Influence Maximization

Project Report submitted by

Seedella Sai Vikas (420233)
Krishna Sandeep Vedagiri (420247)
Matta Rahul (420206)

Under the supervision of

Mrs. B.S.S. Monica
M.Tech

Department of Computer Science and Engineering,
National Institute of Technology, Andhra Pradesh

December 2023

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Seedella Sai Vikas (420233)
Krishna Sandeep Vedagiri (420247)
Matta Rahul (420206)

Place: Tadepalligudem
Date: December 11, 2023

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH

Certificate

This is to certify the thesis entitled "INFLUENCE MAXIMIZATION" submitted by ,Seedella Sai Vikas Roll No. 420233, Krishna Sandeep Vedagiri Roll No. 420247, Matta Rahul Roll No. 420206 to National Institute of Technology, Tadepalligudam in partial fulfilment of the requirements for the award of the degree of Bachelors of Technology in Computer Science and Engineering is a record of bonafide research work carried out by them under my supervision and guidance. This work has not been submitted elsewhere for the award of any degree.

Mrs B.S.S.Monica
(Project Guide)

Place: Tadepalligudam
Date: 11/12/2023

ABSTRACT

In this dynamic world of social networks, the pursuit of effective strategies for maximizing influence has become a pivotal research area. This project deals with the optimization of the Greedy algorithm and the subsequent advancements with CELF (Cost-Effective Lazy Forward) and CELF++ for influence maximization. The Greedy algorithm, a fundamental approach for identifying influential nodes, serves as the baseline. However, due to its computational intensity in large networks, the study explores the evolution of CELF, a refined version that strategically prunes the candidate set during execution, thereby mitigating the algorithmic complexity.

Building upon CELF, the project introduces CELF++, an enhanced algorithm that incorporates advanced data structures, such as priority queues, to further elevate computational efficiency. Both CELF and CELF++ aim to address the limitations of the Greedy algorithm, providing cost-effective and scalable solutions for influence maximization in social networks. Through a comprehensive exploration of these algorithms, this research contributes valuable insights into the optimizing influence maximization strategies, offering a nuanced understanding of their applications and implications in the dynamic landscape of social network analysis. The findings contribute not only to the theoretical understanding of influence maximization but also offer practical guidance for leveraging these algorithms in real-world scenarios.

Table of Contents

List of Figures

List of Tables

LIST OF SYMBOLS AND ABBREVIATIONS

CELF	Cost Effective Lazy Forward
CELF++	Cost Effective Lazy Forward++
IC	Independent Cascade
LT	Linear Threshold
DNI	Distant Nodes Influence

Chapter 1

Introduction

1.1 Area of Work

- Social network analysis is a rapidly growing field that seeks to understand and model the complex relationships between individuals in a network.
- One of the key challenges in social network analysis is influence maximization, which involves identifying a small set of individuals in a network who can have the greatest impact on the behavior of the rest of the network.
- This problem has important applications in fields such as marketing, public health, and social media.
- Influence maximization is the process of identifying a set of influential nodes in a social network that can maximize the spread of a particular behavior, idea, or product.
- Finding these nodes is essential for targeted advertising, viral marketing, and the spread of ideas inside a social network.

1.2 Problem Statement

To identify a small set of individual nodes in social network, which can have greatest impact on behavior on rest of the network i.e, which can maximize the spread of the influence.

1.3 Influence Maximization

Due to rapid growth in world population, social media has become an essential platform for companies and organizations to market their product for wide reach and maximizing popularity of the goods and services among the consumers. Social media affect and influence others via the so called word-to-mouth effect, for example, many companies have advertised their products or brands on social networks by launching influence campaigns, giving free products to a few influential individuals (seed nodes), with the hope that they can promote the products to their friends, or few companies even hire a brand ambassador for their marketing they select some celebrity with high popularity to endorse their products, which in turn gets shopped by their fans or admirers.

Social network is a complex network representation of relationships between users. Correspondingly, this reliance on social networks is resulted in wide spread research by many companies and researchers to work on improving and enhancing the accuracy and other metrics in the Influence Maximization problem (IM). In this work, we focus on the component, which is the problem of finding a small set of k seed nodes in a social network to maximize their influence spread i.e the expected total number of activated nodes after the seed nodes are activated, under certain influence diffusion models.

General graph networks will be present in the following form:

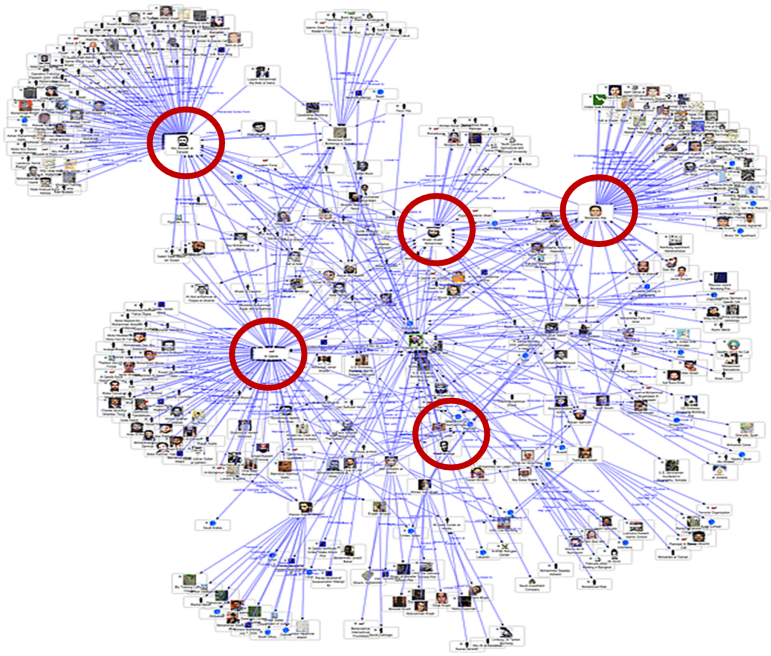

Figure 1.1: General graph networks

1.3.1 Influence Maximization in Homophilic Network

This is a case where there is homophily in the network, In this work, we concentrate on balancing the Influence maximization. Balanced Influence Maximization is the problem of selecting the k most influential nodes (seed nodes) that maximizes the spread of influence and maintains the balance between different groups or categories.
Example : 2 groups (majority , minority)

We have found that greedy algorithm fails to give the optimal solution when there is homophily in the network. Greedy approach dealt only the Influence maximization without considering about the categorical ratio existence in the final activated set. But in many general graph networks there will be high chance for the presence of homophily.

1.3.2 Optimizing Greedy Approach

In this work we focused on computational side of Greedy algorithm, where it is a fundamental approach for identifying influential nodes in a social network, where nodes with the highest potential to maximize information diffusion are selected iteratively. However, this method can be computationally expensive, especially in large networks. Introducing pruning of the candidate set nodes during algorithm execution gives us the approach called CELF, which substantially reduces the computational burden. CELF++ further refined this strategy by incorporating more advanced data structures, such as priority queues, to enhance the algorithm's efficiency. These optimizations allow CELF and CELF++ to identify k influential nodes more effectively than the basic Greedy algorithm.

The problem of finding the k-most influential nodes can be formulated as NP-Hard combinatorial optimization problem. The problem of finding the k-most influential nodes is finding the optimal subset of nodes as seed nodes and this is of exponential time complexity.

Chapter 2

Literature Review

Influence Maximization: The process of identifying a set of nodes in a network that maximizes the spread of influence, often measured by the adoption of information or behaviors

Homophily: The tendency of individuals to associate with others who are similar to them.

Monotonicity: Monotonicity in influence maximization refers to the property that adding a node to the seed set can only increase the spread of influence in a non-decreasing manner.

Submodularity: Submodularity characterizes the diminishing returns property, indicating that the marginal gain of adding a node to a larger seed set decreases as the seed set size increases in influence maximization algorithms.

Structural homophily: The tendency that a node connects to other similar node in the network.

Influence diffusion homophily: The tendency that a node can influence other similar node.

Balance: When categorical ratio between the nodes in the network is preserved in the active set.

Greedy Algorithm: An algorithmic approach for influence maximization that selects nodes iteratively, adding the most influential node at each step based on a defined heuristic.

CELF (Cost-Effective Lazy Forward) Algorithm: An optimization of the Greedy algorithm that efficiently updates influence estimates during node selection, reducing computational overhead.

CELF++ Algorithm: An extension of CELF that further improves the efficiency of influence maximization by incorporating additional optimizations or heuristics.

Seed Set: A subset of nodes in a network chosen as initial adopters to maximize influence propagation in influence maximization algorithms.

Marginal Gain: The additional influence gained by adding a specific node to an existing set of selected nodes, a crucial factor in the Greedy and CELF algorithms.

Cascade Model: A mathematical representation of how influence spreads through a network, defining the rules governing the adoption of behaviors or information by connected nodes.

Heuristic: A rule of thumb or a practical approach used to make decisions in the absence of complete information, often employed in influence maximization algorithms.

Lazy Evaluation: A strategy, particularly in CELF and CELF++, where certain computations are deferred until necessary to optimize the efficiency of influence estimation.

Approximation Algorithm: An algorithm that provides solutions close to the optimal solution within a reasonable amount of time, often used in influence maximization due to its NP-hard nature.

Random Graph Model: A mathematical model that generates graphs based on a random process, such as the Erdos-Renyi model, often used to evaluate influence maximization algorithms.

Convergence Criteria: The conditions under which an algorithm is considered to have reached a stable or optimal solution during its execution.

Kempe et al. have demonstrated that the influence maximization problem is NP-hard and a simple greedy algorithm can guarantee the best possible approximation factor in polynomial time. CELF algorithm was proposed by Chen et al. in 2009, with the help of sub modularity property they developed an efficient algorithm named CELF that scales to larger problem achieving the near optimal solution faster compared to greedy approach. Goyal et al. have proposed a CELF++ algorithm by optimizing the previously existing CELF algorithm by addition of a priority queue which helps in decreasing the run time and faster computation.

Azaouzi M et al. conducted a survey on new trends in influence maximization models. In this survey they divided the diffusion models into two categories i.e., induvial and group node based models. They compared and contrasted various models into the above two categories in the survey.

Chapter 3

Properties of social Network

3.1 Sub-modular Functions

A sub-modular function is a mathematical idea that helps us understand a certain type of set function. A set function gives a value to each subset of items taken from a given set. In the context of influence maximization, functions have a property called "Diminishing Returns", which means that when we add more elements to a set, the marginal gain we get from each new addition in the function's value becomes smaller. The finite set represents the users or nodes in a social network. The set function gives a value to each subset of nodes, showing the influence spread achieved by choosing that subset as seed nodes.

$$F(S \cup \{u\}) - F(S) \geq F(T \cup \{u\}) - F(T) \quad \forall S \subseteq T \tag{3.1}$$

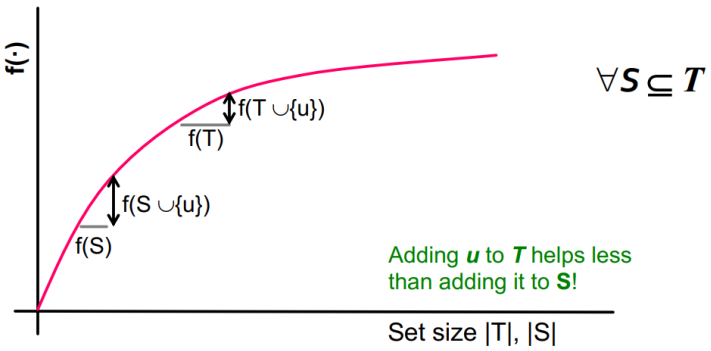

Figure 3.1: Diminishing returns.

This shows the diminishing returns property of sub-modular functions. Adding an additional node to the seed set provides less incremental benefit as the seed set grows. Researchers use the characteristics of sub-modular functions to design approximation algorithms for influence maximization. These algorithms efficiently help to find a near optimal solution i.e., a subset of seed nodes based on the sub modularity of the influence function, which allows them to find a balance

between computational efficiency and maximizing influence spread in social networks. Approximating algorithms for maximizing the spread of influence in diffusion models can be developed in a general framework based on sub-modular functions

3.2 Homophily

Homophily is a property where there is a tendency of an individual to associate with others who are similar to them (i.e, property of like to connect with like). Measuring homophily involves assessing the degree of similarity among connected nodes within a network. One common method is to use the Homophily Index, which is calculated by comparing the observed proportion of connections between nodes with similar characteristics to the expected proportion under a null model.

If we consider a party of 50 old age people and 50 teenagers. Consider the relationship between them as friendship. This scenario we can correlate tossing a coin twice the possibilities are: HH, TT, HT, TH. If we randomly pick an edge then the probability of connection between head and tail is ½ . if we randomly pick a friendship between them, if the probability is less than ½ then we say there exists a homophily. So

$$\text{Homophily Index} = 1 - \text{Actual/Expected}$$

If Homophily Index $< 1/2$, then we can say that the graph contains homophilic
If Homophily Index $> 1/2$, then we can say that the graph is hetrogenity.
Where expected is half of the total number of edges present in the network and actual is the count of edges present between the groups

Chapter 4

Diffusion Models

Diffusion models are mathematical and computational models used to study how the spread of information, innovations, behaviours or influence happens within a network or population over time. These models determine how the information is propagated within the networks or in between the networks. If we consider spreading of an information or idea through a social network represented by a directed graph G, we will consider each individual node present in 2 states either being active (an adapter of the idea) or inactive(not adapted yet).
There are two basic diffusion models:

1. Linear Threshold Model
2. Independent Cascade Model

4.1 Linear Threshold Model

The Linear Threshold Model is used to describe how information or influence spreads within a social network. The Model spreads the influence in a network by considering the activation of nodes based on their increasing influence from their network connections and individual thresholds. The selection of seed nodes is important in determining the extent of influence spread in the network.
In this model we use a social network represented as a graph where nodes or users are connected by edges (which represent their connections) and each node is randomly associated with a threshold value between 0 and 1. These threshold values represents the minimum level of influence required for the node to become active. Then we choose a set of initial nodes which will start the influence process (We might pick these based on their likelihood to influence others)

Input: Graph G, Seed set S, k
Output: Set of activated nodes in the graph

Algorithm 1 Linear Threshold Model

Set the number of iterations T to a large value.
For each node u in V, assign a unique activation probability p_u.
for $t = 1$ to T **do**
    Set the state of all nodes in S to activated.
    Initialize the set of activated nodes A to S.
    while A is not empty **do**
        Choose a node v from A uniformly at random.
        Compute the activation weight w_v as the sum of the weights of all activated neighbors of v in A.
    end while
    Record the number of activated nodes for this iteration, denoted by σ_t.
end for
Compute the influence spread for each node u as the average number of activated nodes over T iterations when u is in the seed set, denoted by σ_u.
Initialize the set of selected nodes U to S.
while $|U| < k$ **do**
    Select the node u with the maximum influence spread σ_u among nodes in $V \setminus U$.
    Add u to U.
end while
Output the set of selected nodes U.

Now we initialize the activation states of all nodes in the network. Seed nodes are set to "active" $(\sigma_i = 1)$ and all other nodes are "inactive" $(\sigma_i = 0)$.

Let us take an example for a better understanding:
Consider a node v where:
i. The node v has random threshold $\Theta_v \in [0,1]$
ii. The node v is influenced by each neighbour w according to a weight $b_{v,w}$ such that

$$b_{v,w} \leq 1 \quad : w \in \text{neighbor of } v \tag{4.1}$$

iii. The node becomes active when at least (weighted) Θ_v fraction of its neighbours are active

$$\sum_{w \text{ active neighbor of } v} b_{v,w} \geq \theta_v \tag{4.2}$$

In this way, the activation process starts with a set of seed nodes that are initially active. At each time step, nodes that have a sufficient level of influence become active and can activate their neighbours. The process continues until no more nodes can be activated The influence propagation

process continues through iteration causing a cascade effect. As influence spreads more nodes become active and the process stops when there are no more changes in activation states. The final set of active nodes represents the outcome of the influence maximization.

4.2 Independent Cascade Model

The primary objective of using the Independent Cascade Model is to identify the influence spread for the given initial seed nodes. The Independent Cascade Model is a way to predict how things go viral on social networks. It is like a tool used by researchers to understand how information or trends spread in the field of network theory and influence maximization. It is designed to simulate the spread of influence in a social network or graph.
Input: Graph G, Seed Set S, k.
Output: Set of k nodes to maximize influence spread.

Algorithm 2 Independent Cascade Model Algorithm

Input: Graph G, Seed Set S
Output: Activated Nodes A
Initialize set of activated nodes A to S.
Initialize set of newly activated nodes $NewlyActivated$ to S.
while $NewlyActivated$ is not empty **do**
    Initialize an empty set $Temp$.
    for each node v in $NewlyActivated$ **do**
        for each neighbor u of v not in A **do**
            Generate a random number $r \in [0,1]$.
            if $r \leq$ activation probability of edge (u,v) **then**
                Add u to $Temp$.
            end if
        end for
    end for
    Add $Temp$ to A.
    Set $NewlyActivated$ to $Temp$.
end while
Return Activated Nodes A.

In this model, the process of propagation is based on the concept of "independent cases", So, it's about how an action of one individual can trigger the activation of their neighbours with a certain probability. Now let us see how this model works:

Let us take a directed finite graph G = (V, E)
Choose a set S of initial seed nodes. These nodes are initially activated, This can be done randomly, or by selecting a specific set of nodes that are thought to be influential.
Each edge (v,w) has a probability (weight) P_{vw}.

If node 'v' is active, it gets one chance to make w active, with probability P_{vw}.
P_{vw} is Probability that node v can activate node w.

- In each time step, each activated node has a single chance to activate each of its inactive neighbours. The probability of a successful activation is given by a propagation probability, which is typically assumed to be the same for all edges in the network.
- Once a node is activated, it remains activated forever. Inactive nodes that are successfully activated become active in the next time step.
- The process terminates when no new nodes can be activated.
- The final set of active nodes represents the outcome of the influence maximization.

In summary, the Independent Cascade Model is a valuable tool for understanding and predicting how information and influence spread through social networks. By selecting the right initial seed nodes and considering the probability of influence transmission, we can analyse the dynamics of influence propagation. The ultimate goal is to identify the most effective strategies for maximizing the reach and impact of a message, idea, or behaviour in a network. This model's insights can inform real-world strategies for viral marketing, public health campaigns, and many other applications where spreading influence is essential for success.

4.2.1 Working of Approximation Algorithm

We use the greedy algorithm to find the appropriate seed set that maximizes the influence. We run the algorithm for the k iterations
Input: We compute the Influence set X(u) of each node u: $X_u = /v_1, v_2, ...soon$. That is if we activate node u, nodesv1,v2,.....so on will eventually get activate.
Algorithm:-
1.At each iteration we select the node that maximizes the influence when we do the union with the seed set.
2.At each iteration i, activate the node u that gives the largest marginal gain i.e max f(S(i-1) U u)
Implementation of this Algorithm:-

- We maintain a priority queue to store the marginal gains of each node we pop the nodes from the queue and run the algorithm until we reach the k seed nodes.
- We compute the influence for each node in each relation and we take the expected value of it.
- Expected Influence = New Influence sum/Number of relations.
- Influence gain normalizer = $(1 - \lambda)/ |V|$, where ($|V|$ = No of vertices)
- New Objective = Influence Gain Normalizer * Expected Influence
- Gain = New Objective – Current Objective

- We push the node with the computed gain into the priority queue

Hill Climbing Algorithm Time Complexity: $O(K \times n \times R \times m)$
Where K is the k-most influential nodes, n is the total number of nodes, R is the number of Parallel works(realizations) we simulate and m is the number of edges.
Delta(S) ($\delta(S)$)= Measure of balance, We compute delta S the trade off between Balance and P_m.
P_m: Fraction of Majority nodes that were present in the Initial graph nodes.
f(S) = M(S) + m(S) where M(S) is the n.o of majority nodes in activated set and m(S) is the no.of minority nodes in the activated set. Trade off between Balance and P_t:

$$\Delta(S) = M(S) - p_t f(S)$$

Figure 4.1: Trade off between Balance and P_t

In the above equation M(S) is the observed size of majority nodes in the final Influence set(Activated set) after the Diffusion phenomenon. Majority target size in active set = p_t*f(S)
Intution: The intuition behind maintaining the categorial balance is to preserve the majority vs minority nodes ratio(P_m) in the activated set.The ideal case is delta(S) is 0 when $P_t = P_m$.
No Homophily: In the no homophily networks the edge connections are randomized . We generate the network using network generation model. Then, we use the Influence Diffusion model(Independent cascade model) for the information propagation.

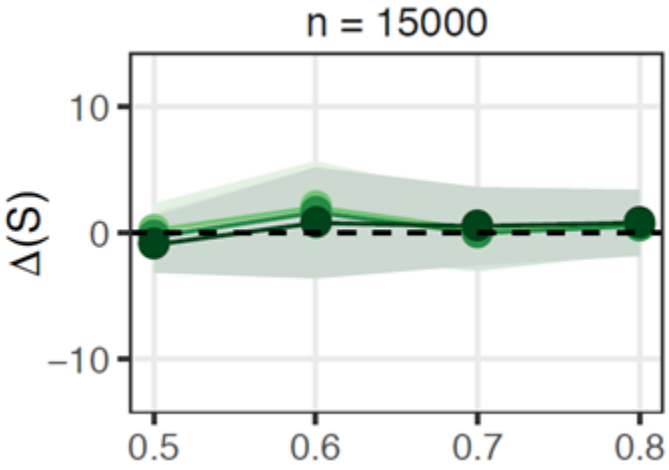

Figure 4.2: No Homophily

As there is no homophily the greedy algorithm selects the k most influential nodes by maintaining the categorical balance between majority and minority nodes.Because of randomized edge connections. So the value of delta(S) is approximately 0.
Structural Homophily: A social network phenomenon where individuals who share similar network attributes or structural positions are more likely to form connections or ties with each other.

$$h(v, w) = \begin{cases} h & \text{if } v \text{ and } w \text{ are from the same category,} \\ 1 - h & \text{otherwise.} \end{cases}$$

4.3 Construction of Structural Homophily Graph

4.3.1 Network Generation model

We construct the network with the help of 3 parameters(n,p_m,h)

- Where n is the total number of nodes
- P_m is the fraction of majority nodes
- h is the structural homophily index

We add a single directed edge in each timestep. We construct G(t + 1) from G(t) by adding an edge to G(t) at timestep t + 1 following below scenarios:
There are 3 scenarios in constructing an edge between 2 nodes.
Scenario-1 : If we consider V as a newly created node and we choose W from the set of existing nodes with the probability proportional to take from paper .hand we assign the probability P_M(if V belongs to a majority group).The probability of forming an edge between them is alpha.

Scenario-2 : If we choose V form set of existing nodes with Probability Proportional to and choose w from all existing nodes with probability proportional to .The probability of forming edge between them is beta.

Scenario-3 : If we choose V from set of existing nodes with Probability Proportional to and new node w .The probability of forming an edge between them is Gama.
We consider Alpha = Beta = Gamma =1/3 (i.e Equally likely)
Influence Diffusion model : Influence Diffusion model does the work of propagating the information from node to node. It contains one parameter that is b_p (Base Probability), Base Probability of a node is successfully influencing a neighbor node.

4.4 Homophily vs Delta(S)

When there is homophily in the network the greedy algorithm fails to find the balanced k-most influential nodes because,if the homophily is present in the network then more number of connections will be present in between the same group of users. So,there will be more number of majority users in the seed set than needed and it was leading to imbalance between majority and minority groups in the activated set.

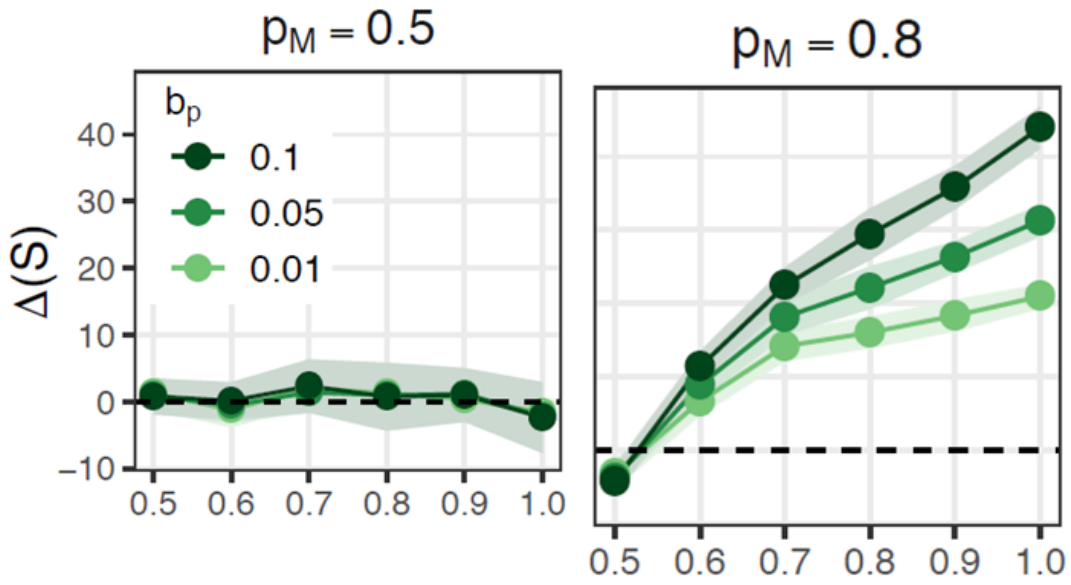

Figure 4.3: $\Delta(S)$ when $p_m = 0.5$ and $p_m = 0.8$

Chapter 5

Methodologies

5.1 Greedy Algorithm

Greedy Algorithm is a simple yet efficient heuristic approach used to find the seed set S that can maximize the influence in the network. It identifies a small set of nodes in the given network such that they have maximum influence on the network. The Greedy algorithm proposed in the Kempe et al [3], it basically finds the node with the biggest spread, adds it to the seed set and then finds the node with the next biggest marginal spread over and above the spread of the original and so on until k seed nodes are found

Algorithm 3 Greedy Algorithm for Influence Maximization

Input: Graph G, Seed Set size K
Output: Seed Set S

Start by selecting a small set of seed nodes S.
while $|S| < K$ **do**
    for each node u not in S **do**
        Estimate the expected number of nodes that would be influenced if u is added to S, denoted by $f(u)$.
    end for
    Select the node u with the highest value of $f(u)$ and add it to S.
end while
Return Seed Set S.

The expected number of nodes influenced if node u is added to the seed set S can be estimated using a probabilistic diffusion model, such as the Independent Cascade Model (ICM) or the Linear Threshold Model (LTM). The expected influence spread for node u, denoted by f(u), can be computed using the following equations:
For Independent Cascade Model:
where N is the total number of nodes in the network, is the node in the network, and is the proba-

bility that node is influenced by node u through a path that starts from the seed set S.
For the Linear Threshold Model:
Where is the expected fraction of neighbours of that will be influenced by adding node u to the seed set S, assuming that the threshold function for each node is a linear combination of its incoming edges.
The Greedy Algorithm selects nodes based on their immediate impact on the spread of influence in the network, without considering the impact of future selections. While this approach may not always guarantee an optimal solution, it often produces near-optimal results in practice.

5.2 CELF Algorithm

The CELF (Cost-Effective Lazy Forward) algorithm is an extension of the Greedy algorithm that aims to reduce the computational cost of influence maximization by avoiding the recomputation of influence spreads for nodes that are not likely to be selected as seed nodes. The CELF algorithm was developed by Leskovec et al. Although the Greedy algorithm is much quicker than solving the full problem, it is still very slow when used on real-world large-scale networks. CELF was the next version to it.

Algorithm 4 CELF (Cost-Effective Lazy Forward) Algorithm

Input: Graph G, Seed Set size K
Output: Seed Set S

1: Start by selecting a small set of seed nodes S.
2: **for** each node u not in S **do**
3: Estimate the expected number of nodes that would be influenced if u is added to S, denoted by $f(u)$.
4: **end for**
5: Sort the nodes by their expected influence in descending order.
6: Initialize a list L with the nodes sorted in step 3.
7: **while** $|S| < K$ **do**
8: Select the node u with the highest marginal gain, i.e., the node that maximizes $f(u|S) - f(u|S+\{u\})$, where $f(u|S)$ is the expected influence spread of u given the current seed set S, and $f(u|S+\{u\})$ is the expected influence spread of u given the seed set S plus the selected node u.
9: Add u to the seed set S.
10: Update the expected influence spreads of the nodes in L using the lazy evaluation technique, i.e., only compute the influence spreads of nodes that have not been evaluated yet.
11: **end while**
12: **Return** the Seed Set S.

CELF exploits the sub-modularity property of the spread function, which implies that the marginal spread of a given node in one iteration of the Greedy algorithm cannot be any larger than its marginal spread in the previous iteration. This helps us to choose the nodes for which

we evaluate the spread function in a more sophisticated manner, rather than simply evaluating the spread for all nodes. More specifically, in the first round, we calculate the spread for all nodes (like Greedy) and store them in a list/heap, which is then sorted. Naturally, the top node is added to the seed set in the first iteration, and then removed from the list/heap. In the next iteration, only the spread for the top node is calculated. If, after resorting, that node remains at the top of the list/heap, then it must have the highest marginal gain of all nodes. Why? Because we know that if we calculated the marginal gain for all other nodes, they'd be lower than the value currently in the list (due to submodularity) and therefore the "top node" would remain on top. This process continues, finding the node that remains on top after calculating its marginal spread, and then adding it to the seed set. By avoiding calculating the spread for many nodes, CELF turns out to be much faster than Greedy, which we'll show below.

The celf() function below that implements the algorithm, is split into two components. The first component, like the Greedy algorithm, iterates over each node in the graph and selects the node with the highest spread into the seed set. However, it also stores the spreads of each node for use in the second component.

The second component iterates to find the remaining k1 seed nodes. Within each iteration, the algorithm evaluates the marginal spread of the top node. If, after resorting, the top node stays in place then that node is selected as the next seed node. If not, then the marginal spread of the new top node is evaluated and so on.

Like greedy(), the function returns the optimal seed set, the resulting spread and the time taken to compute each iteration. In addition, it also returns the list lookups, which keeps track of how many spread calculations were performed at each iteration. We didn't bother doing this for greedy() because we know the number of spread calculations in iteration i is Ni1.

CELF exploits the sub-modularity property of the spread function, which implies that the marginal spread of a given node in one iteration of the Greedy algorithm cannot be any larger than its marginal spread in the previous iteration. This helps us to choose the nodes for which we evaluate the spread function in a more sophisticated manner, rather than simply evaluating the spread for all nodes. More specifically, in the first round, we calculate the spread for all nodes (like Greedy) and store them in a list, which is then sorted. Naturally, the top node is added to the seed set in the first iteration, and then removed from the list. In the next iteration, only the spread for the top node is calculated. If, after resorting, that node remains at the top of the list, then it must have the highest marginal gain of all nodes. This process continues, finding the node that remains on top after calculating its marginal spread, and then adding it to the seed set. By avoiding calculating the spread for many nodes, CELF turns out to be much faster than Greedy, which we'll show below.

5.3 CELF ++

The CELF++ (Cost-Effective Lazy Forward++) is an extension of the CELF algorithm for influence maximization that further reduces the computational cost by improving the efficiency of the lazy evaluation technique used in CELF.

The CELF++ algorithm maintains a priority queue Q of nodes sorted by their expected influence, and uses the queue to efficiently evaluate the marginal gains of nodes. The lazy evaluation

technique is used to avoid re-computing the influence spreads of nodes that have already been evaluated. The algorithm terminates when the desired number of seed nodes have been selected. Although CELF++ maintains a larger data structure to store the look-ahead marginal gains of each node, the increase of the memory consumption is insignificant. Overall, the CELF++ algorithm is an effective and efficient algorithm for influence maximization in social networks. It has been shown to outperform other state-of-the-art algorithms in terms of both running time and influence spread. Here's how the CELF++ algorithm works:

Algorithm 5 CELF++ (Cost-Effective Lazy Forward++) Algorithm

Input: Graph G, Seed Set size K
Output: Seed Set S

1: Start by selecting a small set of seed nodes S.
2: **for** each node u not in S **do**
3: Estimate the expected number of nodes that would be influenced if u is added to S, denoted by $f(u)$.
4: **end for**
5: Sort the nodes by their expected influence in descending order.
6: Initialize a list L with the nodes sorted in step 3.
7: Initialize a priority queue Q.
8: **for** each node u in L **do**
9: Add $(u, f(u))$ to Q.
10: **end for**
11: **while** $|S| < K$ **do**
12: Let $(u, f(u))$ be the first node in Q.
13: **if** $f(u|S) < f(u)$ **then**
14: Remove $(u, f(u))$ from Q and continue with the next node in Q.
15: **end if**
16: Add u to the seed set S.
17: **for** each node v in L that has not been evaluated yet **do**
18: Compute its marginal gain, i.e., $f(v|S) - f(v|S + \{u\})$, and update its value in Q if it is greater than the current value.
19: **end for**
20: **if** the marginal gain of any node in L is greater than or equal to the marginal gain of u **then**
21: Add that node and its marginal gain to Q and continue with the next node in Q.
22: **else**
23: Remove $(u, f(u))$ from Q and continue with the next node in Q.
24: **end if**
25: Update the expected influence spreads of the nodes in L using the lazy evaluation technique, i.e., only compute the influence spreads of nodes that have not been evaluated yet.
26: **end while**
27: **Return** Seed Set S.

5.4 Datasets

5.4.1 Karate Club Dataset

The Karate Club dataset is a well-known social network graph that represents friendships between 34 members of a karate club at a US university. It is a simple undirected graph with 34 nodes known as club members and 78 edges known as friendship ties. It is a common dataset often used as a benchmark for testing and comparing network analysis algorithms. The Karate Club dataset is usually represented as an edge list, where each row represents an edge between two nodes. The two columns in the edge list correspond to the node IDs of the two endpoints of the edge.

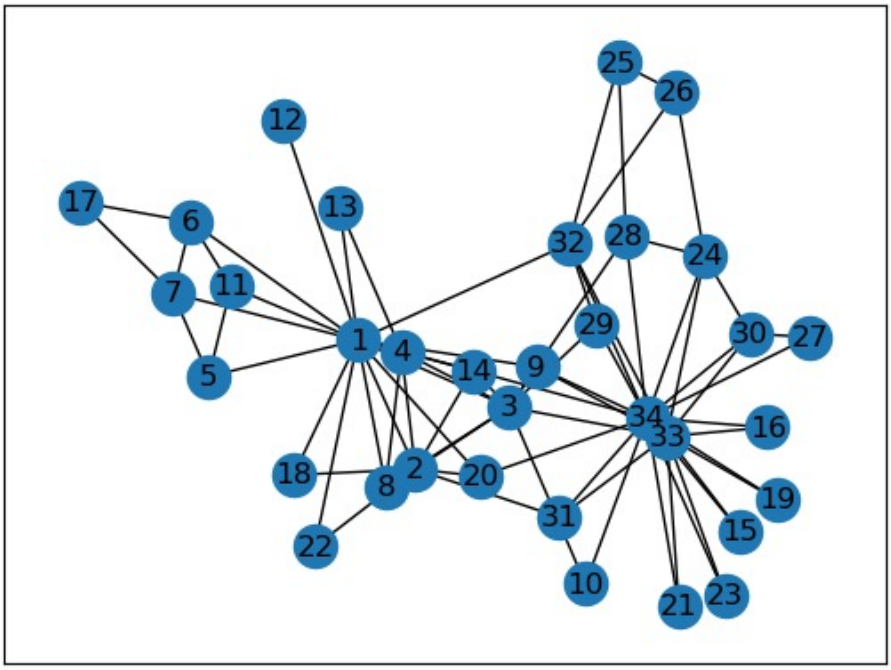

Figure 5.1: 2D view of karate club network

5.4.2 Erdos Renyi Graph (random graph)

Erdős-Rényi random graph generator: It is a method used for generating random graph datasets with specified number of nodes and edges, proposed by Paul Erdos and Alfred Renyi. In this method, a graph with n nodes is generated by randomly adding edges between nodes with a certain probability p. The probability of an edge being present between any two nodes is independent of the presence or absence of any other edge in the graph. For this work we used he G (n, p) model, where each possible edge between the n nodes is present with probability p, independently from every other edge. To be precise for this while performing this experiment a random graph is

generated with 100 nodes and edge probability of 0.1. We obtained a graph with 100 nodes and 523 edges.

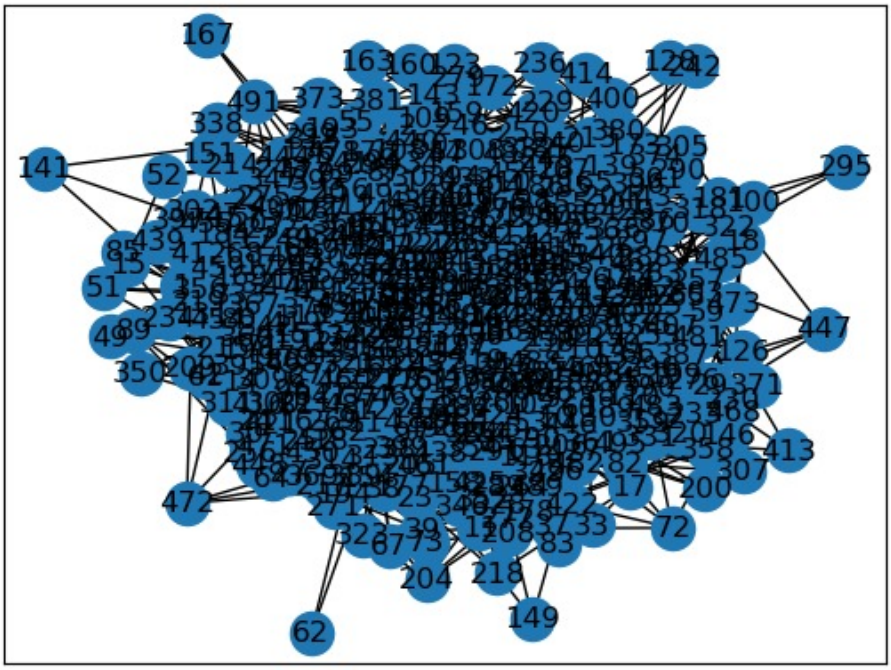

Figure 5.2: 2D view of Erdos Renyi Graph (random graph)

Chapter 6

Metrics

The metrics used to benchmark the proposed model along with the baseline models are DNI (Distant Nodes Influence) and Time complexity or Run Time of each approach. DNI, which refers to the idea that the influence of a set of nodes can extend beyond their immediate neighbors in the network. DNI can be calculated with the help of various propagation models like Independent Cascade (IC) and Linear Threshold (LT) models to calculate the spread of the influence from the seed set S. Run Time or Time Complexity is the measure used to evaluate the running time of the model. Approaches with low Time Complexity are preferred as they run faster and can handle large data. In this work, Independent Cascade (IC) model is used to calculate the spread or DNI of the frameworks.

6.1 Results And Declarations

For evaluating the performance of our proposed model, it is tested on various datasets named Karate Club network dataset, Erdős-Rényi random graph generator. These datasets comprise nearly 34, 100 nodes.
It was executed in Google Collaboratory IDE and Anaconda Jupiter notebook, and Python 3.10.10 was utilized. To evaluate the performance, various models are used as benchmarks. The baseline models used are Greedy, CELF, CELF++ were trained on edge lists of the above two datasets. The metrics used to benchmark the proposed model along with the baseline models are DNI (Distant Nodes Influence) and Time complexity or Run Time of each approach. The Run time of the proposed model is compared with various models in the table below along with the edge version of the models of the same models.

Approach	Greedy	CELF	CELF++
erdos renyi graph	362.8 sec	74.21 sec	71 sec
karate club	11.49 sec	2.99 sec	1.27 sec

Table 6.1: Run time of various approaches compared with the base line greedy

Approach	Greedy	CELF	CELF++
erdos renyi graph	67	67	100
karate club	30	31	32

Table 6.2: DNI for various approaches compared with the base line greedy

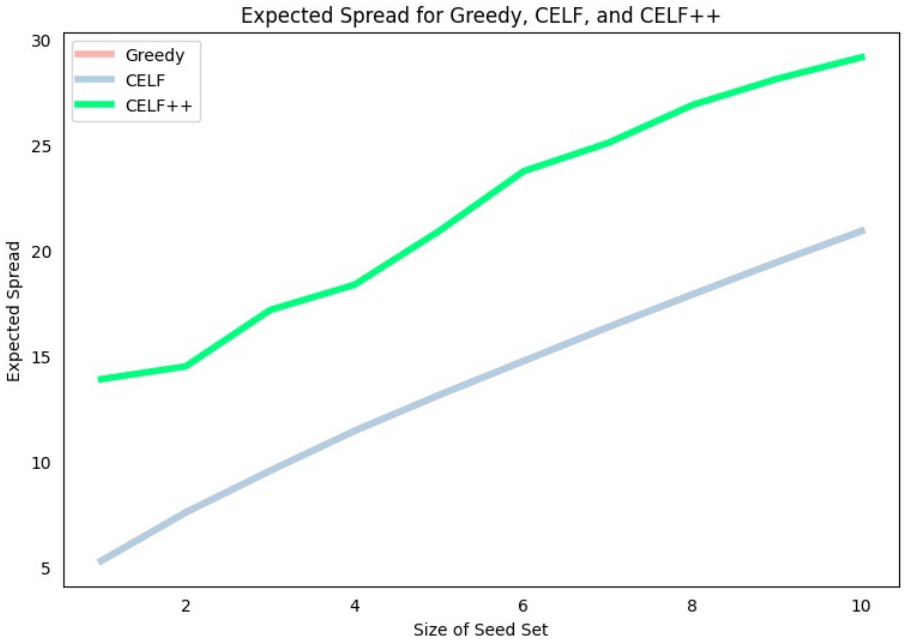

Figure 6.1: Spread Comparision Graph

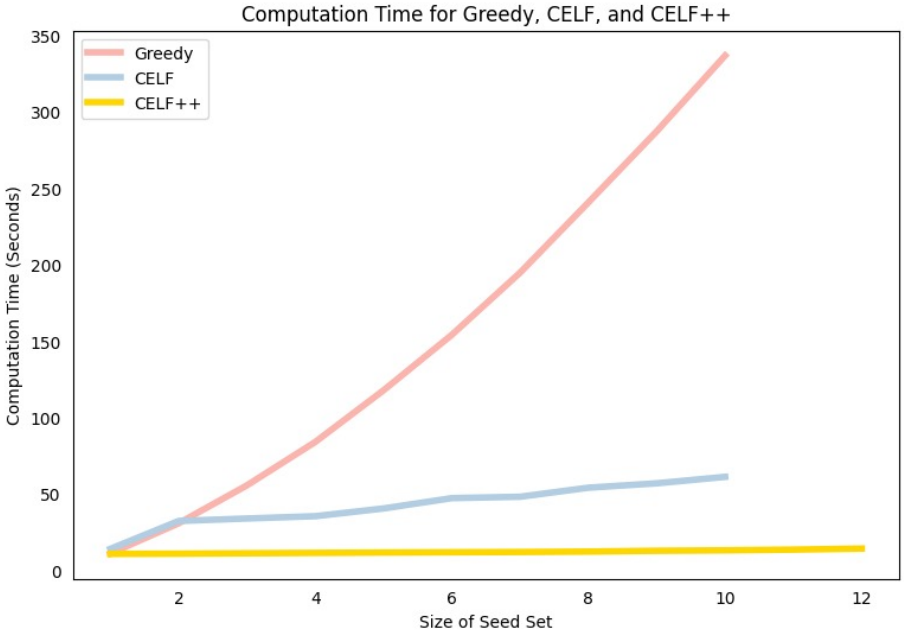

Figure 6.2: Computation Time Comparision Graph

Chapter 7

Conclusion and Future Scope

In this work we demonstrated that CELF and CELF++ performed very well compared to greedy in terms of run-time. CELF and CELF++ did not show much difference in the spread compared to Greedy. We have integrated CELF and CELF++ directly on the given social network. To get the better efficiency in terms of spread, we will integrate CELF and CELF++ with the graph embeddings generated by the deep learning model.

In this study, our investigation into the performance of influence maximization algorithms unveiled compelling results regarding the efficiency of CELF and CELF++ compared to Greedy. Notably, both algorithms exhibited superior run-time performance when compared to the traditional Greedy algorithm. This finding underscores the potential of CELF and its enhanced variant, CELF++, to expedite the identification of influential nodes within social networks. Despite their notable efficiency gains, our analysis revealed that CELF and CELF++ did not substantially differ from Greedy in terms of influence spread. The influence spread, a critical metric for evaluating the effectiveness of these algorithms, remained relatively consistent across all three methods. This suggests that while CELF and CELF++ significantly optimize computational time, their impact on the extent of information diffusion within the network does not markedly surpass that achieved by the Greedy approach.

As a future work to further enhance the efficiency of influence maximization, we will extended our exploration by directly integrating CELF and CELF++ with graph embeddings generated by a deep learning model. This novel approach aims to capitalize on the strengths of both traditional algorithms and advanced machine learning techniques. By combining the strategic node selection capabilities of CELF and CELF++ with the expressive power of graph embeddings, we anticipate Furthermore, there are various techniques emerging to efficiently generate the graph or node embeddings. These methods can help to further generate better embeddings to find the better seed set using the Influence Maximization Algorithms. The use of GANs for generating graph or node embeddings has shown promising results in various tasks, including link prediction, node classification, and graph generation.

References

[1] Kempe D., Kleinberg, J., and Tardos, É. (2003, August). Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 137-146).

[2] J. Leskovec et al. Cost-effective outbreak detection in networks. In KDD 2007.

[3] Goyal, A., Lu, W., and Lakshmanan, L. V. (2011, March). Celf++ optimizing the greedy algorithm for influence maximization in social networks. In Proceedings of the 20th international conference companion on World wide web (pp. 47-48).
[4] "Balanced Influence Maximization in the Presence of Homophily" is a research paper by Md Sanzeed Anwar, Martin Saveski, and Deb Roy.

[5] Azaouzi, M., Mnasri, W., and Romdhane, L. B. (2021). New trends in influence maximization models. Computer Science Review, 40, 100393.

[6] Wu, G., Gao, X., Yan, G., and Chen, G. (2021). Parallel greedy algorithm to multiple influence maximization in social network. ACM Transactions on Knowledge Discovery from Data (TKDD), 15(3), 1-21.

[7] Arora, A., Galhotra, S., and Ranu, S. (2017, May). Debunking the myths of influence maximization: An in-depth benchmarking study. In Proceedings of the 2017 ACM international conference on management of data (pp. 651-666).

[8] Y. Zeng, X. Chen, G. Cong, S. Qin, J. Tang, Y. Xiang, Maximizing influence under influence loss constraint in social networks, Expert Syst. Appl. 55 (2016) 255–267, http://dx.doi.org/10.1016/j.eswa.2016.01.008.

[9] A. Goyal, W. Lu, L.V. Lakshmanan, Simpath: An efficient algorithm for influence maximization under the linear threshold model, in: 2011 IEEE 11th International Conference on Data Mining, IEEE, 2011, pp. 211–220.//

Bibliography

[1]https://ethen8181.github.io/machine-learning/networkx/max$_i nfluence/max_i nfluence.html$

[2]$https://hautahi.com/im_g reedycelf$

[3]$https://homes.cs.washington.edu/\ marcotcr/blog/greedy-submodular/$

[4]$https://www.openu.ac.il/personal_s ites/moran-feldman/publications/Handbook$2018.$pdf$.

[5]$https://snap.stanford.edu/class/cs224w-$2019/

Acknowledgment

The success and outcome of this project required a lot of guidance and assistance from many people, and We are privileged to have got this all along with the completion of my project. Everything We have done is because of such guidance and help, and We will never hesitate to thank them.
We owe our sincere gratitude to our project guide Mrs. B.S.S. Monica, Department of Computer Science, National Institute of Technology, Andhra Pradesh, who took keen interest and guided us all along, till the completion of our project work by providing all the necessary information.
We are grateful and lucky enough to receive consistent motivation, support, and guidance from all the staff of the Computer Science Department who have helped us to complete our project work successfully. We would also like to extend our sincere gratitude to all my friends for their timely support.

Thank You.

Signatures

Seedella Sai Vikas	**Vedagiri Krishna Sandeep**	**Matta Rahul**
420233	**420247**	**420206**
Date:	**Date:**	**Date:**

Mrs. B.S.S. Monica
Date: