INFLUENCE MAXMIZATION USING INFLUECNE AND SUSCEPTIBILITY EMBEDDINGS

by

MADDINENI SAKETH	***411947***
KALAKONDA VISWA SAHITHI	***411936***
GONDA VAMSHI	***411925***

Under the guidance of

Dr. NAGESH BHATTU SRISTY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

TADEPALLIGUDEM – 534102, INDIA

APRIL-2023

INFLUENCE MAXIMIZATION USING INFLUENCE AND SUSCEPTIBILITY EMBEDDINGS

Submitted in the partial fulfillment of the requirements

of the degree of

Bachelor of Technology

by

Maddineni Saketh	***411947***
Kalakonda Viswa Sahithi	***411936***
Gonda Vamshi	***411925***

Supervisor
Dr. NAGESH BHATTU SRISTY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

TADEPALLIGUDEM- 534102, INDIA

APRIL 2023

Place: Tadepalligudem

PROJECT WORK APPROVAL

This project work entitled "INFLUENCE MAXMIZATION USING INFLUECNE AND SUSCEPTIBILTY EMBEDDINGS" was worked out by Maddineni Saketh (411947), Kalakonda Viswa Sahithi (411936) and Gonda Vamshi (411925) is approved for the degree of Bachelor of Technology in Computer Science and Engineering

Examiners

_____

_____

_____

Supervisor(s)

_____

_____

_____

Chairman

_____

Date:

Place: Tadepalligudem

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' 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.

Maddineni Saketh	Kalakonda Viswa Sahithi	Gonda Vamshi
411947	411936	411925
Date :	Date :	Date :

CERTIFICATE

It is certified that the work contained in the thesis titled "**INFLUENCE MAXMIZATION USING INFLUECNE AND SUSCEPTIBILTY EMBEDDINGS**" by "Maddineni Saketh (411947), Kalakonda Vishwa Sahithi (411936), and Gonda Vamshi (411925)" have been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr. NAGESH BHATTU SRISTY

Computer Science and Engineering

NIT Andhra Pradesh

April-2023

ABSTRACT

Influence maximization is a fundamental problem in social network analysis, which aims to identify a small set of individuals in a network that can influence a large fraction of the population. An individual's decision whether to adopt a product or innovation will be highly dependent on the choices made by the individual's peers or neighbors in the social network. Graph embeddings have emerged as a powerful tool for analyzing and modeling complex networks. These methods learn low-dimensional representations (embeddings) of nodes in the network, which capture the structural properties of the network and the interactions between nodes. These embeddings can then be used to estimate the influence and susceptibility of each node in the network. In this work, we propose a novel approach for influence maximization using graph embeddings and the CELFIE algorithm.

Our method first learns a low-dimensional representation of the network using a graph embedding technique and then uses the learned embeddings to identify the most influential nodes in the network. We performed influence maximization on large scale publicly available social networks such as Karate Club, *Erdős-Rényi* random graph generator, and Facebook Politician Dataset. Rather directly performing influence maximization on social network, influence maximization is performed on embeddings generated from network. For generation of these embeddings deep learning model named generative adversarial networks are used, during the recent periods there is wide increase in the application of GAN's.

We demonstrate the effectiveness of our approach on several real-world social networks and show that it outperforms existing methods for influence maximization in terms of both efficiency and effectiveness. The whole framework relies on generating the node embeddings, which are further subjected to influence maximization algorithms for improved efficiency. Our results suggest that graph embeddings can be a valuable tool for understanding and predicting social influence in complex networks.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CELF	Cost Effective Lazy Forward
CELF++	Cost Effective Lazy Forward++
CELFIE	Cost-Effective Lazy Forward with Influence Embeddings
GAN	Generative Adversarial Network
IC	Independent Cascade
LT	Linear Threshold
DNI	Distant Nodes Influenced

LIST OF TABLES

Chapter 1

INTRODUCTION

The influence maximization problem is a fundamental problem in network analysis and social media marketing. It involves identifying a set of influential individuals or nodes in a network, with the goal of maximizing the spread of information or influence within the network. The problem is particularly relevant in today's digital age, where social media platforms such as Facebook, Twitter, and Instagram have become powerful tools for spreading ideas, opinions, and marketing messages. By identifying key individuals who can trigger a cascade of influence, marketers and advertisers can effectively target their messages and reach a wider audience. In recent years, various algorithms and techniques have been developed to solve the influence maximization problem, making it an active area of research in computer science, network science, and social psychology.

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.
- The usage of influence and susceptibility embeddings is one of the methods for influence maximization that scholars have suggested recently.

- Susceptibility embeddings capture the possibility of a node being influenced by others, whereas influence embeddings capture the potential for a node to do so.
- These embeddings are learned from the structural features of the network and can be used to identify the most influential nodes in the network.

1.2 Problem Statement:

To identify a small set of individuals in social network who can have greatest impact on behavior on rest of the network i.e., who 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* - the expected total number of activated nodes after the seed nodes are activated, under certain influence diffusion models.

We start of by taking the small set of **k** seed nodes i.e., seed nodes to start the influence maximizations, these nodes are considered the most “influential” nodes in the graph, such that we have a wide or larger influence spread among the social network. How many nodes eventually get activated depends on which nodes were activated to begin with So, the key aspect is to select the which **k** nodes to be selected for maximum spread. During the past many researchers have tried to work with node embeddings instead of directly working with the graph itself as many machine learning and deep learning algorithms works better or efficiently with some form of vector input rather than direct raw graph, as the social networks have no natural ordering, thus they need a different approach for ML.

To overcome this problem, we come up with node embeddings as they are a way of representing nodes as vectors and they also capture the topology of network. For these added benefits many researchers are using embeddings for machine learning prediction tasks and many more. Generative Adversarial Networks (GANs) are a type of deep learning model that uses two neural networks, a generator and a discriminator, to generate new, previously unseen data that is similar to a given training dataset. During the recent times GAN’s have a seen a rapid growth in popularity due to its high versatility and efficiency in machine learning tasks. GAN’s generate data the look similar to original data. If you give GAN an image then it will generate a new version of the image which looks similar to the original image. Similarly, it can generate different versions of the text, video, audio.

Panagopoulos G et.al, [1] have proposed an algorithm named CELFIE (Cost Effective Lazy Forward with Influence Embeddings), an influence maximization method that utilizes learnt influence representations from diffusion cascades to overcome the use of diffusion models. This algorithm is a fast influence maximization method-based diffusion probabilities to find the **k** starting nodes that would maximize the reach of a diffusion cascade starting from the seed nodes.

1.4 Motivation

- The main motivation behind trying to solve the influence maximization problem and inventing new approaches is to identify the most influential individuals in a social network and maximize the spread of influence through them.
- This has important applications in fields such as marketing, public health, and social media, where the ability to identify and target influential individuals can have a significant impact on the success of a campaign or intervention.
- For example, in marketing, identifying the most influential individuals can help companies target their advertising and promotions more effectively, leading to increased sales and brand awareness.
- In public health, identifying influential individuals can help public health officials target interventions such as vaccination campaigns or disease prevention programs more effectively, potentially saving lives and reducing the spread of infectious diseases.
- In social media, identifying influential individuals can help social media platforms and marketers target their content and advertising more effectively, leading to increased engagement and revenue.
- Therefore, there is a strong motivation to develop new and effective approaches for influence maximization, especially as social networks continue to grow in size and complexity. The development of new algorithms and techniques for influence maximization can help researchers and practitioners better understand and predict social influence, leading to more effective interventions and campaigns in a wide range of fields.

Chapter 2
LITERATURE REVIEW

This section of the thesis consists of the Literature Survey done for the project.

2.1 Literature Survey

Panagopoulos G et al. [1] have suggested a influence maximization approach called CELFIE, an influence maximization method that utilizes learnt influence representations from diffusion cascades to overcome the use of diffusion models. They were able to achieve better results in terms of seed set quality and speed, instead of directly training the model with a social graph they worked with the node embeddings generated from the same graph. Panagopoulos G et al. [2] have proposed a method called IMINFECTOR, that uses representations learned from diffusion cascades to perform model-independent influence maximization. With this approach they were able to outperform various baseline models in terms of scalability and accuracy.

Kempe et al. [3] 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. [4] 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. [5] have proposed a CELF++ algorithm by optimizing the previously existing CELF [4] algorithm by addition of a priority queue which helps in decreasing the run time and faster computation.

Azaouzi M et al. [6] 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.

G Wu et al. [7] proposed a multiple influence maximization problem where multiple information can propagate in a single network with different propagation probabilities. Further

they proposed a greedy framework to solve these types of problems with an approximation ratio of 1/3. To enhance the performance, they proposed parallel algorithms with semaphores to work simultaneously and efficiently with lower run time. C Feng et al. [8] tried to solve the problems in limited influence maximization in real-world scenarios, due to user's privacy concerns it becomes a challenging task to target users in the network unless the user wishes. They followed adaptive approach with seeding and diffusion uncertainty to overcome this problem.

A Aurora et al. [9] tried to perform an in-depth benchmarking study of influence maximization techniques on social networks. They designed a benchmarking platform which provides a way to evaluate and compare the existing techniques systematically and thoroughly under identical experimental conditions. Singer Y [10] has proposed an alternative approach to tackle the challenge of spreading information effectively through influential users in a social network. This approach uses an adaptive method which selects users in a manner which targets their influential neighbors.

In [11-43] several models have been discussed about the proposed literature

2.2 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 1: GREEDY ALGORITHM FOR INFLUENCE MAXIMIZATION

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*, estimate the expected number of nodes that would be influenced if *u* is added to *S*, denoted by f(u).
3. Select the nodes with the highest value of f(u) and add it to S.
4. Repeat steps 2 and 3 until the desired number of seed nodes have been selected.

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:

$$f(u) = \sum_{i=1}^{N} P_r\left(S \to u \to v(i)\right)$$

where *N* is the total number of nodes in the network, $v(i)$ is the i^{th} node in the network, and $P_r(S \to u \to v(i)$is the probability that node $v(i)$ is influenced by node u through a path that starts from the seed set S.

For the Linear Threshold Model:

$$f(u) = \sum_{i=1}^{N} \sigma\left(u, S, v(i)\right)$$

Where $\sigma(u, S, v(i))$ is the expected fraction of neighbours of $v(i)$ 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.

2.3 CELF (Cost-Effective Lazy Forward) 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 re-computation 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 2: 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, estimate the expected number of nodes that would be influenced if u is added to S, denoted by f(u).
3. Sort the nodes by their expected influence in descending order.
4. Initialize a list L with the nodes sorted in step 3.
5. While |S| < k:
    a. 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.
    b. Add u to the seed set S.
    c. 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.
    d. 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, 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.

2.4 CELF++ (Cost-Effective Lazy Forward++) Algorithm:

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 3: 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, estimate the expected number of nodes that would be influenced if u is added to S, denoted by f(u).
3. Sort the nodes by their expected influence in descending order.
4. Initialize a list L with the nodes sorted in step 3.
5. Initialize a priority queue Q.
6. For each node u in L, add (u, f(u)) to Q.
7. While |S| < k:
    a. Let (u, f(u)) be the first node in Q.
    b. If f(u|S) < f(u), remove (u, f(u)) from Q and continue with the next node in Q.
    c. Add u to the seed set S.
    d. For each node v in L that has not been evaluated yet, 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.
    e. If the marginal gain of any node in L is greater than or equal to the marginal gain of u, add that node and its marginal gain to Q and continue with the next node in Q. Otherwise, remove (u, f(u)) from Q and continue with the next node in Q.
    f. 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.
8. Return the Seed Set S.

2.5 CELFIE (Cost-Effective Lazy Forward with Influence Embeddings):

The CELFIE (Cost-Effective Lazy Forward with Influence Embeddings) is an extension of the CELF algorithm for influence maximization that takes into account node importance to

improve the efficiency and effectiveness of the seed node selection process. Here's how the CELFIE algorithm works:

ALGORITHM 4: CELFIE (COST-EFFECTIVE LAZY FORWARD WITH INFLUENCE EMBEDDINGS) 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, estimate the expected number of nodes that would be influenced if u is added to S, denoted by f(u).
3. Sort the nodes by their expected influence in descending order.
4. Initialize a list L with the nodes sorted in step 3.
5. While |S| < k:
    a. 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.If f(u|S) < f(u), remove (u, f(u)) from Q and continue with the next node in Q.
    b. Add u to the seed set S.
    c. 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.
    d. For each node v in L that has not been evaluated yet, estimate its importance imp(v) and compute its marginal gain, i.e., (f(v|S) - f(v|S + {u})) * imp(v), where imp(v) is the estimated importance of node v.
    e. Sort the nodes in L by their importance-weighted marginal gain in descending order
6. Return the Seed Set S.

The CELFIE algorithm uses the estimated node importance function imp(v) to weight the marginal gains of nodes in the candidate set L. By doing so, it focuses on selecting nodes that are not only likely to have a high influence spread but are also important in terms of their potential impact on the network. The algorithm terminates when the desired number of seed nodes have been selected.

2.6 Propagation models:

There are many ways of propagation of influence but in this work, we primarily focus on mainly two models named Independent Cascade Model and Linear Threshold Model.

2.6.1 Independent Cascade Model:

The Independent Cascade Model (IC Model) is a widely used probabilistic model for the spread of influence in social networks. Here is the algorithm for the IC Model in influence maximization:

ALGORITHM 5: INDEPENDENT CASCADE MODEL (IC MODEL)

Input: *Graph G, Seed Set S, k.*

Output: *Set of k nodes to maximize influence spread.*

1. Set the number of iterations T to a large value.
2. For each node u in V, assign a unique activation probability p_u.
3. For t=1 to T, do the following:
    a. Set the state of all nodes in S to activated.
    b. Initialize the set of activated nodes A to S.
    c. While A is not empty, do the following:
        i. Choose a node v from A uniformly at random.
        ii. For each neighbor w of v that is not in A, do the following:

$$With\ probability\ p\_w, activate\ w\ and\ add\ it\ to\ A.$$

    d. Record the number of activated nodes for this iteration, denoted by sigma_t.
4. 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 sigma_u.
5. Initialize the set of selected nodes U to S.
6. While |U| < k, do the following:
    a. Select the node u with the maximum influence spread sigma_u among nodes in V \ U.
    b. Add u to U.
7. Output the set of selected nodes U.

The IC Model algorithm first assigns a unique activation probability to each node, indicating the likelihood that the node will activate its neighbors. Then, for a large number of iterations, the algorithm simulates the spread of influence from the seed nodes S to the rest of the network. In each iteration, the algorithm randomly selects an activated node and attempts to activate its neighbors with the assigned activation probability. The number of activated nodes in each iteration is recorded to compute the influence spread for each node u.

2.6.2 Linear Threshold Model:

ALGORITHM 6: LINEAR THRESHOLD MODEL (LT MODEL)

Input: *Graph G, Seed Set S, k.*

Output: *Set of k nodes to maximize influence spread.*

1. Set the number of iterations T to a large value.
2. For each node u in V, assign a unique activation probability p_u.
3. For t=1 to T, do the following:
    a. Set the state of all nodes in S to activated.
    b. Initialize the set of activated nodes A to S.
    c. While A is not empty, do the following:
        i. Choose a node v from A uniformly at random.
        ii. Compute the activation weight w_v as the sum of the weights of all activated neighbors of v in A.:

$$If\ w_v \geq theta_v, activate\ v\ and\ add\ to\ A.$$

    d. Record the number of activated nodes for this iteration, denoted by sigma_t.
4. 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 sigma_u.
5. Initialize the set of selected nodes U to S.
6. While |U| < k, do the following:
    a. Select the node u with the maximum influence spread sigma_u among nodes in V \ U.
    b. Add u to U.
7. Output the set of selected nodes U.

The Linear Threshold Model algorithm first assigns a unique threshold weight to each node, indicating the amount of influence required from its neighbors to activate the node. Then, for a large number of iterations, the algorithm simulates the spread of influence from the seed nodes S to the rest of the network. In each iteration, the algorithm randomly selects an activated node and computes its activation weight as the sum of the weights of all activated neighbors. If the activation weight is greater than or equal to the node's threshold weight, the node is activated and added to the set of activated nodes A.

Finally, the algorithm selects the k nodes with the highest influence spread and returns them as the set of nodes to maximize influence spread

Chapter 3

PROPOSED APPROACH

3.1 Model Architecture:

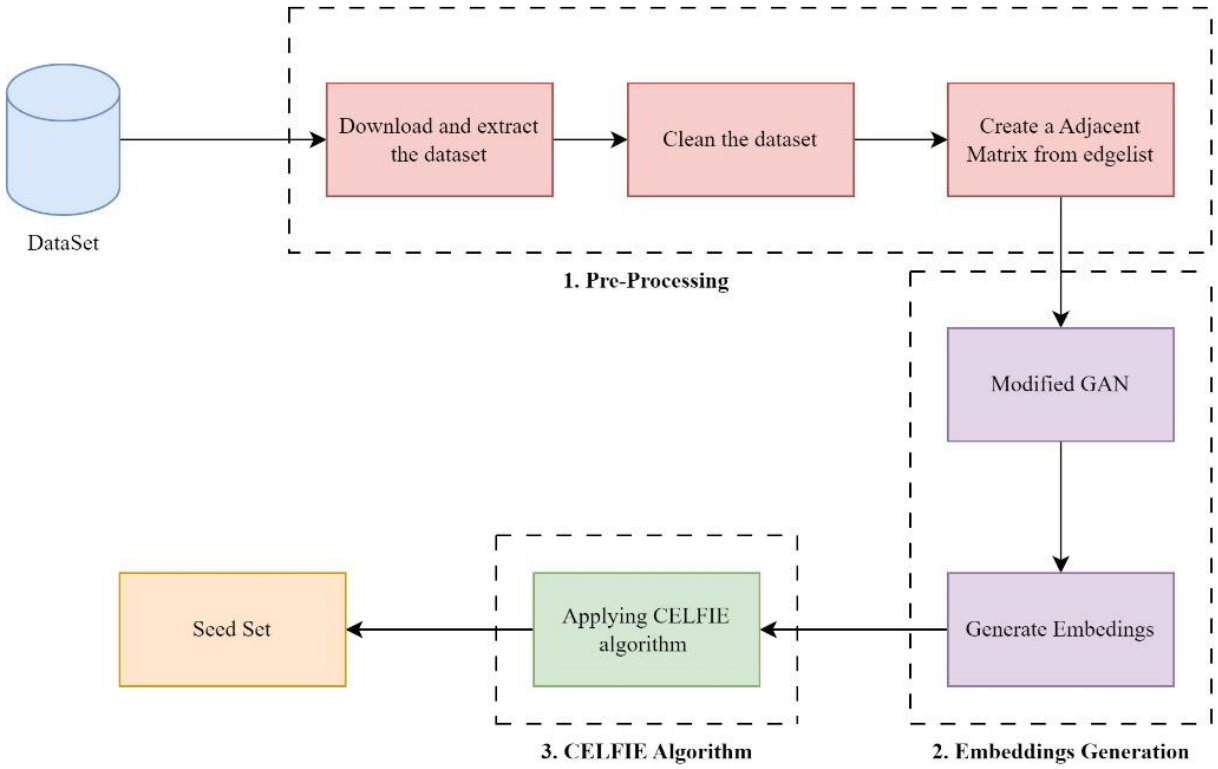

Fig. 3.1. Proposed System framework.

The proposed model uses GANs to generate synthetic node embeddings. The model uses the CELFIE algorithm to iteratively select nodes that maximize the marginal gain in influence spread when added to the set of selected nodes. The use of GANs allows for more accurate representation of the influence of nodes in the network, while the CELFIE algorithm ensures that nodes are selected in an order that maximizes the influence spread. This model has the potential to outperform traditional influence maximization algorithms by better capturing the complex relationships and dynamics in social networks.

The proposed influence maximization framework is a relied on two factors the Embeddings generation and the Influence Maximization approaches used to select the Seed Set S.

The framework is divided into 3 phases, preprocessing, embeddings generation, CELFIE algorithm.

3.2 Pre-processing:

The proposed model, which aims to select the seed set which can have maximize the influence on the network is datacentric. So, the data need to be pre-processed before proceeding to the next phase.

Preprocessing the dataset is an important step in any data analysis or machine learning project. In the context of influence maximization, the goal of preprocessing is to prepare the social network data for use in algorithms that seek to identify the most influential nodes in the network. One of the key preprocessing steps is to clean and normalize the data. This includes removing any irrelevant or duplicate data, resolving missing values or errors, and ensuring consistency across different sources of data. For instance, we might need to remove nodes or edges that do not have any significant impact on the overall network structure, such as isolated nodes or edges that represent insignificant relationships. Normalization might involve scaling or transforming the node and edge attributes to ensure that they are comparable and that their influence can be accurately measured. The preprocessed dataset should also be converted into a suitable format for use in algorithms, such as a matrix or graph representation.

Another important pre-processing step is feature extraction, which involves transforming the raw data into a set of features or attributes that capture the relevant information about each node and edge in the network. Feature extraction methods might include extracting graph-level or node-level attributes, such as degree centrality, clustering coefficient, or community membership, that are known to be relevant for identifying influential nodes. Machine learning techniques, such as clustering or dimensionality reduction, might also be used to extract relevant features from the

data. Once the features have been extracted, they can be used as input for influence maximization algorithms, such as those based on the Independent Cascade or Linear Threshold models. Pre-processing the dataset is crucial for ensuring that the influence maximization algorithms are accurate and effective in identifying the most influential nodes in the network.

As we are working with three datasets named Karate Club Dataset, *Erdős-Rényi* random graph generator, and Facebook Politicians Dataset. Karate club dataset can be downloaded directly or it can be imported from NetworkX library. Similarly, *Erdős-Rényi* random graph generator algorithm can be used to generate a random graph with specified number of inputs and outputs.

After downloading and extracting the data from the Facebook dataset, the dataset comprises of 5908 nodes with a total of 41729 connections between the nodes. As the indexing of nodes starts from zero. The dataset needs to cleaned by removing the null entries and various other formatting issues like converting to data into integer format. After cleaning the data, to work on the graph an adjacent matrix is created by using the above information. So, after the preprocessing phase we expect to get a matrix of size 5908*5908 with entries as either 0 or 1 indicating the edge between the nodes, if there is an edge between the two nodes matrix entry for those respective nodes is 1 else it is marked as 0. This preprocessing is employed for all the three datasets.

3.3 Embeddings Generation using GANs:

Generative Adversarial Networks (GANs) have been successfully applied to a wide range of data types, including images, text, and audio. More recently, GANs have been used for generating graph or node embeddings, which are low-dimensional representations of the nodes in a graph that capture their structural properties and relationships. GANs can be trained to generate embeddings that preserve the graph topology and can be used for tasks such as node classification, link prediction, and community detection.

In a GAN (Generative Adversarial Network), there are two primary components: the generator and the discriminator. The generator takes as input a random noise vector and produces a fake data sample, while the discriminator takes as input a data sample and produces a probability score indicating whether the sample is real or fake. The two components are trained together in a

competitive game, where the generator tries to produce more realistic fake samples that can fool the discriminator, and the discriminator tries to correctly distinguish between real and fake samples.

The generator is typically a neural network that takes as input a random noise vector and produces a fake data sample. The goal of the generator is to produce samples that are as similar as possible to the real data samples in the training data. The generator is trained using the feedback signal from the discriminator, which evaluates the realism of the generated samples. During training, the generator tries to minimize the discriminator's ability to distinguish between real and fake samples by adjusting its parameters based on the feedback from the discriminator.

The discriminator is also typically a neural network that takes as input a data sample and produces a probability score indicating whether the sample is real or fake. The goal of the discriminator is to correctly distinguish between real and fake samples with high accuracy. The discriminator is trained using both real data samples from the training set and fake data samples generated by the generator. During training, the discriminator tries to maximize its ability to distinguish between real and fake samples by adjusting its parameters based on the feedback from the generator.

The training process of the GAN involves iteratively updating the parameters of the generator and the discriminator networks. In each iteration, the generator produces a batch of fake data samples from the random noise vector, and the discriminator is trained on a batch of real data samples from the training set along with the batch of fake data samples produced by the generator. The discriminator then evaluates the probability of each sample being real or fake, and the generator adjusts its parameters based on the feedback from the discriminator. This process is repeated multiple times until the generator produces fake data samples that are indistinguishable from the real data samples in the training set.

One of the key challenges in training GANs is to maintain the balance between the generator and the discriminator networks. If the generator is too strong, it can produce highly realistic fake samples that can easily fool the discriminator, and if the discriminator is too strong, it can easily distinguish between real and fake samples, making it difficult for the generator to

learn from the feedback. Balancing the generator and discriminator networks requires careful tuning of the hyperparameters and network architectures, as well as appropriate regularization techniques to prevent overfitting.

In the context of generating graph or node embeddings, the generator produces fake embeddings that capture the structural properties and relationships of nodes in the graph, while the discriminator tries to distinguish between real and fake embeddings based on their graph topology. The generator and discriminator networks are trained together to optimize a specific objective function that encourages the generator to produce embeddings that are difficult for the discriminator to distinguish from real embeddings. By optimizing the objective function, the generator learns to produce high-quality embeddings that capture the essential features of the graph, while the discriminator learns to accurately distinguish between real and fake embeddings.

From the above processed adjacent matrix representation of the Facebook Politician network, we employ a deep learning model named GAN to generate the nodes embeddings for the network. GANs, or Generative Adversarial Networks, are a type of neural network that consists of two components: a generator and a discriminator:

The generator is in charge of creating fresh samples of data, and the discriminator works to separate created data from actual data. The generator and discriminator are fed data samples during the training phase, and their parameters are then iteratively updated based on how well they perform. Finding a balance where the generator can provide data samples that look genuine enough to trick the discriminator into believing they are real is the objective. GANs can be applied to a wide range of tasks, including the production of text, images, and videos. GANs can be used to create artificial networks that mimic the structural characteristics of real-world graphs in the context of node embeddings. Then, node embedding models that can learn representations of nodes in the graph that capture their connections to other nodes can be trained using these artificial graphs. This method can be helpful in situations when acquiring significant volumes of real-world graph data is challenging or expensive.

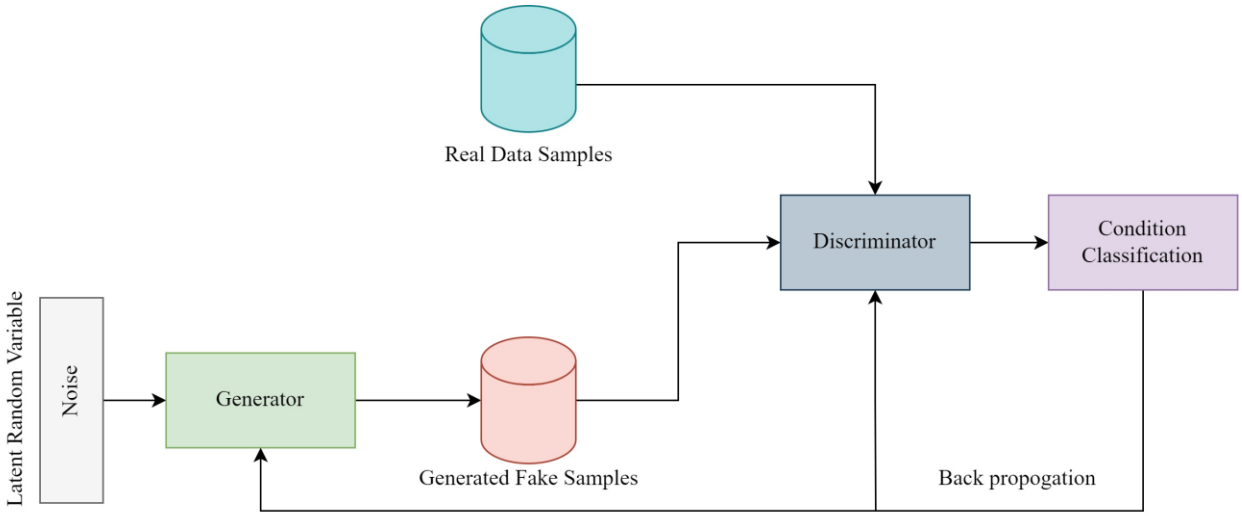

Fig. 3.2. An outline of Generative Adversarial Networks (GANs).

The architecture of a GAN for generating graph or node embeddings typically consists of a generator network and a discriminator network. The generator network takes a random noise vector as input and produces a vector of node embeddings as output. The discriminator network takes a node embedding as input and outputs a binary classification indicating whether the embedding is real or fake. During training, the generator network is trained to produce embeddings that are difficult for the discriminator network to distinguish from real embeddings, while the discriminator network is trained to accurately distinguish between real and fake embeddings.

The objective function of a GAN for generating graph or node embeddings can be defined as follows:

$$\min_{G}\max_{D} V(D,G) = \frac{E(xP_{data})[\log(D(x))]}{E(zP_z)[\log\left(1-D\left(G(z)\right)\right)]} \qquad (1)$$

where G is the generator network, D is the discriminator network, x is a real node embedding sampled from the training data, z is a random noise vector, P_{data} is the distribution of real node embeddings in the training data, and P_z is the distribution of noise vectors. The first term in the objective function encourages the discriminator network to correctly classify real node embeddings as real, while the second term encourages the generator network to produce embeddings that are classified as real by the discriminator network.

GAN uses a generator network to generate node embeddings that preserve the graph topology. The discriminator network is also the same architecture that takes the node embeddings and the adjacency matrix of the graph as input. The objective function of this GAN is similar to the standard GAN objective function, but with additional terms that encourage the generator network to produce embeddings that are structurally similar to the original graph.

GANs provide a powerful framework for generating graph or node embeddings that capture the structural properties and relationships of nodes in a graph. The objective function of a GAN for generating graph or node embeddings encourages the generator network to produce embeddings that are indistinguishable from real embeddings, while the discriminator network is trained to accurately distinguish between real and fake embeddings. After the embeddings are generated from the phase II the third phase proceeds with the embeddings stored in a data frame where the number of rows are the number of nodes in the graph with the column number as size of embeddings dimensions.

3.4 Applying CELFIE algorithm:

CELFIE (Cost Effective Lazy Forward Inference for Influence Estimation) is an extension of the CELF algorithm for influence maximization that aims to reduce the computational cost of evaluating the marginal gain of adding a new node to the seed set. The algorithm maintains an influence likelihood matrix that stores the expected influence of each node in the network given the current seed set. The influence likelihood matrix is computed by simulating multiple independent cascade models for each node in the network and averaging the resulting influence values over all simulations.

The CELFIE algorithm starts by initializing the seed set with a single node that has the highest influence likelihood value in the influence likelihood matrix. Then, it iteratively adds nodes to the seed set in decreasing order of their marginal gain, which is the difference between the influence likelihood value of adding the node to the current seed set and the influence likelihood value of the current seed set. The marginal gain is evaluated lazily using the influence likelihood matrix, which is updated only when necessary to compute the marginal gain of a new node.

The influence likelihood matrix is a square matrix of size n x n, where n is the number of nodes in the network. Each element (i, j) of the matrix represents the expected influence of node j given that node i is in the seed set. The influence likelihood matrix can be computed by simulating multiple independent cascade models starting from each node in the network and averaging the resulting influence values over all simulations. The number of simulations required to obtain an accurate estimate of the influence likelihood values depends on the network topology and the desired level of accuracy.

After generating the embeddings from the GAN architecture, each node is mapped with their respective embeddings. From the above processed adjacent matrix, the top nodes based on their degree are sorted in descending order, and the top S nodes i.e., 100 nodes are stored in S_{nodes}. Now these top nodes with their respective embeddings are stored in S_{emb}. After finding the top 100 nodes remaining nodes are stored in a variable named T_{nodes}, correspondingly their embeddings are stored in T_{emb}. After gathering and storing the values the embeddings and nodes we perform dot product between S_{emb} and T_{emb}.T to calculate ILM. ILM (Influence Likelihood Matrix) is used to efficiently calculate the marginal gain of each node with respect to other nodes in the network. It can be calculated using:

$$ILM = \begin{bmatrix} \emptyset(D_1) \\ \vdots \\ \emptyset(D_I) \end{bmatrix}, D = OT^T + B + B' \qquad (2)$$

Where $\emptyset(.)$ is the SoftMax function, O is the hidden layer matrix (embeddings of source nodes), T is the output layer matrix (embeddings of target nodes), B is a matrix containing the embeddings that capture the ability of each node to influence others (b_u in Eq. (3)) repeated for I rows, where I is the number of nodes that started more than one cascade, and B′ is the embeddings of influence susceptibility repeated for N columns. The SoftMax function from Eq. (4) is applied in every row of the matrix to derive the diffusion probabilities.

The hidden layer of the network represents the source embeddings O of the nodes, and the output layer the target T. The likelihood for a node u to influence v is given by:

$$f_{u,v} = O_u . T_v + b_u + b_{u'} \qquad (3)$$

$$P_{u,v} = \emptyset\left(f_{u,v}\right) = \frac{e^{f_{u,v}}}{\sum_{u \in G} e^{f_{u,v}}} \qquad (4)$$

The key advantage of the CELFIE algorithm is that it reduces the computational cost of influence maximization by avoiding the need to simulate multiple independent cascade models for each candidate node. Instead, the algorithm uses the precomputed influence likelihood matrix to estimate the expected influence of each node, which significantly reduces the computational overhead. The influence likelihood matrix can be updated incrementally as nodes are added to the seed set, which further improves the efficiency of the algorithm.

Once the influence likelihood matrix is computed, the CELFIE algorithm can efficiently evaluate the marginal gain of adding a new node to the seed set by computing the difference between the influence likelihood value of adding the node to the current seed set and the influence likelihood value of the current seed set. The marginal gain of each candidate node is evaluated lazily using the influence likelihood matrix, which is updated only when necessary to compute the marginal gain. The algorithm iteratively adds nodes to the seed set in decreasing order of their marginal gain until the desired number of nodes is selected.

After the Seed Set is returned from the CELFIE algorithm. It is subjected to performance evaluation and various other tests to evaluate the framework performance.

Chapter 4

EXPERIMENTAL PROCEDURE

4.1 Datasets:

For this work we are working on three datasets to conduct the experimentations and evaluate the proposed model with the existing models and further using to three different datasets help us to understand the working of each algorithm in different scenarios and compliant to the real word datasets. The datasets we will be using for this work are Karate Club dataset, Erdős-Rényi random graph generator, and Facebook Politician dataset.

4.1.1 Karate Club edge list 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.

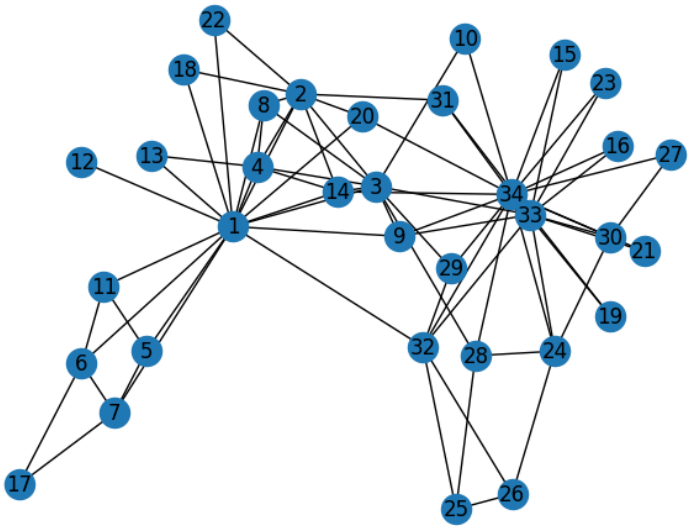

Fig. 4.1. An 2D view of Karate Club network.

4.1.2 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 500 nodes and edge probability of 0.19. We obtained a graph with 500 nodes and 2414 edges.

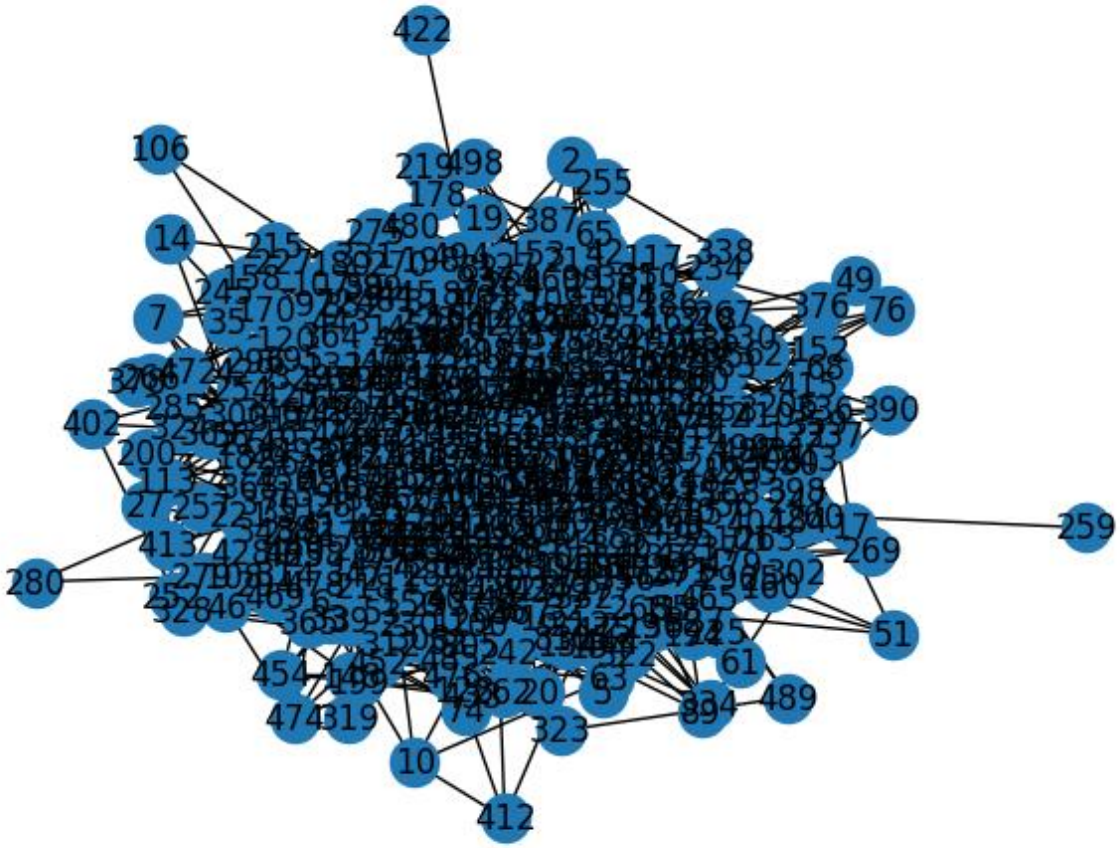

Fig. 4.2. An 2D view of Erdős-Rényi random graph network.

4.1.3 Facebook Politician dataset:

The dataset consists of circle or Facebook friends from Facebook. This data was collected from survey participants using the Facebook app. The dataset includes node features (profiles), circles, and ego networks. Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, while feature vectors from this dataset have been provided, the interpretation of those features has been obscured. Among the large dataset we have utilized the part of it which contains the information regarding the politicians in the space. To anonymize the data, we are provided with a edge list consisting of 5908 nodes and 41729 edges. The graph is a simple undirected graph with edge weight initially taken as 1.

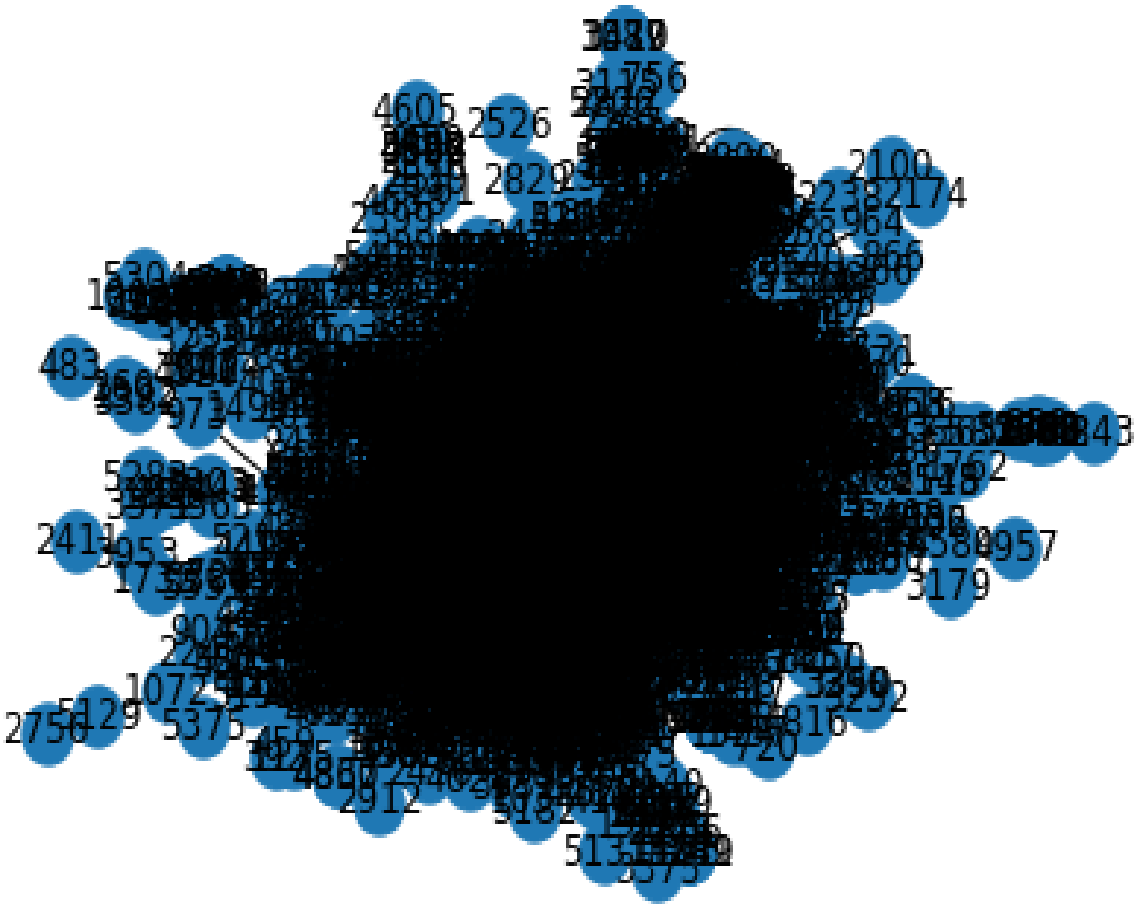

Fig. 4.3. An 2D view of Facebook Politician dataset network.

4.2 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.

4.3 Baseline models:

To benchmark our proposed model, we compare our model with the existing state of art models like CELFIE (Cost-Effective Lazy Forward with Influence Embeddings), CELF++ (Cost-Effective Lazy Forward++) models and the pre-existing well established models like Greedy and CELF (Cost-Effective Lazy Forward). These models are trained on the edge-list of the three datasets and the Seed Set S is calculated. Now the Seed Set S generated from these baseline models is used to calculate the DNI (Distant Nodes Influenced) to calculate the spread of the model. Meanwhile these each algorithms take a certain time to generate the Seed Set S, this Runtime is used to determine the performance of the models. Along with these models, Greedy, CELF, CELF++ algorithms were trained on the embeddings to evaluate the usage of embeddings instead of the edge list as an input for these algorithms.

Chapter 5

RESULTS AND DISCUSSIONS

For evaluating the performance of our proposed model, it is tested on various dataset named Karate Club network dataset, *Erdős-Rényi* random graph generator, and Facebook Politicians dataset. These datasets comprise nearly 34, 500 and 5908 nodes. This experimentation was performed in an AMD Ryzen 4500U processor with a clock speed of 4.1GHz and 8 GB RAM. It was executed in Google Collaboratory IDE and Anaconda Jupiter notebook, and Python 3.9.10 was utilized. To evaluate the performance various models are used as benchmarks. The baseline models used are Greedy, CELF, CELF++, CELFIE were trained on edge lists of the above three 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 and embeddings version of the same models.

Table 5.1. Run Time of various approaches compared with the proposed method.

Approach	**Graph 1**	**Graph 2**	**Graph 3**
Greedy [3]	100 ms	1 min 10sec	---------
CELF [4]	21ms	7.27 sec	20min 49sec
CELF++ [5]	19ms	7.62sec	20min 14sec
CELFIE [1]	49ms	13.3sec	39min 52sec
CELF + Embeddings	13ms	5.52sec	14min 54sec
CELF++ Embeddings	12ms	6.93sec	15min 55sec
CELFIE + Embeddings	38ms	11.8sec	30min 48sec

Table 5.2. DNI (Distinct Nodes Influenced) for various approaches compared with the proposed model

Approach	**Graph 1**	**Graph 2**	**Graph 3**
Greedy [3]	6	484	----------
CELF [4]	30	494	5856
CELF++ [5]	9	496	5889
CELFIE [1]	27	499	5896
Greedy + Embeddings	19	496	----------
CELF + Embeddings	30	498	5896
CELF++ Embeddings	21	498	5901
CELFIE + Embeddings	32	500	5905

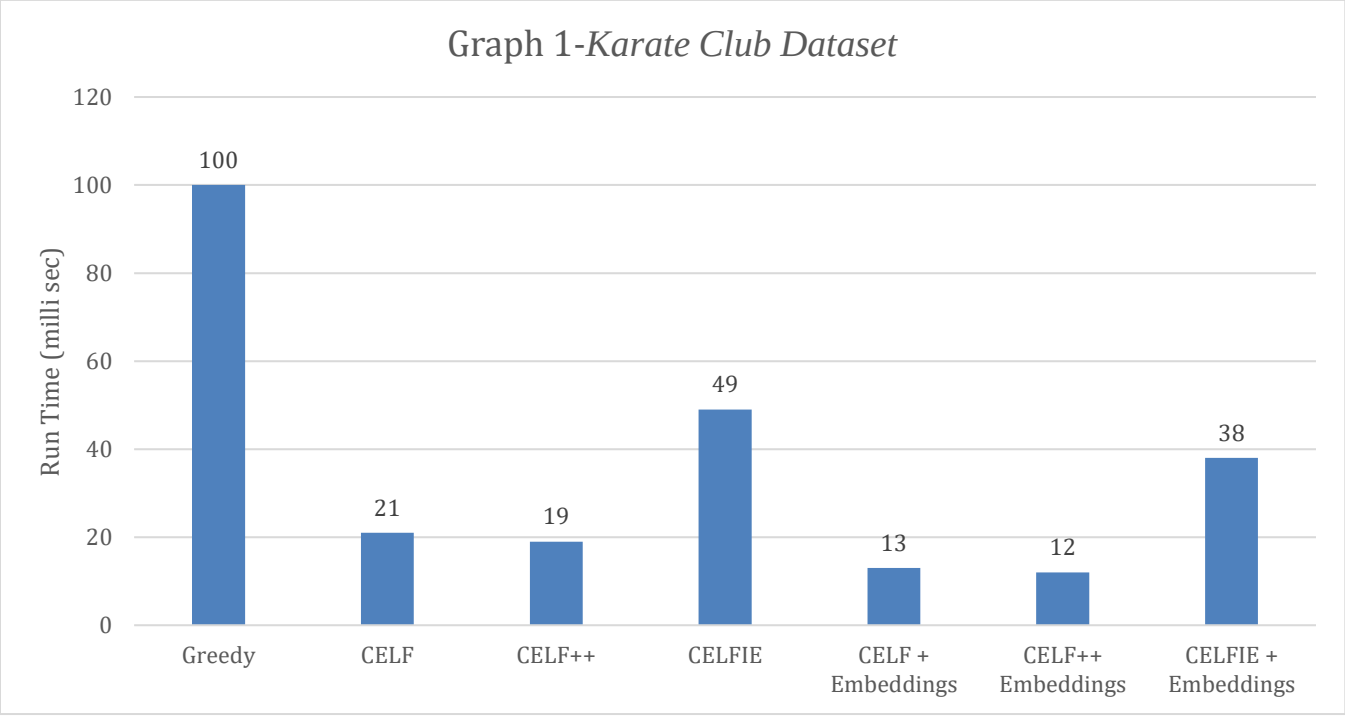

Fig 5.1. Comparison of Run Time in milli-sec on graph 1.

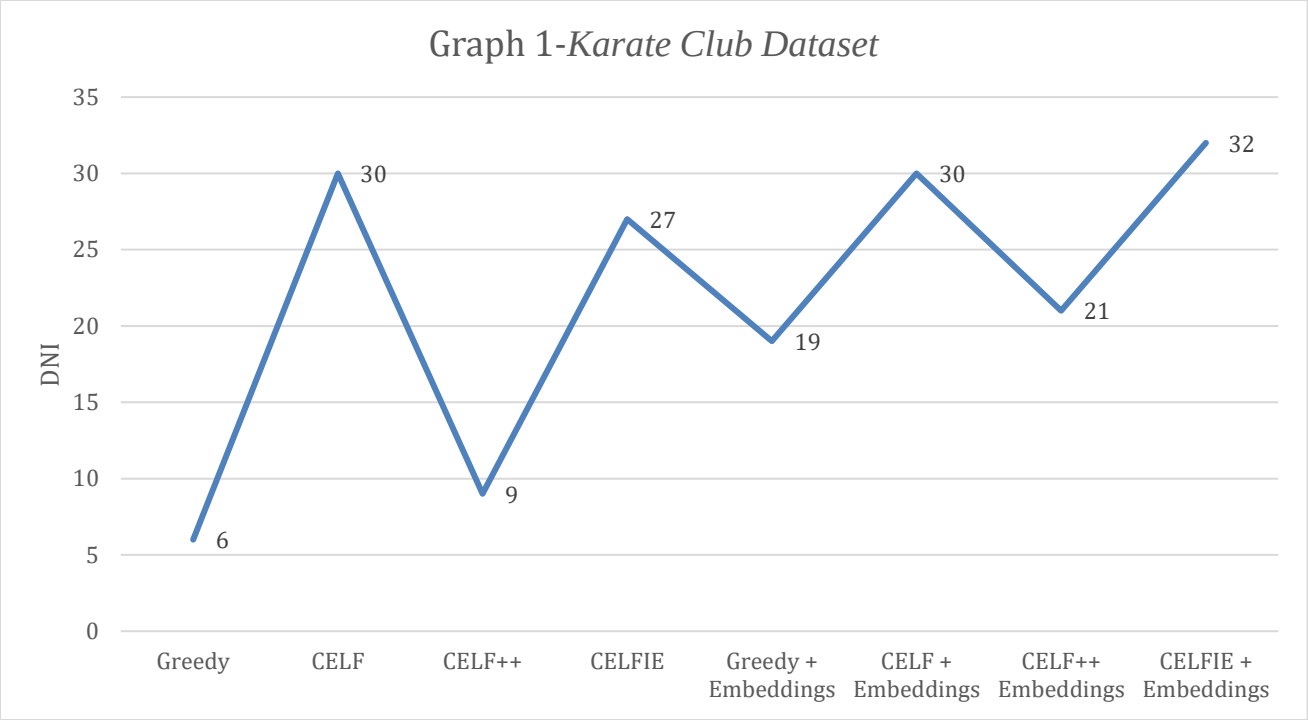

Fig 5.2. Comparison of DNI for various approaches on graph 1.

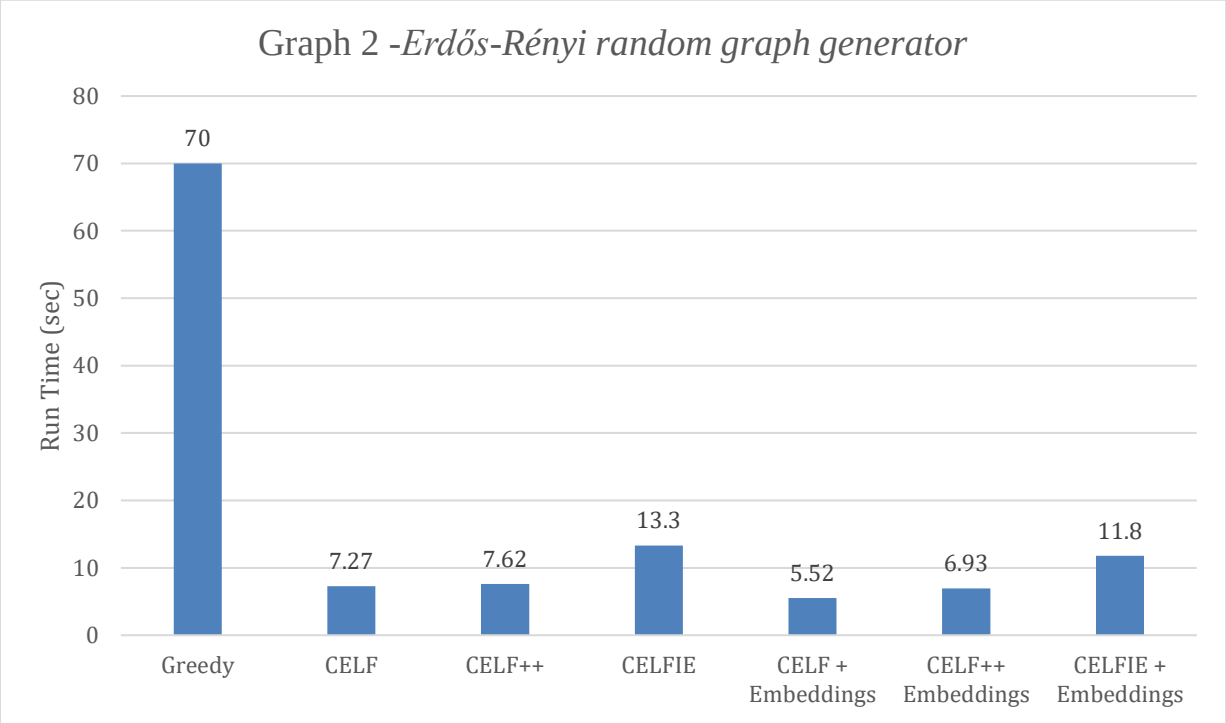

Fig 5.3. Comparison of Run Time in milli-sec on graph 2.

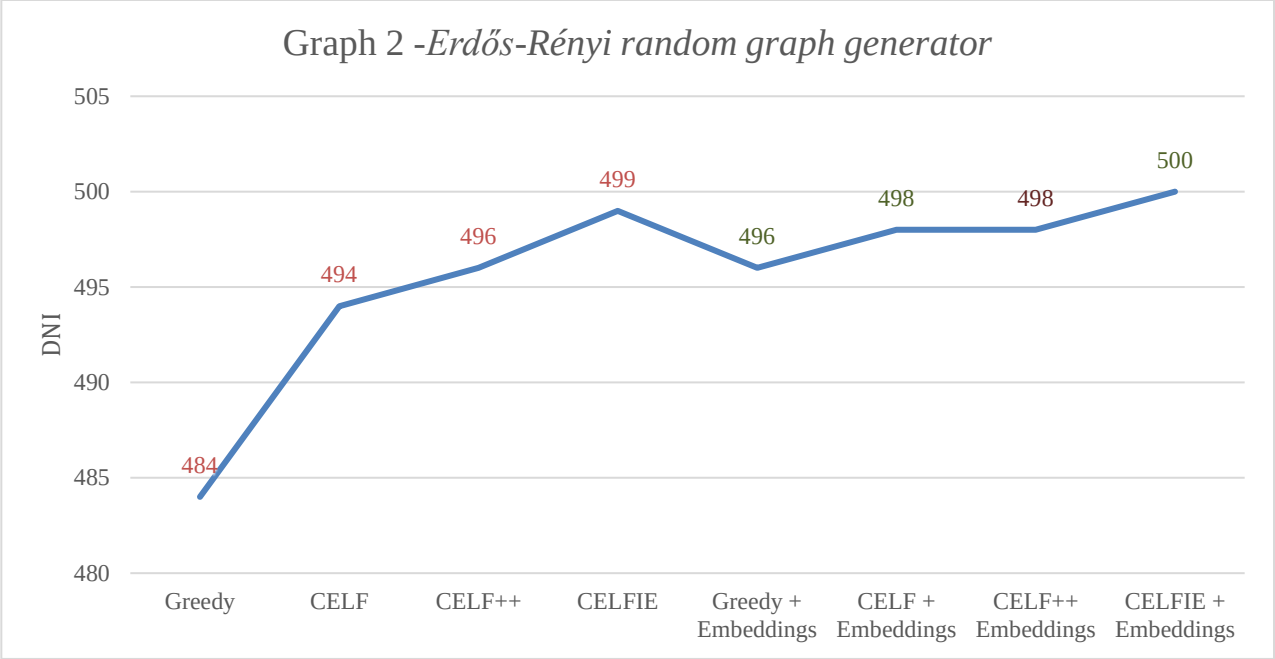

Fig 5.4. Comparison of DNI for various approaches on graph 2.

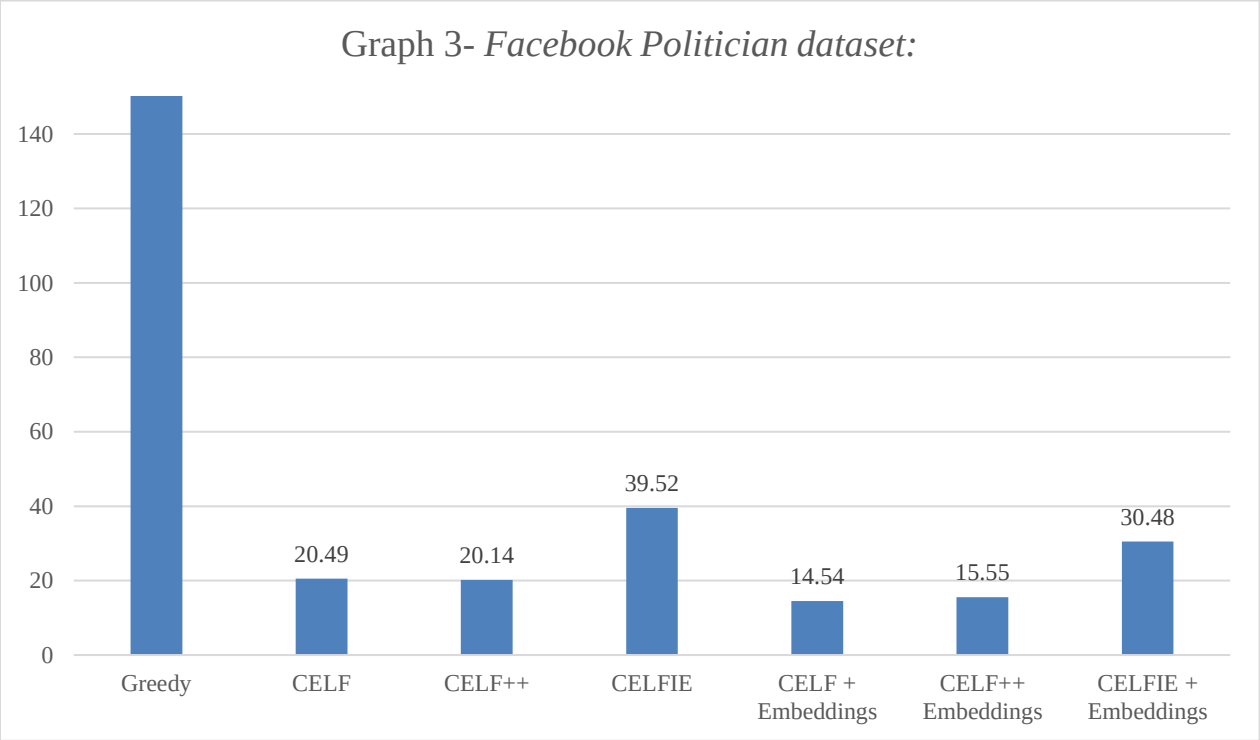

Fig 5.5. Comparison of Run Time in milli-sec on graph 3.

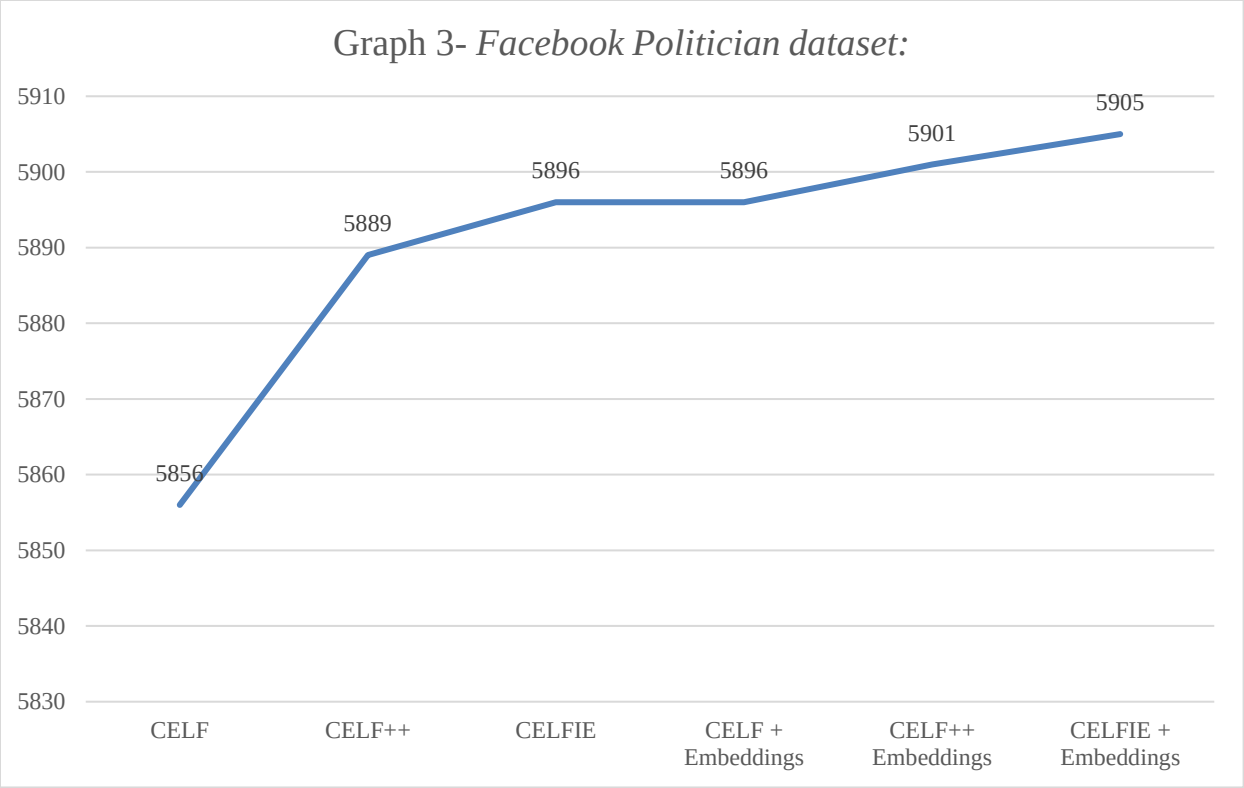

Fig 5.6. Comparison of DNI for various approaches on graph 3.

From the above Table 1 and bar graphs we can observe that the CELFIE algorithm when implemented with embeddings was able to get best results in terms of DNI spread when compared to other models in the framework. Further when the models are implemented using embeddings, they were able to get better results in terms of Run Time. For graph 1 we saw an improvement of 33.14% compared to the edge list implementation on the graph. Similarly, we saw a faster run time for graph 2 and graph 3 by a margin of 17.34% and 25.2% over the existing methodologies.

Similarly, we saw an improvement of 4.9% on the datasets when compared with the existing baseline models, which is a better improvement as the existing models are already state of the art architectures. So, our proposed model may take more than the existing model in some scenarios but the spread or DNI achieved by the proposed model is overwhelming.

Chapter 6

CONCLUSION AND FUTURE SCOPE

In this work, an Influence Maximization technique with Influence and Susceptibility Embeddings is proposed. Finding the nodes which have more influence on the network or the nodes which can influence majority of the network is essential for targeted advertising, viral marketing, and the spread of ideas inside a social network. In this work using embeddings generated from GANs, influence maximization algorithm named CELFIE can leverage the power of deep learning to identify important nodes that may be missed by traditional graph-based methods. GAN-generated embeddings can capture complex relationships between nodes and can identify patterns that may not be apparent from the network structure alone. This can lead to more effective influence maximization and can help identify distant nodes influenced, which can amplify the spread of influence. After performing the experimentations, the proposed model was able to we were able to achieve 23% faster results in performing the influence maximization task and 4.9% further improvement in the DNI over the state-of-the-art models. And we can clearly observe using the embeddings rather than edge list to select the seed nodes have better performance in terms of Seed Set quality like DNI and faster computation.

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. However, their potential application in influence maximization remains relatively unexplored. The integration of GANs with the CELFIE algorithm may offer a powerful approach to solving the influence maximization problem with improved accuracy and efficiency. Moreover, future research could investigate the use of other generative models, such as variational autoencoders and flow-based models, for generating graph embeddings. Additionally, the incorporation of other information, such as textual and visual data, into the embedding generation process may further enhance the accuracy of influence

maximization. Overall, the future scope of influence maximization using GANs is vast and has the potential to advance the field of social network analysis.

Furthermore, the application of GANs for influence maximization also raises ethical considerations, particularly in terms of the potential manipulation of social networks for commercial or political gain. As such, future research should also explore the ethical implications of using GANs for influence maximization, and develop methods to ensure the responsible and transparent use of these technologies. Additionally, the scalability and generalizability of GAN-based influence maximization models must be addressed, as larger and more complex social networks may require significant computational resources to generate accurate embeddings. In summary, while the integration of GANs with influence maximization algorithms is still in its infancy, it holds significant potential for enhancing the accuracy and efficiency of influence maximization, and could pave the way for new and innovative applications in the field of social network analysis.

References

[1]. Panagopoulos, G., Malliaros, F. D., & Vazirgianis, M. (2020, May). Influence maximization using influence and susceptibility embeddings. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 14, pp. 511-521).

[2]. Panagopoulos, G., Malliaros, F., & Vazirgiannis, M. (2020). Multi-task learning for influence estimation and maximization. *IEEE Transactions on Knowledge and Data Engineering*.

[3]. Kempe D., Kleinberg, J., & 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).

[4]. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007, August). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 420-429).

[5]. Goyal, A., Lu, W., & 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).

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

[7]. Wu, G., Gao, X., Yan, G., & 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.

[8]. Feng, C., Fu, L., Jiang, B., Zhang, H., Wang, X., Tang, F., & Chen, G. (2020). Neighborhood

matters: Influence maximization in social networks with limited access. *IEEE Transactions on Knowledge and Data Engineering*.

[9]. Arora, A., Galhotra, S., & 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).

[10]. Singer, Y. (2016). Influence maximization through adaptive seeding. *ACM SIGecom Exchanges*, *15*(1), 32-59.

[11]. Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web* (WWW '15 Companion). ACM, New York, NY, USA, 243-246. DOI=http://dx.doi.org/10.1145/2740908.2742839.

[12]. Zhang, J., Liu, B., Tang, J., Chen, T., & Li, J. (2013, August). Social influence locality for modeling retweeting behaviors. In *IJCAI* (Vol. 13, pp. 2761-2767).

[13]. Aral, S., and Dhillon, P. S. 2018. Social influence maximization under empirical influence models. Nature HumanBehaviour2(6):375.Aral, S., and Walker, D. 2012. Identifying influential and susceptible members of social networks. Science 337(6092):337–341.

[14]. Bourigault, S.; Lamprier, S.; and Gallinari, P. 2016. Representation learning for information diffusion through social networks: an embedded cascade model. In9th ACM International Conference on Web Search and Data Mining, 573–582. ACM.

[15]. Budak, C.; Agrawal, D.; and El Abbadi, A. 2011. Limiting the spread of misinformation in social networks. In20th international conference on World wide web, 665–674. ACM.

[16]. Chen, W.; Wang, C.; and Wang, Y. 2010a. Scalable influence maximization for prevalent

viral marketing in large-scale social networks. In16th ACM SIGKDD international conference on Knowledge discovery and data mining, 1029–1038.ACM.

[17]. Chen, W.; Wang, C.; and Wang, Y. 2010b. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and datamining, 1029–1038. ACM.

[18]. Cohen, E.; Delling, D.; Pajor, T.; and Werneck, R. F.2014. Sketch-based influence maximization and computa-tion: Scaling up with guarantees. InProceedings of the 23rdACM International Conference on Conference on Informa-tion and Knowledge Management, 629–638. ACM.

[19]. Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web* (WWW '15 Companion). ACM, New York, NY, USA, 243-246. DOI=http://dx.doi.org/10.1145/2740908.2742839.

[20]. Zhang, J., Liu, B., Tang, J., Chen, T., & Li, J. (2013, August). Social influence locality for modeling retweeting behaviors. In *IJCAI* (Vol. 13, pp. 2761-2767).

[21]. Aral, S., and Dhillon, P. S. 2018. Social influence maximization under empirical influence models. Nature HumanBehaviour2(6):375.Aral, S., and Walker, D. 2012. Identifying influential and susceptible members of social networks. Science 337(6092):337–341.

[22]. Bourigault, S.; Lamprier, S.; and Gallinari, P. 2016. Representation learning for information diffusion through social networks: an embedded cascade model. In9th ACM Inter-national Conference on Web Search and Data Mining, 573–582. ACM.

[23]. Budak, C.; Agrawal, D.; and El Abbadi, A. 2011. Limiting the spread of misinformation in social networks. In20th international conference on World wide web, 665–674. ACM.

[24]. G. Tong, W. Wu, S. Tang, D.Z. Du, Adaptive influence maximization in dynamic social networks, IEEE ACM Trans. Netw. 25 (1) (2017) 112–125, http://dx.doi.org/10.1109/TNET.2016.2563397.

[25]. D. Li, C. Wang, S. Zhang, G. Zhou, D. Chu, C. Wu, Positive influence maximization in signed social networks based on simulated annealing, Neurocomputing 260 (2017) 69–78, http://dx.doi.org/10.1016/j.neucom. 2017.03.003.

[26]. W. Liu, Y. Li, X. Chen, J. He, Maximum likelihood-based influence maximization in social networks, Appl. Intell. 50 (2020) 3487–3502, http: //dx.doi.org/10.1007/s10489-020-01747-8.

[27]. E. Mossel, G. Schoenebeck, Reaching consensus on social networks, in: Proceedings of the the First Symposium on Innovations in Computer Science, 2009, pp. 214–229, https://repository.upenn.edu/statistics_papers/ 166.

[28]. F. Wang, E. Camacho, K. Xu, Positive influence dominating set in online social networks, in: International Conference on Combinatorial Optimization and Applications, Springer, 2009, pp. 313–321, http://dx.doi.org/10. 1007/978-3-642-02026-1_29.

[29]. Y. Li, W. Chen, Y. Wang, Z.L. Zhang, Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, 2013, pp. 657–666, http://dx.doi.org/10.1145/ 2433396.2433478. 19 M. Azaouzi, W. Mnasri and L. Ben Romdhane Computer Science Review 40 (2021) 100393

[30]. X. He, G. Song, W. Chen, Q. Jiang, Influence blocking maximization in social networks under the competitive linear threshold model, in: Proceedings of the 2012 Siam International Conference on Data Mining, SIAM, 2012, pp. 463–474, http://dx.doi.org/10.1137/1.9781611972825.40.

[31]. H. Kim, K. Beznosov, E. Yoneki, A study on the influential neighbors to maximize information diffusion in online social networks, Comput. Soc. Netw. 2 (1) (2015) 1–15, http://dx.doi.org/10.1186/s40649-015-0013-8.

[32]. W. Liu, K. Yue, H. Wu, J. Li, D. Liu, D. Tang, Containment of competitive influence spread in social networks, Knowl.-Based Syst. 109 (C) (2016) 266–275, http://dx.doi.org/10.1016/j.knosys.2016.07.008.

[33]. F. Wang, G. Wang, D. Xie, Maximizing the spread of positive influence under LT-MLA model, in: Asia-Pacific Services Computing Conference, Springer, 2016, pp. 450–463, http://dx.doi.org/10.1007/978-3-319-49178- 3_34.

[34]. [40] J. Sheng, L. Chen, Y. Chen, B. Li, W. Liu, Positive influence maximization in signed social networks under independent cascade model, Soft Comput. 24 (19) (2020) 14287–14303, http://dx.doi.org/10.1007/s00500-020- 05195-x.

[35]. A. Goyal, W. Lu, L.V. Lakshmanan, Celf++ optimizing the greedy algorithm for influence maximization in social networks, in: Proceedings of the 20th International Conference Companion on World Wide Web, 2011, pp. 47–48, http://dx.doi.org/10.1145/1963192.1963217.

[36]. [42] Y. Wang, G. Cong, G. Song, K. Xie, Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 1039–1048, http://dx.doi.org/10. 1145/1835804.1835935.

[37]. 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, http://dx.doi.org/10.1109/ICDM.2011.132.

[38]. E. Cohen, D. Delling, T. Pajor, R.F. Werneck, Sketch-based influence maximization and computation: Scaling up with guarantees, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014, pp. 629–638, http://dx.doi.org/ 10.1145/2661829.2662077.

[39]. [45] D. Kim, D. Hyeon, J. Oh, W.S. Han, H. Yu, Influence maximization based on reachability sketches in dynamic graphs, Inform. Sci. 394 (2017) 217–231, http://dx.doi.org/10.1016/j.ins.2017.02.023.

[40]. T.W. Valente, Social network thresholds in the diffusion of innovations, Soc. Netw. 18 (1) (1996) 69–89, http://dx.doi.org/10.1016/0378-8733(95) 00256-1.

[41]. N. Chen, On the approximability of influence in social networks, SIAM J. Discrete Math. 23 (3) (2009) 1400–1415, http://dx.doi.org/10.1137/ 08073617X.

[42]. M. Doo, L. Liu, Probabilistic diffusion of social influence with incentives, IEEE Trans. Serv. Comput. 7 (3) (2014) 387–400, http://dx.doi.org/10.1109/ TSC.2014.2310216.

[43]. 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.

ACKNOWLEDGEMENTS

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 my sincere gratitude to our project guide **Dr. Nagesh Bhattu Sristy**, 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.

Maddineni Saketh	Kalakonda Viswa Sahithi	Gonda Vamshi
411947	411936	411925
Date :	Date :	Date :