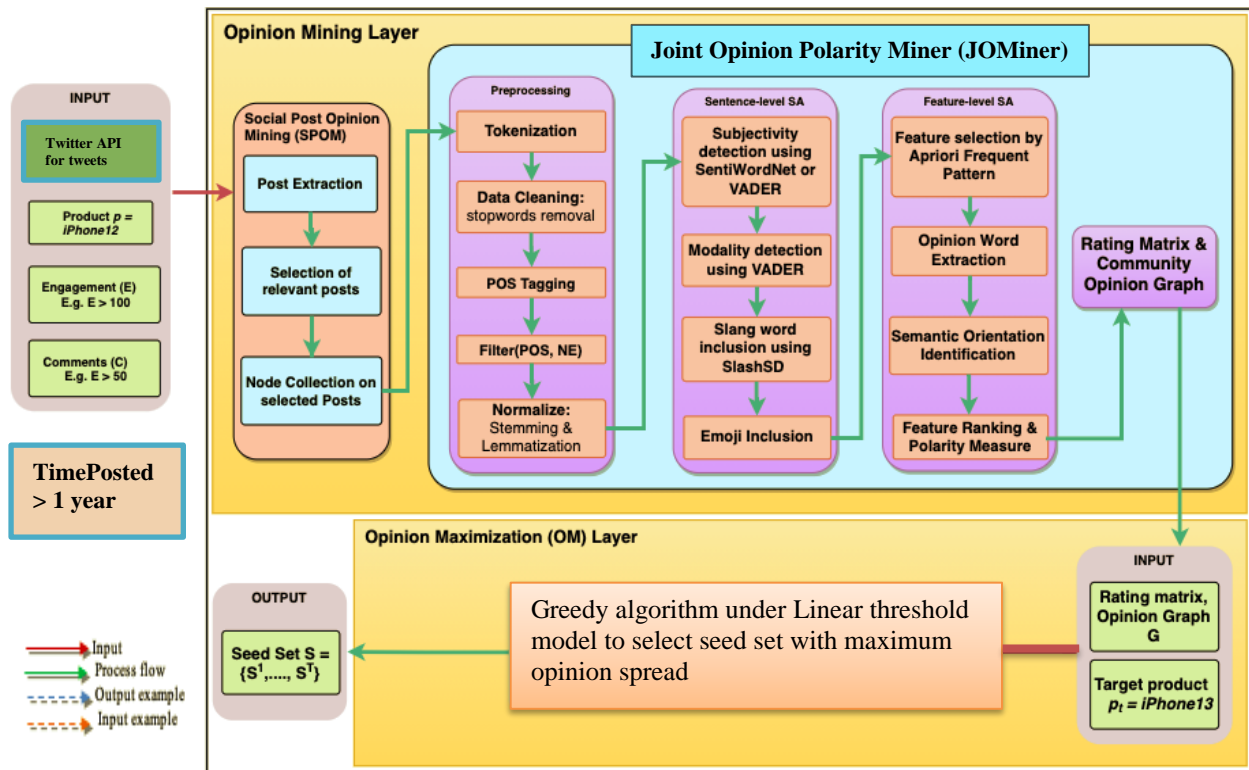


Figure 20: ACOMax Architecture

2). Algorithm of Active Community Opinion and Maximization (ACOMax) System

The major goal of the proposed Active (A), Community (C), Opinion mining (O), and Maximization (Max)-ACOMax is to mine community opinions (likes, comments, and retweets) from social network posts to enhance a user-item rating matrix for opinion maximization process for selecting small set of seed users to distribute sample product. Thus, ACOMax takes minimum threshold (likes, shares ad comments greater than a certain number), social network graph G to

extract product relevant posts and users considering friendship between nodes and nodes (users) with positive opinions as input to select final seed users as output (as shown in Algorithm 1) who will spread influence the most using opinion maximization.

Algorithm 1: ACOMax (Active Community Opinion Mining and Maximization)

Inputs: Tweets related to product and associated Comments, Likes and Retweets related to given target product p , seed user size for each round: $k^{(1)}, \dots, k^{(n)}$, campaign round number T .

Output: Seed user set S .

Procedure:

BEGIN:

1. In first stage, multiple tweets are selected related to the targeted product and selection of posts with enough comments and likes related to product p . ACOMax calls SPOM (detailed in 3.3) to extract posts and users' opinions from the. We only extract **time-based posts** (i.e., posts within 1 year).

1.1 Initialize Product p , Likes l , Retweets Rt and Comments C

1.2 **Profile matrix** $PM = [U, Pt, Rt, C, El]$ #Initialize Matrix for retrieved

1.3 **FOR** each tweet p in Tweets **DO**

1.3.1 Execute TwitterAPI to extract real time tweets related to product p .

1.3.2 For p in posts **DO**

1.3.3 If $PM(l1) > l$ and $PM(Rt1) > Rt$ and $PM(C1) > C$ #Check if count of likes, retweets & comments is greater than selected by seller

1.3.3.1 Add in $PM = [Pt, user u, product p, l1, Rt1, C1]$ #Add likes, retweets, comments, and users related to posts in the matrix

1.3.4 Else

1.3.4.1 do not select post

1.4 **Return** PM consisting of only relevant posts

2. After we have the relevant posts from stage 1, we select **time-based opinions** (i.e., comments within last six months) and **users who existed on Twitter for minimum a year**. Next, we perform Joint Opinion Mining (Joint-OpinionMiner, explained in section 3.4) of

comments of selected users. Next, select the influential users and then calculate polarity score of opinions (positive, negative, and neutral).

2.1. FOR each comment *c* in PM **DO**

2.1.1 Execute Tokenization, Data cleaning, POS-tagging, filter out named entities, Normalization (lemmatization)

2.2.2 Pass normalized comments and perform sentence-level opinion mining using VADER to remove objective sentences.

If *C* == “subjective”

C = Replace slang words from *C* using SlangSD to classify sentence as positive, negative, or neutral.

If *C* == “positive” and “neutral”

OMatrix = *C*

C = Emoji inclusion using VADER to classify sentence as positive, negative, or neutral.

If *C* == “positive” and “neutral”

OMatrix = *C*

Else

Do not select comments

2.2.3 FFT = Identifying frequent features from *C* in *OMatrix* using Apriori algorithm with minimum support 1%.

2.2.4 Identify opinion words for extracted features and determine semantic orientation.

2.2.5 Store nodes who have the most positive opinion in *R* matrix (influenced nodes matrix).

3. Construct a social network graph *G* consisting of influential nodes in *R* over relevant posts using Gephi

3.1 Construct influence graph $G = (V, E)$ from user stored at the end of step 2 as input to Twitter follower network API.

3.2 Assign influence probability score to each edge between nodes by computing the below formula:

$$P_{u,v} = \frac{(\#retweets\ of\ u\ on\ v) + (\#replies\ of\ u\ on\ v) + (\#quotes\ of\ u\ on\ v)}{\#tweets\ of\ v}$$

3.3 Select the initial set of the users before the opinion maximization process for influence spreading on neighbours.

4. In this step we perform opinion maximization of top k users using MLT

FOR each round q in 1,.....T **DO**

4.1. initialize seed set of users $S^{(q)} \leftarrow \emptyset$

4.2. select k users using greedy algorithm under MLT that maximize the total estimated opinion.

4.3. add newly activated users to seed set S.

end for

Output: Return seed set $(S^{(1)}, \dots, S^{(T)})$

END

Algorithm 1: ACOMax to generate opinion network and perform OM

Steps in the proposed ACOMax system: ACOMax will collect top 50 tweets related to the product i.e., iPhone12 in this case. For the case of example, here are some example tweets.



The image shows a vertical list of five tweets from Twitter. Each tweet includes a profile picture, the user's name and handle, the date or time, and the text of the tweet. The tweets are as follows:

- Nikki** (@nicholettenoele) · Jun 3: That iPhone 12 Pro battery got me throwing away my charger
- Alexandra Lay** (@alexandralay) · Jun 1: Honestly I love my iPhone 12 camera I'm so glad I broke my phone a couple weeks ago and had to upgrade
- Nokuthula** (@Nokuthula_Siba) · Jun 3: iPhone 12 makes sure with pictures 📷📷 I'm in love with my phone
- Lagarto Rodriguez** (@BBudsky) · 5h: I got an iPhone 12 and the camera is pretty cool. Here's a panorama of the backyard
- Jonathan Skillings** (@jeskillings) · 14h: Aside from hardware differences, the iPhone 12 Pro and 12 Pro Max aren't very "pro" in terms of software, says @trickholland. He'd like to see iOS 15 have more advanced features, esp. ones that take advantage of the larger Max screens. #WWDC21

Step 1: Collect tweets using TwitterAPI to create a profile matrix of the selected product (present in Table 16) from the tweets related to product. We store user information if count of retweets,

likes & comments is higher than selected threshold by seller, if Likes ≥ 50 , Retweets > 50 and Comments > 100 . For example, the first row shows a tweet for product=iPhone12 with User_id=1 who have posted the tweet, that has 130 likes. From this step we retrieve tweets which satisfies the criteria and store relevant user data and corresponding comments data in the matrix PM.


UserID	PublishDate	Quote	Retweet	CommentCount	Tweet
1	2021-03-02	130	100	50	I have an iPhone12, i upgraded from iPhone11. The overall new features of the phone is awsm.
2	2021-04-03	100	90	40	The camera quality is nicely done.
3	2021-05-19	50	60	30	I like the camera on my new iPhone 12  . Great for selfies!
4	2021-04-03	70	80	80	La calidad de la cámara está por debajo

Table 16: Profile matrix for p = iPhone12

Step 2: Extract user comments from multiple tweets and reconstruct profile matrix (Table 17) by considering **time-based** user tweets i.e., comments within a year. Retain comments only in the English language and filter out comments in other languages.


UserID	TimePosted	Quote	Retweets	Comment C	English
1	2021-03-02	130	100	I have an iPhone12, i upgraded from iPhone11. The overall new features of the phone is awsm.	1
2	2021-04-03	100	90	The camera quality is nicely done.	1
3	2021-05-19	50	60	I like the camera on my new iPhone 12  . Great for selfies!	1
4	2021-04-03	70	80	La calidad de la cámara está por debajo	0

Table 17: Updated Profile matrix for p = iPhone12

As we can see in [Table 17](#), comment by user 4 is not in an English language, so we will filter it out and reconstruct profile matrix by only taking relevant user_id, comments and Approve(A) for the next step. Thus, enhanced user profile matrix for product p is present in [Table 18](#).


UserID	A=(Quote + retweets + reply)	Comment C
1	230	I have an iPhone12, i upgraded from iPhone11. The overall new features of the phone is awsm.
2	190	The camera quality is nicely done.
3	110	I like the camera on my new iPhone 12  . Great for selfies!

Table 18: Enhanced Profile matrix PM

Step 3: Input profile matrix ([Table 18](#)) to clean user comments C using first step of Joint Opinion Miner algorithm which performs following steps in preprocessing and generate [Table 19](#) as a result:

3.1 Tokenization, for example, second row from [Table 18](#) as an input and will produce this

Output: 'The', 'camera', 'quality', 'is', 'nicely', 'done', '.'

3.2 Data Cleaning to remove symbols and stop words from the text using spaCy STOPLIST.

Output: 'camera', 'quality', 'nicely', 'done'


3.3 POS-Tagging to assign adjectives to the words i.e., ('camera', 'NN'), ('quality', 'NN'), ('nicely', 'RB'), ('done', 'VBN')]

3.4 Lemmatization to convert words to their first form, i.e., {camera, quality, nice, do}

3.5 Slang words are replaced into their original words using SlangSD lexicon library.

For example, awsm = awesome.

3.6 Replacing opinion displayed using Emoji with the text using demoji library.

For example,  = love.

User_id	A	Comment C
1	230	have, iphone12, upgrade, from, iphone11, overall, new feature, phone, awesome
2	190	camera, quality, nice, do
3	110	like, camera, new, iphone 12, love , great selfies

Table 19: Preprocessed user opinions

Step 4: Joint Opinion Miner for sentence-level opinion mining will take preprocessed comments as an input from **Table 20** to classify subjectivity of comments, modality detection to generate overall polarity score (positive, negative, or neutral) of the sentence by applying the VADER algorithm presented in section **3.4.3.1 VADER (Valence Aware Dictionary for sEntiment Reasoning)**.

User_id	A	Comment C	Polarity
1	230	have, iphone12, upgrade, from, iphone11, overall, new feature, phone, awesome	Pos
2	190	camera, quality, nice, do	Pos
3	110	like, camera, new, iphone 12, love, great selfies	Pos

Table 20: Sentence Polarity Score

Sentence-level opinion mining disregards objective sentences which only offer facts and not opinions.

Step 5: Input positive opinions data (**Table 20**) to Apriori algorithm for feature extraction in section **3.4.3** to identify frequent features, because those itemsets are likely to be product features. All the frequent features are stored in a dataset called **FeqFT**.

Id	Features
1	Camera {quality, camera}
2	Picture {resolution, camera, picture}
3	Screen resolution {resolution, screen}
4	Picture {resolution, video_call, camera, picture}

Table 21: Frequent features FFT

For example, in the case of “iPhone12”, some users may use “resolution” as a feature for “camera”, some use as “screen”. So, in **table 20**, comment of user 2 is “Camera quality nice do”. This leads to association between features i.e. (resolution = camera) whose support is $3/4 = 70\%$.

Step 6: Once features are extracted (**Table 21**) from user comments, we extract opinion words (OW) near the features to identify polarity of user opinion about the features using WordNet (Miller et al., 1990). For example, in **table 20**, comment of user 2 is “Camera quality nice do”. So, OW= {Nice}.

Step 7: Next, input OW to Identify the Semantic Orientation (SO) i.e., positive, or negative of OW using WordNet and then rank features and polarity measure (section **3.4.6**) and generate **Table 22**.

User_id	A	C	Polarity	Opinion Score
1	230	overall new feature awesome	Positive	4
2	190	Camera quality nice	Positive	3
3	110	Camera love great selfies	Positive	5

Table 22: Feature Polarity Score

Example, SO={nice, positive} and then assign scores in the range [-4, 4] to each of the opinion words. For example, “love”=strong positive opinion, “nice”= positive but not as strong. Second, we consider list of inversion words. For example, “not good” is a negative opinion. If an inversion word appears multiply the original score of the opinion word by -1.

Step 8: Build the Twitter network graph G for twitter follower network (presented in [section 3.4.7](#)) from collected tweets and users and retrieve users’ friends from the list of users using Twitter API that returns friend IDs of a specified user. Each user id and friend id pair is inserted into User_friend table. We collect tweets of top 50 users with most followers in our sample tweets.

user_id	friend_id
1	4
2	5
3	7

Table 23: User friend Table

Step 9: Construct community opinion network graph G from users selected in step 9, add edge between users if they are friends in Twitter and assign influence probability $p_{u,v}$ to edges between every node using below formula:

$$P_{u,v} = \frac{(\#retweets\ of\ u\ on\ v) + (\#replies\ of\ u\ on\ v) + (\#quotes\ of\ u\ on\ v)}{\#Total\ tweets\ of\ v}$$

Step 10: Finally, input community opinion network graph G from step 10 as input to Greedy Algorithm ([section 3.7](#)) to obtain select k-best seed set (S(1),.....S(T)). Repeat step 10-11 until it reaches the terminal round number T or there are no more inactive users in the network.

Features to be analyzed with Definition: In our thesis, we study a friendship network, Twitter, and we have classified the social networks via products. Examples of products include a product

such as “iPhone” or “Samsung. The popularity of a given product post is represented by following different phenomena that we have found analyzing the friendship network.

Definition 3.1 Likes: We define Likes (L) by the number of likes on a given product by clicking a like button on a post. **Example:** if promoter wants to decide $E > 500$, then our system will extract the relevant posts that have likes more than 500.

Definition 3.2 Retweet: We define retweet (Rt) by determining how many users share the product by forwarding it to other people in their network.

Definition 3.3 Comments: We define comments (C) by determining the number of people who comment on a given post. In our proposed system, we use a combination of reach and comments by calculate the number of different users commenting on the related post. $C = nC_1$, where nC_1 is the number of individual comments.

Definition 3.4 Multi-round Linear Threshold: Multi-Linear Threshold (MLT) is a multi-round diffusion model like the conventional LT model (Kempe, Kleinberg, & Tardos. 2003), in which each user u_i in a weighted social network G is associated with a threshold $\theta_i \in [0, 1]$. Before the first round of MLT, all users are inactive towards the target product. At round t of the MLT, an inactive user u_i becomes active when: $P_{ij} \in C^{(t-1)} w > \theta_i$, where $C^{(t-1)}$ is the set of active users before round t starts. At each round, the influence only propagates layers of neighbors before the next round starts. All active users will stay active and cannot back to inactive status.

Definition 3.5 Positive opinion: The goal is to find a set of seed users to maximize the overall opinion spread toward a target product. For that, we select positive comments by determining the number of users people spread praises about a product, for example, “I love the new iPhone12Pro.”

Definition 3.6 Negative opinion: The spread of negative opinions harms the product reputation and washes out the positive opinions. In this case, maximizing the spread over a social network can no longer achieve an optimal outcome. For that, we define negative opinions by determining the number of users people don't like the product, for example, “Buying this product is wastage of time”.

Definition 3.7 Marketing Strategy: In this paper, we are concerned about the seed user selection based upon opinions of users given on multiple products. For simplicity, we refer to the marketing strategy of the target product p as the vector of seed users sets $s = (s^{(1)}, \dots, s^{(T)})$, where $S^{(q)}$ is the set of seed users been selected for round q , and T the terminal round number.

3.3 Social Post Miner (SPOM)

SPOM takes following inputs to the framework for selecting posts about the product p:

1. Threshold for Likes (l) which is the minimum number of users (v_l) connected to the user (v) who likes the product-post.
2. Threshold for Comments (C) which is the minimum number of posts (w) the node (v) has commented on the product p.
3. Threshold for Retweets (Rt) which is the minimum number of times posts (w) is shared.

SPOM Model: Proposed solution framework ACOMax calls SPOM to extract relevant nodes for a product p and filter according to higher influential score determined by Likes lc, Comments cc and Retweets Rt in Algorithm 1. SPOM then extracts and filters relevant posts based on initially set threshold. Next, ACOMax calls OpinionMiner to fetch all the opinions for each relevant post of each relevant node v and apply sentence and word segmentation and some cleaning such as stemming, removing stop words under preprocessing of the data and next, identifying the polarity of the comment, i.e., the comment expressing positive, neutral, or negative opinion. Finally, we identify the relationships among nodes on a product p and how they influence to each other to create the community opinion network.

Algorithm 2: SPOM called by ACOMax for Identification and Selection
Input: Product p, likes lc (minimum number of users connected to node V), Retweets rt and Comments cc
Output: Profile matrix PM of user and item and posts by Comments Matrix C.
Begin 1. Execute Tweepy to get tweets with lc, rt and cc using Tweepy. 2. PM matrix = [U, Pt , rt, cc, lc] #Initialize Matrix for retrieved posts Pt 3. If PM(lc) < 100 and PM(rt) < 100 and PM(cc) < 100 3.1 Remove V from PM 4. Else 4.1 PM = [Pt, user u, product p, lc, rt, cc] #Add count of likes, retweets, comments, and users related to posts in the matrix 4.2 C = [Pt, C]. #Add comments in the matrix 5. End if 6. Return PM END

Example to extract posts and users to create profile matrix of the product: Twitter is an enormously popular microblog on which clients may voice their opinions. Opinion investigation of Twitter data is a field that has been given much attention over the last decade and involves dissecting “tweets” (comments) and the content of these expressions (Alsaedi, & Zubair, 2019). SPOM consists of three major steps Identification of posts, Preprocessing of data, and Extraction of nodes.

1. Identification of Nodes and Posts: For a given product p , we execute a search mechanism over the social network G with the product p . We execute Tweepy to search and retrieve posts related to the product over the social network using Twitter API to identify relevant tweets and users on product p .

2. Preprocessing: From the above step we retrieve and select the posts related to the product. In preprocessing, we have the matrix PM (Table 24) and is sorted in a descending order, Now let us set a threshold Likes (lc)=100, meaning that we are looking for nodes having $lc \geq 100$ from this dataset.

User_id	Likes	Retweets	CommentsCount	TweetDate
396269740024	130	50	300	2021-02-02
296269740125	100	40	200	2021-01-02

Table 24: Profile matrix of posts

We select posts and users and their comments on product p based on the set threshold i.e., $lc > 100$, $rt > 40$, $cc > 200$ and timestamp of the post within a year.

3. Extraction: We then extract time-based user data i.e., comments, likes and retweets from every post for a specific product automatically from the given social network. We then store the relevant nodes data, posts data, and corresponding users’ comments data in the updated profile matrix PM consisting of data like Table 25. Approve $A = (qt + rt + cc)$

T_id	User_id	Tweet	TweetDate	Approve
124	429326	I have an iPhone12, i upgraded from iPhone11. The overall new features of the phone is awsm.	2021-02-02	61153
125	223952	The camera quality is nice.	2021-01-02	33899

Table 25: Final PM table after extraction

Below is the query to retrieve all the data using SPOM:

```

json_response = https://api.twitter.com/2/tweets/search/all
query_params={'query': 'iPhone12', 'start_time': '2020-08-01'}
for status in json_response[data]:
    'UID':str(status._json['user']['id_str']),
    'UserCreatedDate':(status._json['user']['created_at'])
    'ULocation': str(status._json['user']['location']),
    'FollowerCount': str(status._json['user']['followers_count']),
    'FriendsCount': str(status._json['user']['friends_count']),
    'TweetDate': (status._json['created_at']),
    'Tweet': str(status._json['full_text']),
    'LikesCount': str(status._json['favorite_count']),
    'RetweetCount': str(status._json['retweet_count']),
    'CommentsCount': str(status._json['reply_count'])
}

```

3.4 Joint Sentiment Analysis of opinions using Opinion Polarity Miner (OPM):

Our next task is to find useful comments on the posts, analyze the sentiment of comments and decide whether the post has a positive or negative impact on the product. For each post of each node Opinion Polarity Miner (OPM), identifies opinion comments across all the comments on that post w, identifies the semantic orientation (SO) of the comments, and measure the polarity of the comments. Opinion Polarity Miner (OPM) considers three major features on users' comments: Positive opinion (r_p), Negative opinion (r_n), neutral opinion ($r_{neutral}$).

Algorithm 3: Opinion Polarity Miner (OPM) called by ACOMax in step 2.

Input: Product p, Comments C, and Post W
Output: Features set, Polarity score matrix for each post and comment.
Begin 1. Matrix C consisting of all comments on the given product p 2. For each c in C 2.1 Data cleaning and Tokenize comments (algo in page 3.4.1) 2.2 Opinion Extraction (algo in page 3.4.2)

2.3 SA(c, SO) = Semantic analysis of comments (algo in page 3.4.2)

3. End For

4. For each c in SA #calculate number of positive, negative, and neutral comments

4.1 If SA(SO) = positive opinion

4.1.1 POS = POS + 1

4.2 Else If SA(SO) = negative opinion

4.2.1 Neg = Neg + 1

4.3 Else

4.3.1 Neutral = Neutral + 1

4.4 End If

5. Polarity score $Qz = ((POS + Neutral) - Neg) * 100/CM$ #Popularity score of tweets

6. Co = Neg/POS #controversiality of a post

7. OPM[Posts] = [Co, Qz]

END

Algorithm 3: Opinion Polarity Miner (OPM)

Example to extract user opinions to perform sentiment analysis: To explain the Joint sentiment analysis algorithm step by step, let us consider user reviews from the **Table 25**[Error! Reference source not found.](#) as input and perform following tasks:

3.4.1 Preprocessing - Data cleaning is a complex set of tasks that takes as input and produces as output a single, clean data set. In our thesis we use NLTK (Loper & Bird, 2002) for cleansing tasks which include removal of stopwords and symbols, Lemmatizing and to deal with word variations and misspelling. If a post contains most of the comments having any of the data that does not get selected after these steps, we ignore the comments and take Engagement (E) i.e., likes as polarity measure.

- **Tokenization** is a straightforward Natural Language Processing where words are delimited by blank spaces and punctuations. A tokenizer divides a string into substrings by splitting on the specified string.
- **Lemmatization:** Lemmatizing of words are the approaches that produces the normalized form of a word in the text. word Lemmatizing is a technique that gets the root (base) of the word in the text. It normalizes the word by removing the suffix from the word, which gives root meaning for the word.
- **Remove Stop Words:** While processing natural language, some of the words which have high occurrence in the document are stop words (e.g., 'and' 'the', 'am', 'is'), which only

have little emotional meaning and it do not affect the sentiment score when applied to lexical resources. Therefore, it is common practice by many researchers to filter stop words in the domain of analyzing sentiment from the document.

Below are the technical details of how these works:

Tokenization	<ul style="list-style-type: none"> • Segment text into unigrams, using <code>spacy.load()</code> to return a language object containing components needed to process text usually called NLP. • Segment text into bigrams, breaking into set of two words.
Remove Stop words and cleaning of Data	<ul style="list-style-type: none"> • List of most common words which are often a noise rather than features such as ('and' 'i', 'are' etc.). • We removed symbols such as ('\$','!') from the text, then used <code>ENGLISH_STOP_WORDS</code> as a stop list and a custom list of ignore words ('our', 'you'). • We have also removed punctuations (single and double quotes, commas, whitespaces etc.) and dropped duplicate rows from the dataset.
Lemmatization	<ul style="list-style-type: none"> • Lemmatization converts words in the second or third forms to their first form variants unlike stemming which only removes deriving affixes ('ed', 'ly'). • <code>spacy</code> determines the part-of-speech tag by default and assigns the corresponding lemma.

Table 26: Data preprocessing of comments

Algorithm 4: Data Cleaning, Tokenization, Removing Stop words and Lemmatization by OPM.

Input: Comments C

Output: cleanMatrix matrix with comments

Begin

1. `cleanMatrix = []` #initialize matrix to store tokens and clean data

2. Initialize `STOPLIST = stopwords.words('English')`

3. `SYMBOLS = " ".join(string.punctuation).split(" ") + ["-", "...", "'''", "!", "#", "$", "%", "&", "(", ")", "*", "+", "-", ":", "/", ":", ";", "<", "=", ">", "?", "@", "[", "]", "^", "_", "`", "{", "|", "}", "~"]`

2. For each tok in C

2.1 `tok.lower().strip()` #convert comments to lower case

2.2 Identify the tokens using integer offsets (`start_i, end_i`)

#s(`start_i:end_i`) is the corresponding token.

3. If `tok != "-PRON-"` else `tok.lower_()`

3.1 `cleanMatrix = tok`

3.2 `cleanMatrix = [tok for tok in cleanMatrix if tok not in STOPLIST]`

```
3.3 cleanMatrix = [tok for tok in cleanMatrix if tok not in SYMBOLS]
END If
End For
4. return cleanMatrix
END
```

Algorithm 4: Tokenization, Stop word removal, data cleaning and Lemmatization by OPM

3.4.2 Identification of Opinion Words - The identification process has three steps.

1. The first step is to use a part-of-speech tagger to identify phrases in the input text that contains adjectives or adverbs (Brill, 1994).
2. The second step is to perform joint opinion mining, sentence-level opinion mining will remove objective sentences and then we will perform feature-level opinion mining to extract product features on which many people have expressed their opinions.
3. The third step is to extract opinion words from the comment. For example, “This picture quality is awesome”, where “awesome” is the effective opinion of picture quality.

Part of Speech Tagging (POS-tagging) - POS tagging is the part-of-speech tagging (Manning & Schutze, 1999) from Natural Language Processing (NLP) which reflects word’s syntactic categories and helps to find opinion words. Common POS categories in English are noun, pronoun, verb, adverb, adjective, preposition, conjunction, and interjection. We parse each sentence and yield the part-of-speech tag of each word (whether the word is a noun, adjective, verb, adverb, etc.) and identify simple noun and verb groups (syntactic chunking).

Example:

```
1. inputText = "And now for something completely different"
2. pos_tag(inputText)
3. Output: [('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something',
'NN'), ('completely', 'RB'), ('different', 'JJ')]
```

After POS tagging is done, we need to extract features that are nouns or noun phrases using the pattern knowledge. And then, we focus on identifying domain product features that are talked about by customers by using the manually tagged training corpus for domain features.

3.4.3 Sentence-level Opinion Mining: In this step, we will get the overall sentiment of the sentence and we will pass positive opinions for feature extraction. This approach relies on the use

of a lexicon. A lexicon contains entries with data about words (or word stems); data entered about a word could include its part(s) of speech, spelling variants, inflectional variants, encoded syntactical information, and so on. There are a variety of sentiment lexicons out there geared specifically towards sentiment analysis. This level of analysis is closely related to subjectivity classification (Wiebe, Bruce, and O'Hara, 1999), which distinguishes sentences (called objective sentences) that express factual information from sentences (called subjective sentences) that express subjective views and opinions. In this step, we will get the overall sentiment of the sentence and we will pass positive and neutral opinions for feature extraction.

- Selecting Subjective Reviews.
- Slang word inclusion: Replace slangs using Slang Sentiment Dictionary, SlangSD (Wu, Morstatter, & Liu, 2018).
- Emoji Inclusion: Conversion of emoticons to texts using demoji.
- Sentence Modality Detection (using VADER). Example, the sentence “The iPhone’s call quality is good, but its battery life is short”. The sentiment on iPhone’s call quality is positive, but the sentiment on its battery life is negative.

Steps leading till Sentence-level opinion mining: 1. tokenization 2. POS-tagging 3. reduce text to nouns, adjectives, verbs, adverbs (optionally filtering out named entities) 4. normalization: stemming and/or lemmatization. After preprocessing the text is reduced to its content’s words in a normalized form. Now they are ready to be fed into the VADER for assigning sentence polarity.

3.4.3.1 VADER (Valence Aware Dictionary for sEntiment Reasoning)

VADER (Hutto & Gilbert, 2014) is a lexicon and rule-based sentiment analysis method built for analyzing sentiment from social media with more than 9000 lexical words. Vader combines sentiment lexicons (i.e., list of lexical words) and sentence characteristics (semantic orientation of words) to determine a sentence polarity. The Positive, Negative and Neutral scores represent the proportion of text that falls in these categories.

For example: “The iPhone is super cool”. Our sentence was rated as 67% Positive, 33% Neutral and 0% Negative.

1. In our case, lexicon ratings for each word in VADER is “super(2.9) and cool(1.3)”=x (4.2).
2. VADER normalizes the score between [-1 to +1] to get Compound C using this equation 7:

$$x = \frac{x}{\sqrt{x^2 + \alpha}}$$

Sentiment Metric	Score
Positive	0.674
Neutral	0.326
Negative	0.0
Compound C	0.735

Equation 7: Calculate Compound score of a sentence where x = sum of valence scores of words, and α = Normalization constant (default value is 15)

3. $x = 4.2/\sqrt{(4.2)^2 + 15} = 4.2/5.71 = 0.735$.

4. A sentence is a positive sentiment if $C \geq 0.05$, and a negative sentiment if $C = 0.05$ and a Neutral sentiment if ($C > -0.05$ and $C < 0.05$). Hence $x = 0.735$, **sentence is a positive opinion.**

3.4.3.2 Slang word replacement & Emoji detection: Now, we will include slang word removal and emoji detection using VADER and SlangSD (slangsd.com/data/SlangSD.zip).

1. SlangSD is the first sentiment lexicon for slang words, which provides over 90,000 slang words/phrases and their sentiment scores. The user created short form is called as slang words. SlangSD will replace slangs to their proper form.

For example, tmrw=tomorrow, thx=thanks , awsm = awesome.

2. Emoji Detection: Replacing opinion displayed using Emoji with the text in demoji. For example,

```
{'😞': 'tired face',
'😕': 'confused face',
'😴': 'yawning face',
'😌': 'relieved face'}
```

After assigning text to the emoji, we will calculate the polarity score of the sentence. For Example,

Input: 'I am 😊 with the new screen size of iPhone12'

Output: 'neg':0.0, 'neu':0.476, 'pos':0.524, 'compound': 0.6705}

Next all the subjective sentences with positive polarity will be passed for feature-level mining using Apriori algorithm.

3.4.4 Feature Identification using Apriori Algorithm - In this step, product features on which users have expressed their opinions on their comments. For example, if the comment about iPhone12 is "The sound system is very sophisticated," then "sound system" is the feature of the product "iPhone12". Our proposed thesis focuses on finding features that appear explicitly as

nouns or noun phrases in the comments. We mainly focus on finding frequent features in comments, i.e., those discussed by many users.

In our thesis, an itemset is a set of words or phrases that occur together in some sentences. From our POS-tagging step, we have a transactional set of nouns or noun phrases. Using those sets, we apply association rule miner based on the Apriori algorithm (Agrawal & Srikant, 1994) to find association rules. We want to identify whether the frequent features have a positive semantic orientation or negative orientation in this step. For each frequent feature, we find the nearest opinion word, and its orientation, the opinion word's orientation becomes the orientation of frequent features. Then we check for the negation words (e.g., not, never, did not, do not) within a five-word distance in front of an opinion word (Jin et al. 2009). We define the rules for negation words are:

Rule1: A negation word appears in front of conjunction (e.g., and, but). Example – "This colour is good but expires soon." This sentence mainly expresses a negative opinion. So, if the opinion word is in front of the corresponding feature and the conjunction "but/except" appears between the opinion word and feature, then the opinion orientation for the feature is updated with the opposite of its initial orientation.

Rule2: Negation of negative opinion word is positive, e.g., "no problem." Negation of positive opinion word is negative, e.g., "not good." Negation of neutral opinion word is negative, e.g., "does not work" where "work" is a neutral verb.

The goal is to find the set of frequent features by iteratively computing support of each itemset in the candidate set C.

- **Frequent feature itemset:** If feature appears and satisfy a minimum support in the comments.

Example: In the case of “iPhone12”, In our example, Candidate set **C1 = { resolution:4, camera:3, picture:2, screen:1, video_call:1 }**

ID	Features
1	{resolution, camera}
2	{resolution, camera, picture}
3	{resolution, screen}
4	{resolution, video_call, camera, picture}

Input: 2), set of nouns
Output:

Minimum support (e.g., phrases (POS tags).
Frequent features.

Step 1: Eliminate features less than minimum support. i.e.,

- $L1 = \{\text{resolution, camera, picture}\}$

Step 2 : $C2 = L1$ (Apgen-join) $L1$

C2: $\{\text{resolution, camera:2}\}, \{\text{resolution, picture:2}\}, \{\text{camera, picture:2}\}$

- $L2 = \{(\text{resolution, camera}), (\text{resolution, picture}), (\text{camera, picture})\}$

Step 3: $C3 = L2$ (Apgen-join) $L2$

C3: $\{\text{resolution, camera, picture:2}\} \Rightarrow L3$, frequently occurring features = $\{L1 \cup L2 \cup L3\}$.

This leads to association between features i.e. (**resolution=camera**). If resolution=30, camera=28, both=25, total comments=100. **Support** = $|\text{Rule}| / |\text{total}| \Rightarrow 25/100 \Rightarrow 25\%$

If a comment sentence contains a set of features, then for each feature, we compute an orientation score for the feature. **A positive opinion word has a score (+1), neutral opinion word has a score of (+0.5), and a negative opinion word has a score (-1)**. All the scores are then summed up. If the final score is positive, the semantic orientation of the comment is positive. If the final score is negative, then the semantic orientation of the comment is negative.

3.4.5 Extraction of Opinion Words & Semantic Orientation - Opinion words extraction phase has two major tasks:

1. Extract opinion words around frequent features (Algorithm 6), in this phase, we take the list of tokens with corresponding POS-tags, and search for if it contains adjective words and/or frequent features.

Algorithm 6: OpinionExtraction called by OPM.

Input: CleanMatrix, POS, WordNet

Output: Set of opinion comments (OC) along with features FT

Begin

1. Frequent Features matrix (FF) = Association Rule to identify frequent features in comment c

2. For each c in CleanMatrix

3. For each c in FF

3.1 If FF(frequent feature FFT) = POS(c). #Opinion comments extraction for frequent features

3.1.1 OE[OC] = POS

3.1.2 OE[FT] = FF[FFT]

4. For each OC in OE matrix

4.1 If OC has synonyms in WordNet List #Identify semantic orientation of comments

4.1.1 OE[OC] = semantic orientation OR

4.1.2 Add OC with orientation in OE

4.2 Else If OE has antonyms a

4.2.1 OE[OC] = a's semantic orientation OR

4.2.1 Add OC with orientation in OE

5. Return OE(c, OC, FT, OR)

END

Algorithm 6: Opinion Word Extraction

Steps for Opinion Word Extraction: We use WordNet to utilize the adjective synonym set and antonym set to identify the opinion expressed by the word (i.e., positive, or negative opinion).

1. First, a set of opinion words (adjectives, as they are normally used to express opinions) is identified. If an adjective appears near a product feature with nouns/noun phrases in a sentence, then it is regarded as an opinion word. We can extract adjectives as opinion words from the comment using the extracted features, **Example:** “*The strap is horrible and gets in the way of parts the camera you need access to*”. For the first sentence, the feature, ‘*strap*’, is near the opinion word ‘*horrible*’.

2. Identifying semantic orientation of it (Algorithm 7), in this phase, we have a list of extracted opinion words in comment text and need to identify the semantic orientation (positive, negative, or neutral) of each extracted phrase of each comment. We extract phrases containing adjective, adverb, verb, and noun that imply opinion. We also consider some verbs (like, recommend, prefer, appreciate, dislike, and love) as opinion words. Some adverbs like (not, always, really, never, overall, absolutely, highly, and well) are also considered. Therefore, we extract two or three consecutive words from the POS-tagged review if their tag conforms to any of the patterns.

1. We collect all opinionated words like (adjective, noun), (adjective, noun, noun), (adverb, adjective), (adverb, adjective, noun), (verb, noun), and so forth from the processed POS-tagged review.

2. The resulting patterns are used to match and identify opinion phrases from reviews after the POS tagging. However, there are more likely opinion words/phrases in the sentence, but they are not extracted by any patterns. From these extracted patterns, most of adjectives or adverbs imply opinion for the nearest nouns/noun phrases. Table below describes some examples of opinion phrases.

A few examples of extracted opinionated phrases.

(Adjective, noun)	(low battery), (good memories), (awesome camera), and so forth
(Adjective, noun, noun)	(high quality pictures)
(Adverb, adjective)	(extremely pleased), (very easy), (really annoying), (absolutely amazing), and so forth
(Adverb, adjective, noun)	(very compact camera), (very good pictures), and so forth
(Adverb, verb)	(personally recommend)
(Adverb, adverb, adjective)	(not so bad), and so forth
(Verb, noun)	(recommend camera), (appreciate picture), and so forth
(Verb, adverb)	(perform well)

Algorithm 7: Semantic Orientation (c, OC) called by OPM

Input: List of opinion words OE // opinion words with corresponding features

Output: Semantic orientation SO of comment c

Other: orientation // positive = 1, negative = -1, neutral = 0

BEGIN

1. Set orientation = 0
2. **For** each opinion word OC in OE
 - 2.1. orientation1 = orientation of OW
 - 2.2. **If** any negation word appears closely to OW
 - 2.2.1. orientation1 = opposite orientation1
 - 2.3. orientation = orientation + orientation1
3. **End For**
4. **If** orientation > 0
 - 4.1. CSO[SO] = positive
5. **Else If** orientation < 0
 - 5.1. CSO[SO] = negative
6. **Else**
 - 6.1. CSO[SO] = neutral
7. **End If**
8. **Return** CSO(c, SO)

END

Algorithm 7: Semantic orientation

To compute the polarity measure of a comment, we identify the semantic orientation and polarity of the opinion words stored in the table OE. For each opinion word OW_i in the list OE, we search its synonyms or antonyms in WordNet and collect its orientation. For example, $OE[1] = \{\text{love, positive}\}$, $OE[2] = \{\text{cool, positive}\}$. If a negative word comes in front of an opinion word, we consider the semantic orientation of the opinion word is its opposite orientation. According to table OE, we have all the orientation i.e., the polarity (opinion score) of individual comment.

Opinion Score: The final score is positive; if the sum of SO is greater than 0, else negative; if the sum is zero, then SO is neutral.

$$\text{Opinion Score} = \Sigma (\text{sum of scores of opinion words in each review})$$

Equation 8: Calculate overall score of feature opinion

User_id	A	C	Polarity	Opinion Score
1	230	overall new feature awesome	Positive	4
2	190	Camera quality nice	Positive	3
3	110	Camera love great selfies	Positive	5

Table 27: PM containing users with positive opinions about p=iPhone12

3.4.6 Generating Community Opinion Network Graph: Before this step we have collected tweets and users and saved them into a profile matrix PM.

Input: List of top positive users from **table 27**, Twitter follower network of selected users using TwitterAPI (<https://api.twitter.com/2/users/:id/followers>).

Output: Community opinion network graph $G = (V, E)$

Procedure:

Step 1: From the previous step, we get a list of users who have positive opinion about the product. Now, we like to know all their friends. Twitter API returns friend IDs of a specified user but no more than 5,000 IDs at a single request. We need to call multiple times if that user has more than that. Also, Twitter has rate limits. It allows only 15 requests per 15-minute window. Basically, 1 request per minute.

- **Get follower friend id's:** For each id, call api.twitter.com/2/users/:id/followers. This API url returns list of follower friend ids. Each user_id and friend_id pair is inserted into User_friend table.

User_Friend (user_id, friend_id)

Step 2: Create social network opinion graph of top 100 users with most followers.

- Below is the query used to create Node and edges.

Graph_Friend_Edge (Source, Target)

Graph_Friend_Node (id, label)

- Populate the edge table with only top users.

```

Insert into graph_friend_edge(source, target)
select user_id, friend_id from user_friend join user u1 on friend_id=u1.id join user u2 on
user_id=u2.id
where user_id in (Select friend_id from user_friend group by friend_id order by count(*) desc limit
100)
and friend_id in (Select friend_id from user_friend group by friend_id order by count(*) desc limit
100)

```

- Then, populate the node table with following query:

```

insert into graph_friend_node(id, label)
select n.id from(select source id from graph_friend_edge union
select target id from graph_friend_edge) n join user u on n.id = u.id

```

Step 3: Construct community graph $G = (V, E)$ using step 2.

Input: Twitter follower network with tuple (user_id, friend_id) meaning u follows v.

Output: Community influence graph $G = (V, E)$

Step 3.1. $G = \emptyset$ // initially G is an empty graph

Step 3.2. $G.addEdge(u, v)$, for each tuple (u, v) in G add a directed edge “graph_friend_edge” between “graph_friend_node” to G.

For each tuple (u, v) in the Twitter follower network, it adds a directed edge (v, u) to an initially empty directed graph (lines 2-3). After all the tuples are processed, it outputs an influence graph without probabilities (Figure 21).

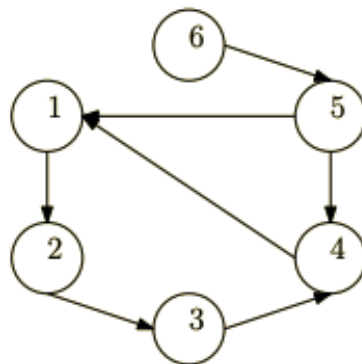


Figure 21: Community Influence Graph without probabilities

Step 3.3. Combine Twitter retweets, replies and mention to compute influence probability using formula for Learning Influence Probabilities as Edge Weights from Twitter by (Cao & Ezeife, 2014) which works as follows:

1. Concatenate Mention, Reply, and Retweet as approve (A) into one table named Tri and group Tri by columns u and v.
2. The updated tri by computing the sum of A for each group.
3. Left-Join Tri and Tweets on column v to obtain a new table named TriTweets.
4. Add a new column named p to TriTweets, where $p = A/t$ (total tweets of u)
5. Drop columns A and t from TriTweets, and Outer Join TwitterFollower and the updated TriTweets to obtain the influence probability table, where each tuple (u, v, p) means the probability that node v influence on node u is p.

$$\beta_1 = \frac{(\text{Number of } (retweets) \text{ of } u \text{ on } v)}{\#tweets \text{ of } v} \quad \beta_2 = \frac{(\text{Number of } replies \text{ of } u \text{ on } v)}{\#tweets \text{ of } v}$$

$$\beta_3 = \frac{(\text{Number of } quotes \text{ of } u \text{ on } v)}{\#tweets \text{ of } v}$$

$$P_{u,v} = (\beta_1 + \beta_2 + \beta_3)$$

Equation 9: Calculate Influence probability between users

Step 3.4 Assign influence probabilities (step 3.3) to each edge in G.

User id	Friend id	$P_{u,v}$
1	2	0.8
2	3	0.98
3	4	0.56
4	1	0.82
5	1	0.70
5	4	0.78
6	5	0.80

Table 28: User-User relationship with influence

Step 3.5 return community opinion network $G = (V, E)$ (figure 19)

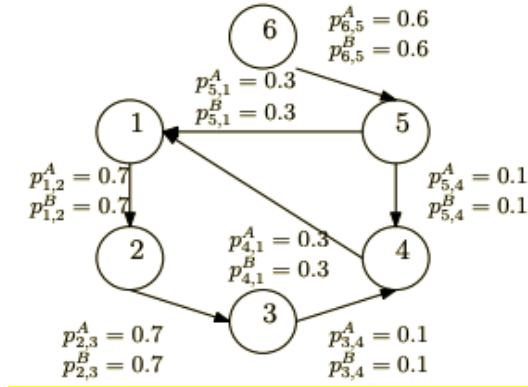


Figure 22: Community opinion network Graph G

3.5 Greedy Algorithm for Opinion Maximization under Multi Linear Threshold (MLT)

model: In MLT diffusion model, a user u can influence his neighbors with specific influence weights and has a threshold θ , which denotes the minimal required influence to be activated by the neighbors. The weights of the directed edges quantify the influence propagates from u to v . Following CONE (Liu, Kong & Lu, 2018), (Zhang et al., 2016) and (Zhang, Dinh & Thai, 2013), we randomly generate the set of thresholds of users $[\theta_1, \dots, \theta_n]$ from uniform distribution within range $[0, 0.1]$ to match the real-world opinions of other users in the social network on the target product before calculating the total opinion spread. On Twitter, users express its opinion by posting likes, retweet, and comment on original tweets.

We assume user u is likely to influence user v only in a fixed-size time-frame T since u expresses its opinion. In the opinion maximization problem, we consider Au as the total number of tweets the user u posts in a certain time period T . The action history contains three kinds of interactions, which are likes, comment and retweet/quote to calculate weights of the directed edges. If user v reacts to user u , it means u has successfully passed the information to v , say u has influenced v . The influence factor ($\text{infl}(u, v) \geq 0$) from user u to v is defined as the ratio of $v \rightarrow u$ reactions to the total actions performed by u .

Given the seed budget k , we select $k^{(q)}$ users who can maximize the total opinions of the final activated users.

Input: Given the seed budget k , set of seed users S .

Output: Users with largest estimated opinion is added to S .

Example: Fig 23 illustrates influence spread process in an influence graph.

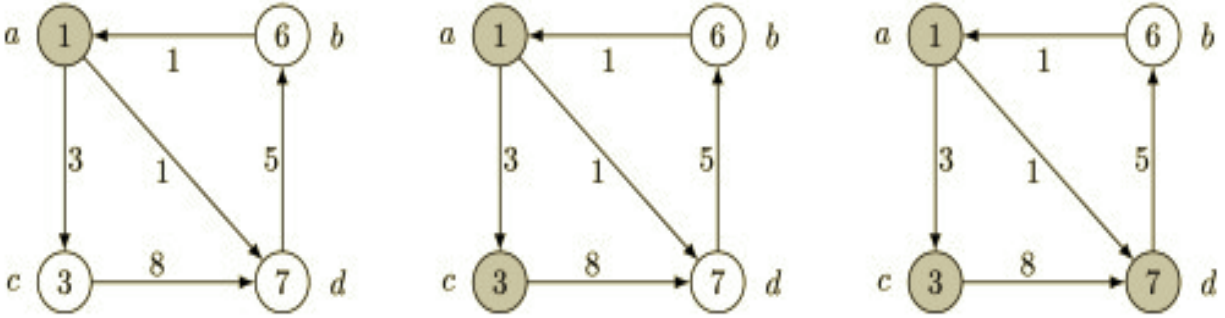


Figure 23: Graph G for greedy algorithm

Step 1: We scan all the inactive users in the network $k(q)$ times. In every scan, we add each inactive user to the temporary seed set S and propagate the influence to obtain the final activated users. The spread of influence $F(X)$ starts from the seed $X = \{a\}$.

Step 2: In this step, we obtained $F1(X) = \{a, c\}$. Among all users in the scan, the one with the largest estimated opinion is added to S . We repeat the scan until run out of budget or there are no more inactive users in the network.

Step 3: In the last step, we get $F2(X) = \{a, c, d\}$ and we compute total opinion. **Total Opinion = (total opinion achieved at the end of current round) – (opinion before round).**

Step 4: $S = \{a, c, d\}$ activated from step 3, influence is propagated, and the newly activated users express their opinions on the target product. Store the updated set S of activated users in C .

Step 5: Repeat step 1-3 until it reaches the terminal round number T or there are no more inactive users in the network. Repeat the scan until run out of budget or there is no user to activate.

Output: Return $(S^{(1)}, \dots, S^{(T)})$

3.6 A Walkthrough Example with Comparison from OBIN:

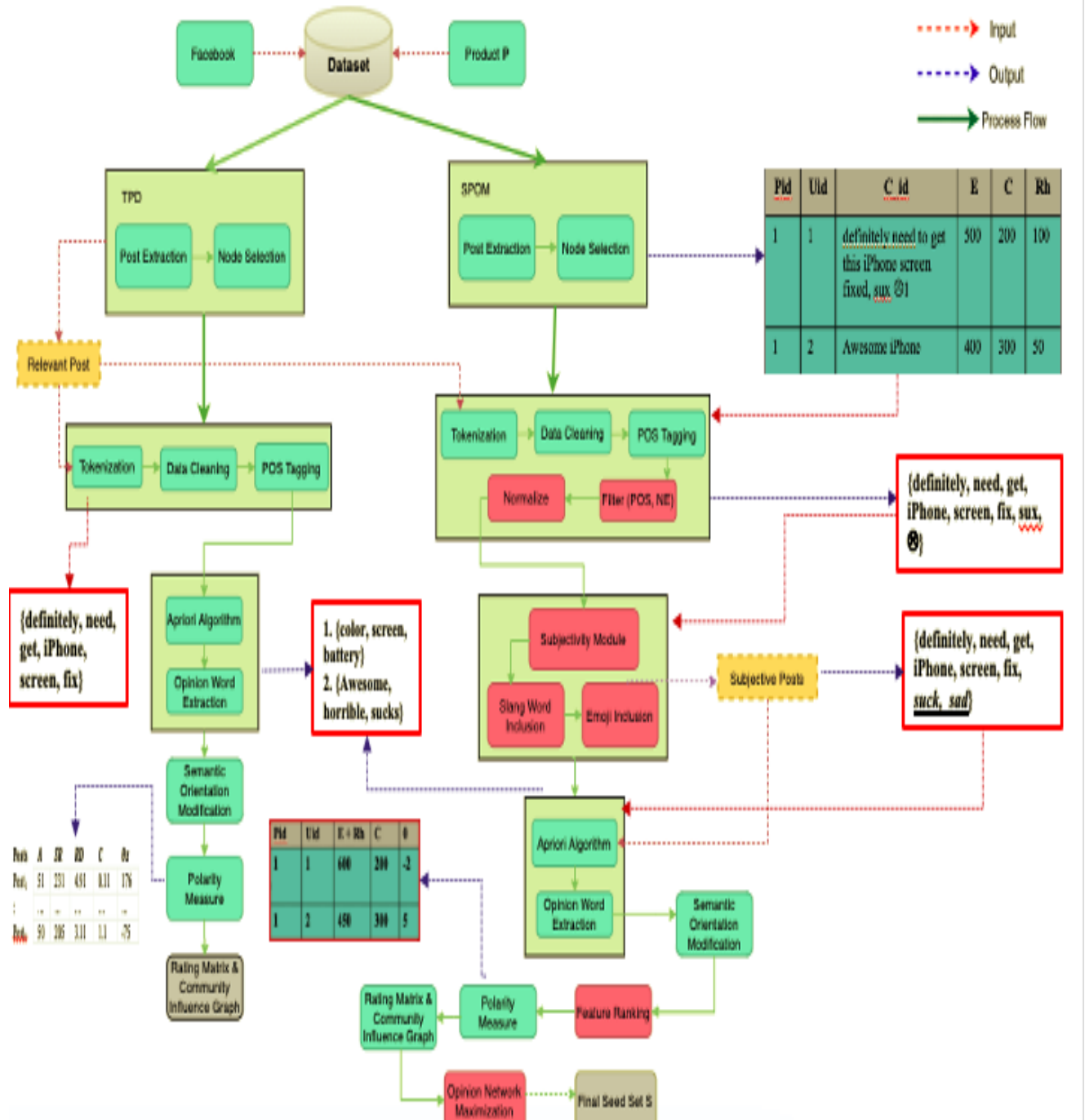


Figure 24: Comparison between OBIN and Proposed ACOMax

Steps of OBIN system

Input: Facebook GraphAPI to extract posts, topic z .

Output: Community Influence Matrix IMAT and Influence Graph

Step 1: Collection of Data from Facebook Graph API with FQL to collect all the relevant nodes for a given topic $z = \text{iPhone}$ and collect topic categories (Table 1) as well.

Output: In table 2, node with id “130422069” and term “iPhone” with 3116728 friends, and we can visit his profile by “iPhone.Page” link

Cat_id	Cat_name
2603	Non-profit Organization
2201	Product/Service

Table 1: Topic Categories

Node id v	Term	A	Link
130422069	iPhone	3116728	iPhone. Page
110054999	iPhone 4	1435239	Iphone-4

Table 2: Nodes and data for $z = \text{iPhone}$

Step 2: Generate Topic-Post Matrix for each relevant node from the given input of: Node V , Topic TT , GraphAPI, likes and comments on posts. Threshold Approve (A) for nodes with likes > 1000 .

Input node $v = 130422069$ and crawl profile page to search relevant posts on topic z to get table 3 data.

Output: Count of likes (A), comments+shares (SR).

Example, post id=“46921” posted by node “130422069” and has 61153 likes on the post, and $SR = 11325$.

Post id	Term	A	SR
46921	iPhone5-The biggest thing to happen to iPhone since iPhone :	61153	11325
180356	Amazing iPhone!	20147	1880

Table 3: User Post Data

Steps of the Proposed ACOMax System

Input: Social network URL (e.g., Twitter), and product p (e.g., iPhone12), predefined threshold (post with minimum number of likes l , Retweet Rt , and comments count CC)

Output: Final seed set $S = \{\text{consisting of users with most favorable opinions on product } p_t\}$

Step 1: Collection of Twitter Data is done through Twitter API to collect **time-based** relevant posts and nodes for product p , i.e., iPhone12. Time-based targeted marketing, i.e., single product (iPhone) and multiple posts of it, so we are not considering multiple products on multiple topics like OBIN.

Output: Generate profile matrix PM and store user data if Quotes ≥ 50 , Retweets > 50 and Comments > 100 .

User Id	Date	Qt	Rt	CC	Content
1	2021-03-02	130	100	50	The camera quality is nicely done.
2	2021-04-03	50	90	80	I like the camera on my new iPhone 12 .Great for selfies!

Table 1: Profile matrix for $p = \text{iPhone12}$

Step 2: Reconstruct profile matrix for each relevant node by retaining comments in English and filter out other and considering **time-based** user tweets i.e., comments on tweets within 1 year.

Input node $v=1$ and extract tweet information from the post to search relevant posts on product p to get table 2 data.

Output: Count of comments + retweets + mentions (A).

User Id	$A = (Rt + CC + Qt)$	Comment C
1	190	The camera quality is nicely done.
2	110	I like the camera on my new iPhone 12 .Great for selfies!

Table 2: Profile matrix for $p = \text{iPhone12}$

Example, filter out nodes with friends less than threshold and tweets other than English language and reconstruct profile matrix by only taking

Step 3: Input: (term + A), (term + SR), (A+ SR) to classify relevant and irrelevant nodes using Linear Support Vector Machine (SVM).

For example, Term={iPhone, Apple}, A>100, and SR>20 extracts posts on the topic z = iPhone. Then, store the nodes, posts, and corresponding users' comments.

Cat Id	TpId	P_id	Cmtid	Pol	Comment
1	2	962538	693210	NULL	i want to have screen like this iPhone
1	2	962538	405279	NULL	iPhone 4 was much better than this.

Table 4: Comments Table

Step 4: Next from Table 4, PCP-Miner takes comments for each post on topic z and perform data preprocessing. For each comment (c), PCP-Miner algorithm performs following steps: For example, first row from table 4:

$C = "i\ want\ to\ have\ screen\ like\ this\ iPhone"$

4.1. Tokenization & POS-Tagging.

POS = {(‘i’, ‘NN’), (‘want’, ‘VBP’), (‘to’, ‘TO’), (‘have’, ‘VB’), (‘screen’, ‘NN’), (‘like’, ‘IN’), (‘this’, ‘DT’), (‘iPhone’, ‘NN’)}

4.2. Feature Extraction using Apriori algorithm to get a set of features in POS={screen_NN, iPhone_NN}.

4.3. Association Rule() algorithm to identify the related feature for {screen}, i.e., FFT = {iPhone}.

4.4. Then extract opinion-words from the POS table, i.e., OE = {want}

4.5. Identify the Semantic Orientation. **Example,** SO = {want, positive}.

relevant user_id, comments and Approve(A) for the next step.

Step 3: Input profile matrix (Table 2) to clean user comments C using first step of Joint Opinion Miner algorithm which performs following steps in preprocessing and generate Table 3 as a result. Example, second row from Table 2 as an input.

3.1 Tokenization. Output: ‘The’, ‘camera’, ‘quality’, ‘is’, ‘nicely’, ‘done’, ‘.’.

3.2 Preprocessing - Removes foreign characters, URLs, RT (Retweet). Data Cleaning to remove symbols and stop words from the text using spaCy STOPLIST.

3.3 POS-Tagging to assign adjectives to the words i.e., (‘camera’, ‘NN’), (‘quality’, ‘NN’), (‘nicely’, ‘RB’), (‘done’, ‘VBN’)]

3.4 Lemmatization to convert words to their first form, i.e., {camera, quality, nice, do}.

3.5 Slang words are replaced into their original words using SlangSD lexicon library. For example, {awesm = awesome}.

3.6 Replacing opinion displayed using Emoji with the text using demoji library. For example, {❤ = love}.

User Id	A	Comment C
1	190	Camera quality nice do
2	110	like camera new love great selfies

Table 3: Profile matrix for p = iPhone12

Step 4: Take Table 3 as an input and obtain the Subjective (which offer opinion and not just facts about the product) posts, modality detection using Sentiment VADER and assign overall polarity to sentences.

UserID	PostID	Pol	Time	Comment
100002	962538	Pos	2013-01-06	i want
100003	962538	Pos	2013-01-06	This is cool

Table 5: Example data in Facebook Comment table

Step 4: Compute polarity: Input table 5 for calculating.

$$\theta_z = (\text{pos} - \text{neg}).$$

For example, Table 6 shows the popularity matrix for post $w = 962538$, where $\theta_z = (5 - 0) = 5$. (Positive) > (Neutral + Negative) i.e., $5 > (2 + 0)$.

Post id w	A	SR	θ_z	Term
962538	20147	1880	5	This is cool

Table 6: Polarity Matrix

Step 5: Finally, create Influence network graph using *PoPGen* (popularity graph generator).

Input: Relevant nodes, posts w , comments c , nodes commented on the posts.

Output: Generates a social influence graph $G_z = (V, E)$ on topic z using the influence matrix.

5.1: Calculate number of times node u responded to topic-posts by node v .

Influence score = number of responses by u to v .

5.2: Filter out nodes that have a lower influence score than a predefined threshold.

Example, first row in table 7, node ‘1033467’ posted a post and node ‘33889’ gave its opinion. So, node = ‘1033467’ has an influence on node = ‘33889’.

Node id vs	Post id w	Node id vt
1033467 (1)	49823667	33889 (4)
1033467 (1)	49823667	458089 (5)
1033467 (1)	49823667	221458 (6)
1033467 (1)	55090883	1120347 (7)

Table 7: Post-user relationship

User Id	A	Comment C	Pol
1	190	Camera quality nice do	Pos
2	110	like camera new love great selfies	Pos

Table 4: Sentence Polarity Score

Step 5: Feature Extraction using Apriori algorithm to get a set of features in $\text{POS} = \{\text{screen_NN}\}$.

User Id	A	Comment C	Features
1	190	Camera quality nice do	Camera
2	110	like camera new love great selfies	Camera, selfies

Table 5: Sentence Polarity Score

5.1. Then extract opinion-words from the table 5, i.e., $\text{OE} = \{\text{nice}\}$ for first row.

5.2. Identify the Semantic Orientation of OE.

Example, $\text{SO} = \{\text{nice, positive}\}$.

User Id	A	Comment C	Pol
1	190	Camera quality nice	Pos
2	110	like camera love great selfies	Pos

Table 6: Sentence Polarity Score

Step 6: Compute polarity. $\text{SO} = \{\text{nice, positive}\}$ and then assign scores in the range $[-4, 4]$ to each of the opinion words. For example, “love”=strong positive opinion, “nice”= positive but not as strong.

User Id	A	Comment C	Pol	Feature Score
1	190	Camera quality nice	Pos	2
2	110	like camera love great selfies	Pos	4

Table 7: Feature Polarity Score

Step 7: We use opinion score to create user-item rating matrix (Table 8) consisting of top-k users on selected products by the seller.

Node id $Vt1$	Node id $Vt2$
33889 (4)	221458 (6)
221458 (6)	114509 (8)
114509 (6)	447880 (9)

Table 8: user-user relationship

First row in table 8, node = '33889' is a friend of node = '221458'. So, selecting node = '1033467', influence spreads through '33889' to node = '221458'.

Step 6: *PoPGen* find if u has a relation with v . *PoPGen* add a node to the vertex list. For example, in table 9, node = 1 has relation with node 3, 4, 5, 6, and 7.

	1	2	3	4	5
1	0	0	1	1	1
2	0	0	0	0	0
3	1	0	0	0	1
4	1	0	0	0	0
5	1	0	1	0	0

Table 9: Influence Matrix IMAT

Next, draw a graph by adding an edge between nodes if value is 1. Example, node 1 is connected to node 3, 4, and 5.

Step 7: OBIN calculates number of times a node responds to all the topic-posts posted by node v .

Influence score (IP) = number of responses by v to u

User Id	Compound Score
1	0.98
2	0.97
3	0.90

Table 8: user-item relationship

Step 8: Finally, create opinion network graph G from positive users and retrieve users' friends using Twitter API that returns friend IDs of a specified user. Each user id and friend id pair is inserted into User_friend table. We collect tweets of top 100 users with most followers.

User id	Friend_id
1	2, 3
2	3, 5
3	4
5	4
6	5

Table 9: user-user relationship

For Example, first row in table 9, node '1' posted a post and node '4' gave its opinion.

Step 9: Input users as nodes with Opinion Score = 'Positive', Approval (quote, comments, retweets).

Assign influence probability among nodes to edges:

$$P_{u,v} = (\beta_1 + \beta_2 + \beta_3)$$

$$\frac{(\#retweets) + (\#replies) + (\#quotes)}{\#tweets\ of\ v}$$

Retrieve friends of top 100 users from Twitter follower network. Generate community opinion social network graph $G(V, E)$, where add nodes V from UserID and edges E from FriendID (Table 7).

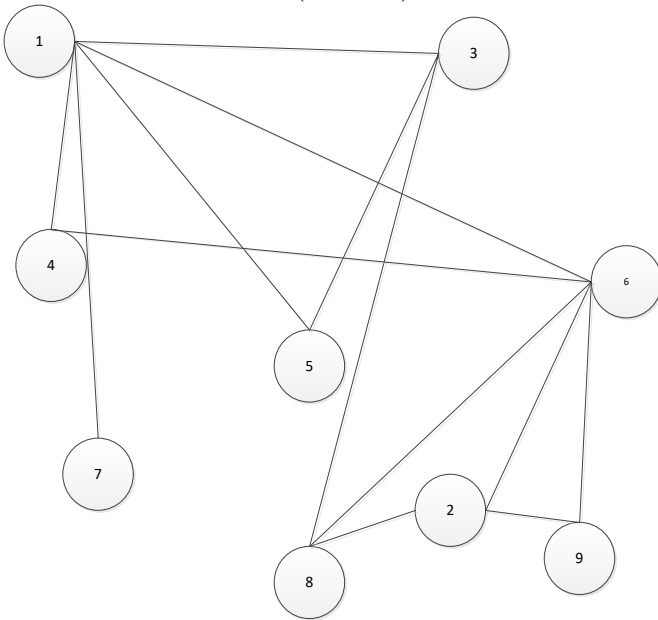
Output: Generates a community opinion network social graph $G = (V, E)$.

Nodes	IP
1	120
2	118
3	116
4	115
5	108

Table 10: Influence Matrix IMAT

Filter out nodes that have a lower influence score than a predefined threshold (fixed number of high scored nodes greater than 100).

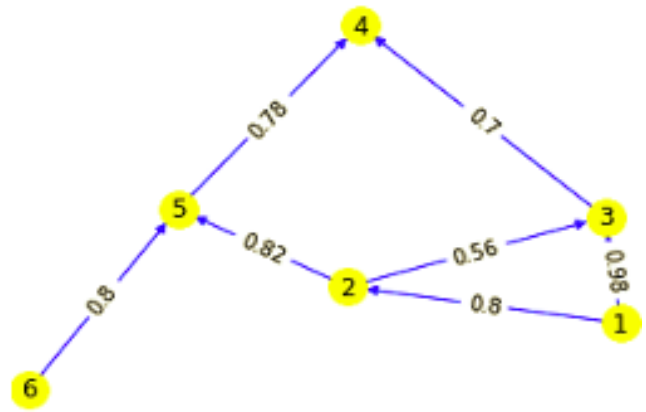
Step 8: Construct influence graph generated from the influence matrix IMAT (Table 9)



User id	Friend_id	IP
1	2	0.80
1	3	0.98
2	3	0.56
2	5	0.82
3	4	0.70
5	4	0.78
6	5	0.80

Table 10: Community influence probability matrix

Step 10: Step 7 and 8 combined will result in community opinion network graph $G = (V, E)$



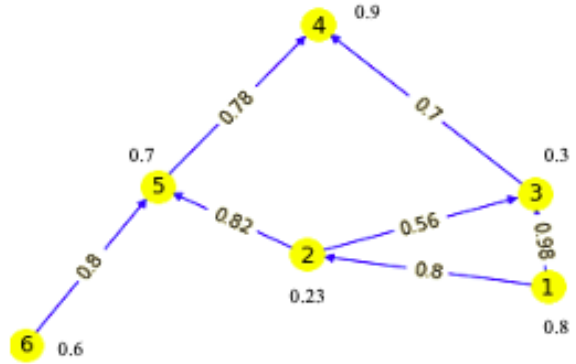
Step 11: Finally, input G from step 10, MLT under Greedy Algorithm to activate k -best seed set users ($S(1), \dots, S(T)$).

Input: Social network graph $G = (V, E)$ from step 7, opinion score R , budget $k=50$, $P_{u,v}$, threshold $\theta = \text{random } [0, 1]$.

Step 11.1: We use Greedy algorithm under MLT to select $k^{(q)}$ users that maximize the total opinion spread by activated users.

Step 11.2: The spread of influence starts from the node $\{1\}$ which results in activation of nodes $\{2, 3, 5, 4\}$.

E.g., $\{1\} = \{\underline{2}, \underline{3}\}$, $\{2\} = \{\underline{5}\}$, $\{5\} = \text{fail to } \underline{4}\}$, $\{5, 3\} = \{\underline{4}\}$.



Step 11.3. Add user to the seed set S , if user has highest opinion spread in each round.

Step 11.4. Scan all inactive users until budget lasts.

Final Output: Seed user set $S = \{(S^{(1)}, \dots, S^{(T)})\}$

CHAPTER 4: EXPERIMENTAL EVALUATION AND ANALYSIS

In this section, we present various experiments to evaluate the efficiency and effectiveness of the proposed approach.

4.1 Dataset Selection

We will conduct our experiments using the users' posts and opinions of Twitter as a friendship network since it is currently the most popular social media website. In this thesis, we perform our experiments on Twitter real-world data set. We extracted data for iPhone12. Our proposed SPOM method automatically extracts the Twitter follower network, Twitter mention network, Twitter reply network, and Twitter retweet network through TwitterAPI and stores the data into data frameworks.

4.1.1 Twitter- API: To collect Twitter data, I have used premium API provided by Twitter for Academic research. There are two different APIs to collect Twitter data.

- (i) The Representational State Transfer (REST) API provides information about individual user accounts or popular topics and allows for sending or liking Tweets and following accounts.
- (ii) The Streaming APIs are used for real-time collection of Tweets and come in two flavors:
 - a. First, the Filter API extracts Tweets based upon a user's query containing keywords, user accounts, or geographic areas.
 - b. The Filter API is used for studying Twitter content found on a predefined set of topics, user accounts, or locations.

For our research, we have used the Streaming Filter API used on a predefined set of product and query parameters such as date of tweet, language, end date.

4.1.1.1 Data Acquisition

In this thesis, we used 239,044 tweets for a single product as our text corpus. We obtained English tweets from Twitter throughout the month. (August 2020 – July 2021). We have used twitter streaming because streaming allows you to actively watch for tweets that match certain criteria in real time. This means that when there aren't any new tweet matching the criteria, then the

program will wait until a new tweet is created and then process it. To use streaming you must create two objects: 1. The stream object uses the Twitter API to get tweets that match some criteria. This object is the source of tweets that are then processed by a stream listener. 2. The stream listener receives tweets from the stream. First, we extracted 239,044 tweets and many tweets are rejected which does not match the criteria in table 20. Out of 239,044 we get 6347 tweets (2261 positive, 1562 negative, 1093 neutral and 1431 objective)

Keyword	iPhone12
Type	Original tweet and not (-is:retweet + -is:reply + -is:nullcast)
Language	English
Start date	01-08-2020
End data	30-07-2021
Approve	Count of (Likes, comments, retweets > 100)

Second, Twitter Follower data (UserID, friendID) for constructing graph G

# Nodes	# Edges	Linkage
106	300	Directed

We first filter out objective reviews (1431) using TextBlob, then we perform frequent feature mining of selected subjective reviews with support = 0.045. Next, we calculate compound score of features containing reviews using VADER, after the execution of models on different parameter for selection of users with most favourable (positive) opinion about the product p = iPhone12.

4.2 Evaluation Measures

We evaluate our proposed ACOMax opinion mining process with OBIN from three performance matrices and total opinion spread with CONE:

1. Recall: We calculate recall to identify the proportion of actual positives are identified correctly. Recall is the ratio of the number of users/nodes activated to the total number of nodes in constructed community opinion network graph.

$$R = \frac{A}{A+B} \times 100\%$$

Equation 10: Recall value

Where in equation 10, A = number of active nodes, B = number of inactive nodes.

2. Precision: We calculate precision to identify the proportion of positive identifications of users as actual positives. Precision is the ratio of the number of active users/nodes retrieved to the total number of active and inactive nodes. $P = \frac{A}{A+C} \times 100\%$ (C = number of irrelevant nodes retrieved).

Equation 11: Precision value

3. F1 Score: F1-score might be a better measure to use to seek a balance between Precision and Recall.

$$F1\text{-score} = 2 \cdot \frac{P \times R}{P + R} \quad \text{Equation 12: F1-score}$$

4.2.1 Performance Analysis: In MLT diffusion model, a user can influence his neighbors with specific influence weights and has a threshold, which denotes the minimal required influence to be activated by the neighbors. The weights of the directed social links quantify the influence propagates from u to v . We quantify the weight $P_{u,v} = (\beta_1 + \beta_2 + \beta_3)$ i.e., (number of retweets + replies + mentions of u on v). We store users with positive opinions and opinion ranking above 90% and construct community network graph from it for opinion maximization. The total budget of seed users are all set as 50, we let $k^{(1)} = k^{(2)} = k^{(n)}$ where k is the number of users to be selected in each round. We obtain the total opinion of the final set of activated users by repeating the algorithm 50 times and report the average total opinion spread.

4.2.2 Statistical Analysis: We calculate confidence interval (C.I) to measure the reliability of our system (Levine, 2010). A confidence interval is a bound on the estimate of a population variable. It is an interval statistic used to quantify the uncertainty on an estimate. The 99% confidence interval (CI) is a range of values calculated from our data, that most likely, includes the true value of what we're estimating about the population. For example, in table 29, we get a total of 100 influential users, the 99% C.I based on Approve (sum of number of likes, retweets, replies and mentions of u on v). Average of each node is between (96.85112732155449, 484.1111368293889).

Top Nodes	Opinions	OBIN	ACOMax
		Average Bound	Average Bound

100	Approve	441.75	484.11
	Influence score	<u>65.61</u>	<u>96.85</u>
200	Approve	243.95	340.67
	Influence score	58.87	<u>88.34</u>

Table 29 Comparison of 99% CI for different number of discovered influential nodes

Table 30 shows the accuracy measure of OBIN and CONE and proposed ACOMax. We can see that the recall value of ACOMax is 97.89%, this is because OBIN does not extract 106 relevant nodes which are subjective in nature because it selects user opinions based on feature words in the reviews because OBIN fails to consider reviews which have slangs, emoji and modalities. Whereas, ACOMax can extract most of the relevant nodes because we convert reviews in a proper English word which makes it easy for algorithm to classify text as positive, negative, neutral or objective in nature. ACOMax could not extract some nodes due to the criteria of less approval rate (number of likes, shares and retweets).

	Precision	Recall	F1-score
OBIN	98.24%	93.71%	95.3%
ACOMax	98.50%	97.89%	98.19%

Table 30 Comparison of discovering influential nodes by OBIN, CONE and ACOMax

Precision is 98.50%, this is because it predicts actual positive users, and we filter out objective texts opinions which makes it easy to distinguish between polarity of text.. We measure the relevant score by measuring the number of positive nodes (influencing) extracted. As we can see, with small number of nodes, OBIN and ACOMax give better performance in Precision, but as we increase the number of nodes, ACOMax performs better because recall value will decrease for OBIN as it will fail to extract relevant nodes again. Hence, combining subjectivity detection with sentence preprocessing (slangs, emoji and modalities) will increase the overall F1-score to extract nodes with the most favourable opinion about the product for construction of opinion network graph.

Figure 25 shows the influence nodes based on extracting relevant score of nodes and how the influence spread achieved by our ACOMax algorithm improves the influence spreads. Following (Figure 25) is a graph of Tweets vs Influence Rank which depicts positive as well as negative influencers. Both the parameters of the graph help to identify the top-level trend setters. We measure the opinion spread by measuring the number of nodes activated by the influential nodes extracted.

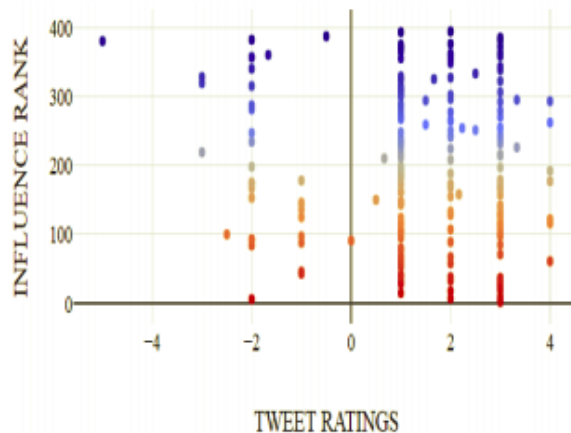


Figure 25: Tweets vs Influence Rank

4.2.3 Total Opinion Spread: Figure 26 shows the opinion spread over the network constructed from Twitter for top 100 users for CONE and proposed ACOMax. We measure the opinion spread by measuring the number of nodes activated by the influential nodes mined. As we can see in figure 26, with small number of seed set ($k=10, 20$ and 50), ACOMax performs better in opinion spread because for a specific product, CONE first predicts/estimates opinion of a user and must start with a huge dataset to select initial set of seed users to start with also known as cold start problem, this will also be visible with computation time of CONE. ACOMax has initial set of active (positive) users to begin with and discovers nodes whose opinion is higher which increases the performance and total opinion spread. Also, ACOMax has a high total opinion spread is because we start with users with actual positive opinion about the target product instead of estimating the opinion which is never close to real opinion/preference of the user. Given the same dataset, ACOMax and CONE will have some common seed but CONE will take a lot of time to retrieve those users because CONE first must find users with positive opinions before spreading

the influence in the meantime ACOMax will already add such users to the seed set, hence saving time and cost for the seller.

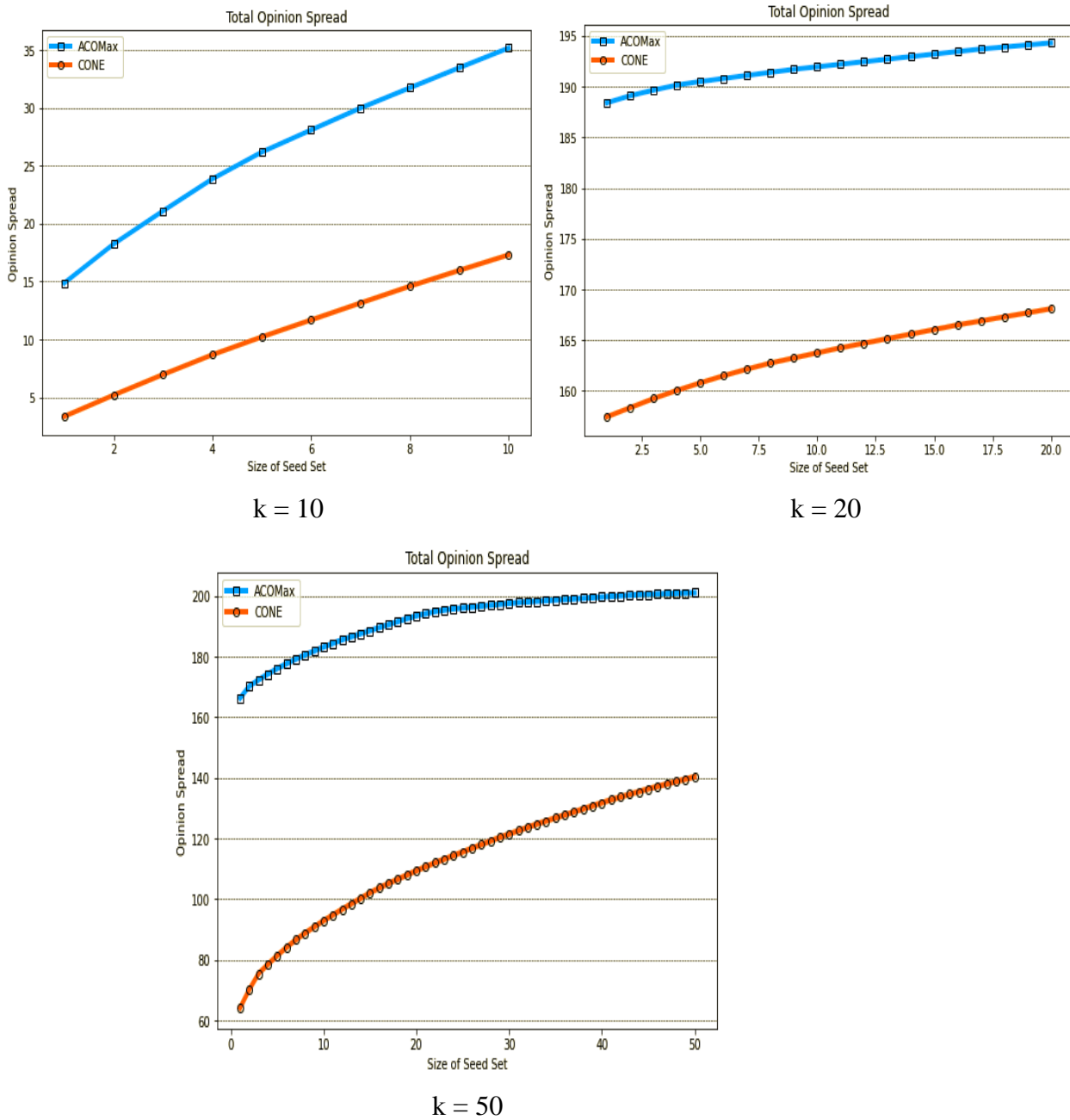


Figure 26 Comparison of opinion spread under opinion maximization (CONE vs ACOMax)

We can see in the above figure, as the budget increase opinion spread for ACOMax increases to more nodes displayed as opinion spread count in figure 26. So, comparison of CONE and ACOMax to be better is based on the total opinion spread in less time to save time and cost.

4.3 Complexity Analysis

We measure and compare the complexity analysis for Opinion Maximization and community opinion network graph generation:

1. ACOMax, for community opinion network graph: $O(n \log m)$, n = number of nodes, m = number of edges. For each user in G , we compute user-user relationship and assign linear threshold score. For, OM, $O((n^2 \log m)k)$, where k = budget (number of active nodes as early adopters), n =number of nodes (106), m =number of edges (300). E.g., for each user-user relation in the dataset ($O(n^2)$) we use the binary search ($O(\log m)$) to search the same value in dataset.

1.1. The first phase is constructing community opinion network graph $G = (V, E)$ which runs in time $O(n \log m)$. For each user in G , we compute user-user relationship and calculate influence probability score through likes, retweets, comments and mentions. We scan the out-neighbors of each node and the total number of out-neighbors of all nodes is $O(n)$.

1.2. Greedy algorithm which runs in time $O((n^2 \log m)k)$, where k is the budget, i.e., the number of A-nodes to be discovered as early adopters of products p , n is the number of nodes, m is the number of edges in G . ($O(n^2)$) is because we have to iterate from user-user for activating them and edges will reduce in every scan as we activate users and add to seed set.

ACOMax	CONE	OBIN
$O((n \log m) + (k (n^2 \log m)))$	$O(n^2 + (n + m) k^3)$	$O((a * n) + m^2)$

Table 31: Complexity Analysis

2. CONE performs opinion estimation and maximization. CONE must perform estimation and maximization. So, best choice of k does not change much while m or n increases as it must search through whole network for positive users while estimating opinions in every round till budget lasts. The time complexity of CONE is roughly $O(m^2 + (m + V) k^3)$ with is very high as visible in figure 27 reason being CONE must perform opinion-estimation (positive, negative or neutral) of users from their historical data and select users with positive opinion in every round which will result in the best choice of k not changing much while m or n increases as it must search through whole network for positive users while estimating opinions in every round till budget lasts.

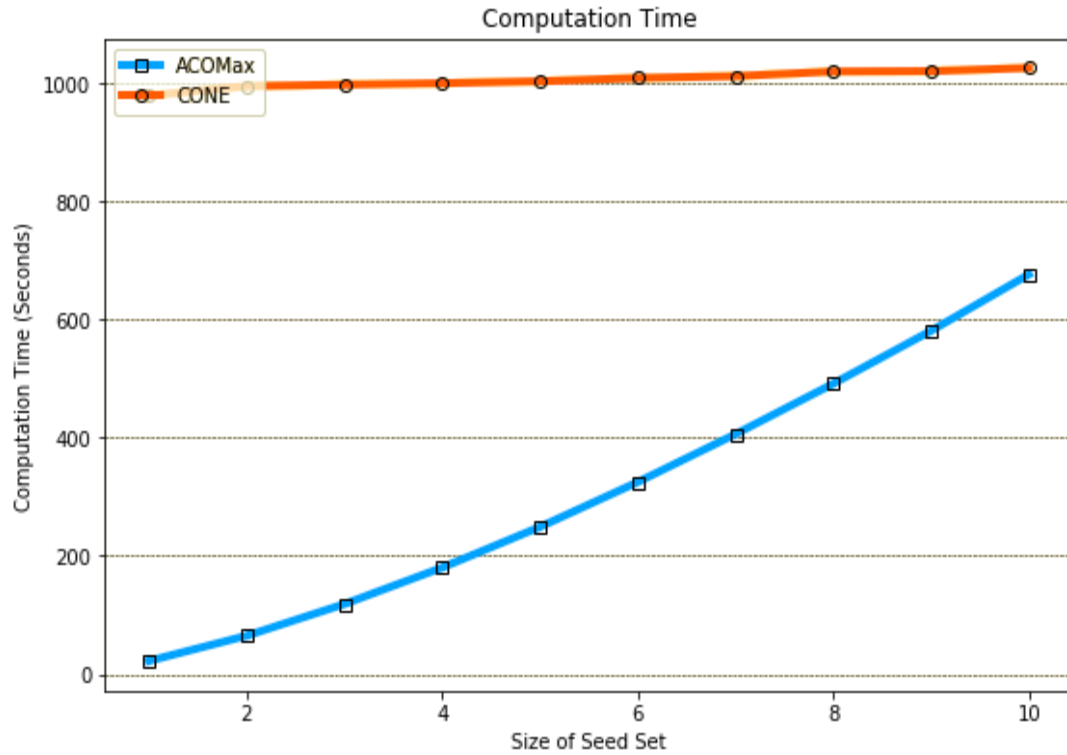


Figure 27 Time complexity (CONE vs ACOMax)

3. OBIN computed complexity for popularity score of posts (a) and for influence graph.

4.4 Implementation and Coding

- ❖ IDE (PyCharm for algorithm, Anaconda for visualizing results and Eclipse).
- ❖ Java as programming language (For Apriori Frequent Feature Opinion Mining)
- ❖ Python Programming Language:
 - Twitter API for crawling the data from Twitter.
 - spaCy and NLTK for tokenization, stopwords removal and POS tagging
 - NLTK for VADER and TextBlob for subjectivity classification.
 - WordNet and NLTK corpus for semantic orientation of opinions on features.
 - Networkx and igraph for Graphs and Visualization.

CHAPTER 5: CONCLUSION AND FUTURE WORK

In this thesis, we proposed a joint approach for opinion mining, which generates the frequent features related to a product. As input, ACOMax takes in raw unprocessed tweets and first classifies the tweets at the sentence level to determine whether they express any opinion or not and pass the filtered positive and neutral opinions for feature opinion mining and calculate feature opinion polarity using WordNet and store top 106 users with positive opinion to construct a community opinion network graph and assign influence probabilities using real opinions expressed through likes, retweets, comment count and quotes on original post about the product p (i.e., Apple iPhone12). We were able to clean the data required that can be used for sentiment analysis. In this research, we first time propose the concept of active learning framework of opinion mining and opinion maximization under the Multi Linear Threshold (MLT) model. We perform experiments on the real-time data from Twitter to show our proposed ACOMax framework outperforms existing algorithm such as OBIN and CONE. Community opinion network generation process in ACOMax is an extended version of OBIN where we added real time influence probability instead of assigning weightage to edges based on how many times user visited the profile of other users. To conclude, we select top 106 users with highest number of followers, overall opinion score of to construct community opinion network graph G to further selecting top- k users with highest opinion spread (influence) over the network as early adopters of the product.

However, as the network grows dramatically and parameters also grow, our system can slow down due to execution time in large-scale network. As, ACOMax combines opinion mining and maximization for influence spread of a selected product which helps in creating the profile of the product and mining the real-time data from social networking sites. In future work, (1) combining the two subtasks (opinion mining and maximization) in a more robust and efficient manner to target large set of users to create a profile of multiple products and targeting different countries. (2) consider dynamic networks where new nodes come in, existing nodes leave, or the influence probability per edge changes as time goes on (i.e., it is not independent to time anymore).

REFERENCES

1. Liu, X., Kong, X., & Yu, P. S. (2018). Active Opinion Maximization in Social Networks. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD 18, 1840-1849.
2. Gionis, A., Terzi, E., & Tsaparas, P. (2013). Opinion maximization in social networks. Proceedings of the 2013 SIAM International Conference on Data Mining.
3. Ahmed, S., Ezeife, C.I. (2013). Discovering Influential Nodes from Trust Network. In: ACM SAC International Conference, Coimbra, Portugal.
4. Bonchi, F., Castillo, C., Gionis, A., Jaimes, A. (2011). Social network analysis and mining for business applications. ACM Transaction TIST 22.
5. H. Zhang, T. Dinh, and M. Thai. 2013. Maximizing the spread of positive influence in online social networks. In ICDCS'13. 317–326.
6. Kempe, D., Kleinberg, J., and Tardos, E., 2003. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '03. ACM, New York, NY, USA, 137–146.
7. Mumu T.S., Ezeife C.I. (2014) Discovering Community Preference Influence Network by Social Network Opinion Posts Mining. Data Warehousing and Knowledge Discovery. DaWaK 2014.
8. Turney, P. D. 2002. Thumbs up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), 417-424.
9. Dave, K., Lawrence, S., and Pennock, D. M. 2003. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In Proceedings of the 12th international conference on World Wide Web (WWW'03), 519-528.
10. Hu, M. and Liu, B. 2004. Mining and Summarizing Customer Reviews. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), 168-177 .
11. Chen, W., Wang, Y., & Yang, S. (2009). Efficient influence maximization in social networks. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 09.

12. Lv, J., Guo, J., & Ren, H. (2014). Efficient Greedy Algorithms for Influence Maximization in Social Networks. *Journal of Information Processing Systems*, 10(3), 471–482.
13. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Vanbriesen, J., & Glance, N. (2007). Cost-effective outbreak detection in networks. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 07*.
14. Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1), 265–294.
15. Sun, L., Huang, W., Yu, P. S., & Chen, W. (2018). Multi-Round Influence Maximization. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD 18*.
16. Gunjan Soni & CI Ezeife. An automatic email management approach using data mining techniques. In *Data Warehousing and Knowledge Discovery*, pages 260-267. Springer, 2013.
17. James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistic and probability*, volume 1, pages 281-297. Oakland, CA, USA., 1967.
18. Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21-27, 1967.
19. Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs, NJ: Prentice Hall.
20. Yi-Cheng Chen, Wen-Yuan Zhu, Wen-Chih Peng, Wang-Chien Lee, and Suh-Yin Lee. Cim: community-based influence maximization in social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2):25, 2014.
21. Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on for text categorization*, volume 752, pages 41-48. Cite seer, 1998.
22. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273-297, 1995.
23. Moosavi, S. A., Jalali, M., Misaghian, N., Shamshirband, S., & Anisi, M. H. (2016). Community detection in social networks using user frequent pattern mining. *Knowledge and Information Systems*, 51(1), 159-186.
24. J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81-106, 1986

25. Qingbo Hu, Guan Wang, and Philip S Yu. Transferring influence: Supervised learning for efficient influence maximization across networks. In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on, pages 45-54. IEEE, 2014.
26. Bing Liu, Wynne Hsu, Shu Chen, & Yiming Ma. (2000). Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems*, 15(5), 47-55.
27. Parthasarathy S., Ruan Y., Satuluri V. (2011) Community Discovery in Social Networks: Applications, Methods and Emerging Trends. In: Aggarwal C. (eds) *Social Network Data Analytics*. Springer, Boston, MA.
28. Goyal, A., Lu, W., & Lakshmanan, L. V. (2011). CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. *Proceedings of the 20th International Conference Companion on World Wide Web - WWW 11*.
29. W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. 2011. Influence maximization in social networks when negative opinions may emerge and propagate. In *SDM '11*. 379–390.
30. Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume 1 (ETMTNLP '02)*. Association for Computational Linguistics, USA, 63–70.
31. Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Massachusetts: MIT press.
32. Mai, L., Le, B. Joint sentence and aspect-level sentiment analysis of product comments. *Ann Oper Res* 300, 493–513 (2021)
33. Liu Y., Yu X., Liu B., Chen Z. (2014) Sentence-Level Sentiment Analysis in the Presence of Modalities. In: Gelbukh A. (eds) *Computational Linguistics and Intelligent Text Processing. CICLing 2014. Lecture Notes in Computer Science*, vol 8404. Springer, Berlin, Heidelberg.
34. Nguyen, Heidi; Veluchamy, Aravind; Diop, Mamadou; and Iqbal, Rashed (2018) "Comparative Study of Sentiment Analysis with Product Reviews Using Machine Learning and Lexicon-Based Approaches," *SMU Data Science Review: Vol. 1 : No. 4 , Article 7*.
35. Mukherjee, Subhabrata & Bhattacharyya, Pushpak. (2013). *Sentiment Analysis : A Literature Survey*.

36. Sentiment Analysis and Subjectivity Bing Liu. (2010). Handbook of Natural Language Processing, 651–690.
37. Mei, Y., Zhao, W., & Yang, J. (2017). Influence maximization on twitter: A mechanism for effective marketing campaign. 2017 IEEE International Conference on Communications (ICC).
38. Jawad Khan, Byeong Soo Jeong, Young-Koo Lee, and Aftab Alam. 2016. Sentiment analysis at sentence level for heterogeneous datasets. In Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory (EDB '16). Association for Computing Machinery, New York, NY, USA, 159–163
39. Eirinaki, M., Pisal, S., & Singh, J., Feature-based opinion mining and ranking, Journal of Computer and System Sciences, Volume 78, Issue 4, 2012, Pages 1175-1184.
40. Hutto, C., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of social media Text. ICWSM.
41. Wu, L., Morstatter, F. & Liu, H. SlangSD: building, expanding, and using a sentiment dictionary of slang words for short-text sentiment classification. Lang Resources & Evaluation 52, 839–852 (2018).
42. Al-Shabi, M. (2020). Evaluating the performance of the most important Lexicons used to Sentiment analysis and opinions Mining. IJCSNS International Journal of Computer Science and Network Security, VOL.20 No.1, January 2020.
43. Alsaeedi, A., & Zubair, M. (2019). A Study on Sentiment Analysis Techniques of Twitter Data. International Journal of Advanced Computer Science and Applications, 10(2), 361-374.
44. Bird, S., Klein, E., & Loper, E. (2009). NLTK: The Natural Language Toolkit. O'Reilly Media, Inc.
45. Giachanou, A., & Crestani, F. (2016). Like It or Not: A Survey of Twitter Sentiment Analysis Methods. ACM Computing Surveys, 49(2), 1-41.
46. Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., & Smith, N. A. (2010). Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments, 42–47
47. Han, J., Kamber, M., & Pei, J. (2012). Data mining: Concepts and techniques. Amsterdam: Morgan Kaufmann.

48. Hemmatian, F., & Sohrabi, M. K. (2017). A survey on classification techniques for opinion mining and sentiment analysis. *Artificial Intelligence Review*, 52(3), 1495-1545.
49. Jurafsky, D., & Martin, J. H. (2014). *Speech and language processing*. Harlow: Pearson Prentice Hall.
50. A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, FastText.zip: Compressing text classification models
51. Kaplan, A. M., & Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of social media. *Business Horizons*, 53(1), 59-68.
52. Goyal, A., Lu, W., & Lakshmanan, L. V. (2011). SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model. 2011 IEEE 11th International Conference on Data Mining.
53. Goyal, A., Bonchi, F., & Lakshmanan, L. V. (2008). Discovering leaders from community actions. *Proceeding of the 17th ACM Conference on Information and Knowledge Mining - CIKM 08*.
54. Cai, J. L. Z., Yan, M., & Li, Y. (2016). Using crowdsourced data in location-based social networks to explore influence maximization. *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*.
55. He, Q., Wang, X., Huang, M., Lv, J., & Ma, L. (2018). Heuristics-based influence maximization for opinion formation in social networks. *Applied Soft Computing*, 66, 360–369.
56. Liang, W., Shen, C., Li, X., Nishide, R., Piumarta, I., & Takada, H. (2019). Influence Maximization in Signed Social Networks with Opinion Formation. *IEEE Access*, 7, 68837–68852.
57. Nayak, A., Hosseinalipour, S., & Dai, H. (2019). Smart Information Spreading for Opinion Maximization in Social Networks. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*.
58. Rossi, M.-E. G., Shi, B., Tziortziotis, N., Malliaros, F. D., Giatsidis, C., & Vazirgiannis, M. (2018). MATI: An efficient algorithm for influence maximization in social networks. *Plus, One*, 13(11).

59. Song, C., Hsu, W., & Lee, M. L. (2016). Targeted Influence Maximization in Social Networks. Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM 16.
60. Yu, P., Hu, Q., & Wang, G. (2014). Transferring Influence: Supervised Learning for Efficient Influence Maximization across Networks. Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing.
61. Wu, X., Fu, L., Meng, J., & Wang, X. (2019). Maximizing Influence Diffusion over Evolving Social Networks. Proceedings of the Fourth International Workshop on Social Sensing (SocialSense'19). ACM, New York, USA, Pages 6-11
62. Chen, H., Loukides, G., Fan, J., & Chan, H. (2019). Limiting the Influence on Vulnerable Users in Social Networks: A Ratio Perspective. Advanced Information Networking and Applications Advances in Intelligent Systems and Computing, 1106–1122.
63. Banerjee, S., Jenamani, M., & Pratihari, D. K. (2019). Maximizing the Earned Benefit in an Incentivized Social Networking Environment. Proceedings of the ACM India Joint International Conference on Data Science and Management of Data - CoDS-COMAD 19.
64. Tang, W., Luo, G., Wu, Y., Tian, L., Zheng, X., & Cai, Z. (2019). A Second-Order Diffusion Model for Influence Maximization in Social Networks. IEEE Transactions on Computational Social Systems, 6(4), 702–714.
65. Ju, W., Chen, L., Li, B., Liu, W., Sheng, J., & Wang, Y. (2019). A new algorithm for positive influence maximization in signed networks. Information Sciences.
66. He, Q., Wang, X., Yi, B., Mao, F., Cai, Y., & Huang, M. (2019). Opinion Maximization Through Unknown Influence Power in Social Networks Under Weighted Voter Model. IEEE Systems Journal, 1–12.
67. Wang, Y., Cong, G., Song, G., & Xie, K. (2010). Community-based greedy algorithm for mining top-K influential nodes in mobile social networks. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 10.
68. Lek, H. H., & Poo, D. C. C. (2013). Aspect-Based Twitter Sentiment Classification. 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, 366–373.
69. Liu, B. (2012). Sentiment analysis and opinion mining. Morgan & Claypool Publishers.

70. Pandarachalil, R., Sendhilkumar, S., & Mahalakshmi, G. S. (2015). Twitter Sentiment Analysis for Large-Scale Data: An Unsupervised Approach. *Cognitive Computation*, 7(2), 254-262.
71. Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, 271-278.
72. Perkins, J. (2010). *Python text processing with Nltk 2.0 cookbook*. Packt Publishing Ltd.
73. S. Pisal, J. Singh and M. Eirinaki, "AskUs: An Opinion Search Engine," 2011 IEEE 11th International Conference on Data Mining Workshops, 2011, pp. 1243-1246.
74. Khan, F. H., Qamar, U., & Bashir, S. (2016). SentiMI: Introducing point-wise mutual information with SentiWordNet to improve sentiment polarity detection. *Applied Soft Computing*, 39, 140–153.
75. K.Jawadwala, M., & Kolkur, S. (2014). Feature Ranking in Sentiment Analysis. *International Journal of Computer Applications*, 94(13), 42–49.
76. Nguyen, H., Veluchamy, A., Diop, M., & Iqbal, R. (2018). Comparative Study of Sentiment Analysis with Product Reviews Using Machine Learning and Lexicon-Based Approaches.
77. Toivonen H. (2011) Apriori Algorithm. In: Sammut C., Webb G.I. (eds) *Encyclopedia of Machine Learning*. Springer, Boston, MA
78. Ravi, K., & Ravi, V. (2015). A survey on opinion mining and sentiment analysis: Tasks, approaches, and applications. *Knowledge-Based Systems*, 89, 14-46.
79. Saif, H., He, Y., & Alani, H. (2012). Semantic Sentiment Analysis of Twitter. *The Semantic Web – ISWC 2012*, 508–524.
80. Zainuddin, N., Selamat, A., & Ibrahim, R. (2017). Hybrid sentiment classification on twitter aspect-based sentiment analysis. *Applied Intelligence*, 1218–1232.
81. Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4), 43–52.
82. Ko, Y.-Y., Chae, D.-K., & Kim, S.-W. (2018). Influence maximisation in social networks: A target-oriented estimation. *Journal of Information Science*, 44(5), 671–682.
83. Liu, Y., Yu, X., Liu, B., & Chen, Z. (2014). Sentence-Level Sentiment Analysis in the Presence of Modalities. *Computational Linguistics and Intelligent Text Processing*, 1–16.

84. Ahmad, T., & Doja, M. (2012). Ranking System for Opinion Mining of Features from Review Documents.
85. Păvăloaia, V.-D., Teodor, E.-M., Fotache, D., & Danileț, M. (2019). Opinion Mining on Social Media Data: Sentiment Analysis of User Preferences. *Sustainability*, 11(16), 4459.
86. Berlingerio, M. (2019). Machine learning and knowledge discovery in databases: European conference, Ecml Pkdd 2018, Dublin, Ireland, September 10–14, 2018: proceedings. Springer.
87. AlBashaireh, R. Z. (2020). In Collecting, mining, and analyzing university-related tweets using sentiment analysis based on machine learning algorithms.
88. H. Soong, N. B. A. Jalil, R. Kumar Ayyasamy and R. Akbar, "The Essential of Sentiment Analysis and Opinion Mining in Social Media : Introduction and Survey of the Recent Approaches and Techniques," 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2019, pp. 272-277.
89. P. Kherwa, A. Sachdeva, D. Mahajan, N. Pande and P. K. Singh, "An approach towards comprehensive sentimental data analysis and opinion mining," 2014 IEEE International Advance Computing Conference (IACC), 2014, pp. 606-612,
90. Huang, S., Shen, D., Feng, W., Zhang, Y., & Baudin, C. (2009). Discovering clues for review quality from author's behaviors on e-commerce sites. Proceedings of the 11th International Conference on Electronic Commerce - ICEC '09.
91. Zhang, Y., & Zhu, W. (2013). Extracting implicit features in online customer reviews for opinion mining. Proceedings of the 22nd International Conference on World Wide Web - WWW '13 Companion.
92. Wang, H., Yuan, Y., Kou, J., Zhang, X., Wang, C., & Duan, J. (2019). Research on Feature Mining Algorithm Based on Product Reviews. 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA).
93. Moosavi, S. A., Jalali, M., Misaghian, N., Shamshirband, S., & Anisi, M. H. (2016). Community detection in social networks using user frequent pattern mining. *Knowledge and Information Systems*, 51(1), 159–186.
94. Tang, L., & Liu, H. (2010). Community detection and mining in social media. Morgan & Claypool.

95. Zhang, J., & Luo, Y. (2017). Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network. Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017).
96. Varpe, A., & Mahajan, M. (2019). Detecting Twitter Compromised Accounts Using Anomalous User Behavior. 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICT).
97. Viswanath, B., Bashir, M., Crovella, M., Guha, S., Gummadi, K., Krishnamurthy, B., & Mislove, A. (2014). Towards Detecting Anomalous User Behavior in Online Social Networks. USENIX Security Symposium.
98. He, Q., Wang, X., Huang, M., Lv, J., & Ma, L. (2018). Heuristics-based influence maximization for opinion formation in social networks. *Applied Soft Computing*, 66, 360–369.
99. Shi, X., Wang, H., Li, J., & Gao, H. (2013). Influence Spread Maximization in Social Network. *International Journal of Information and Education Technology*, 660–666.
100. Moghaddam, S., & Ester, M. (2013). The FLDA model for aspect-based opinion mining. Proceedings of the 22nd International Conference on World Wide Web - WWW '13.
101. Jayamangala, H., & Sheshasaayee, A. (2015). A Review on Models and Algorithms to Achieve Influence Maximization in Social Networks. *Indian Journal of Science and Technology*, 8(29).
102. W. Young Kim, J. Suk Ryu, K. I. Kim and U. Mo Kim, "A Method for Opinion Mining of Product Reviews using Association Rules," in Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, Seoul, 2009.
103. J. Zhang, S. Wang, Q. Zhan, and P. Yu. 2016. Intertwined Viral Marketing in Social Networks. In ASONAM '16.

VITA AUCTORIS

NAME	Mayank Semwal
PLACE OF BIRTH	Dehradun, Uttarakhand, India
YEAR OF BIRTH	1992
EDUCATION	Kendriya Vidyalaya, India (2007 - 2009) Graphic Era (Deemed to be University), Dehradun, Uttarakhand (2009 - 2013) University of Windsor, Ontario, Canada (January 2019 – August 2021)