Self-Aware Fairness 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

April 2024

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: May 4, 2024

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH

Certificate

This is to certify the thesis entitled "Self-Aware Fair 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: 26/04/2024

PROJECT WORK APPROVAL

This project work entitled ”**Self-Aware Fairness Influence Maximization**” was worked out by Seedella Sai Vikas (420233), Vedagiri Krishna Sandeep (420247) and Matta Rahul (420206) is approved for the degree of Bachelor of Technology in Computer Science and Engineering

Examiners

____

____

____

Supervisor (s)

____

____

____

Chairman

____

Date:

Place: Tadepalligudem

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 fair influence maximization between communities using Generative Adversarial Network (GAN) based on sensitive attributes like gender, age, ethnicity etc. We are using the optimized version's of Greedy algorithms like CELF and CELF++ for influence maximization.

We propose a novel method called Adversarial Graph Embeddings for Fair Influence Maximization. Our method integrates adversarial learning with graph embeddings to optimize influence spread while ensuring fairness across different demographic groups. Leveraging graph embedding techniques, this learns representations that capture both structural properties and node attributes, enabling effective influence prediction. Additionally, the adversarial component of our framework minimizes the discrepancy in influence spread between privileged and underprivileged groups, promoting fairness in the influence maximization process. We evaluate this model on real-world social network datasets, demonstrating its effectiveness in achieving competitive influence spread while simultaneously enhancing fairness compared to existing methods. Overall, our framework offers a principled approach to address fairness concerns in influence maximization tasks over social networks, contributing towards more equitable and inclusive societal influence dynamics.

In this project we used KMeans clustering, CELF, CELF++, CELFIE Algorithms on graph embeddings which are novel approaches and compared their influence spreads which is used in original paper.

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 analytics (SNA) is a fast growing field that seeks to understand and model the complex relationships between individuals in a given network.
- The key challenges in social network analytics is influence maximization, which involves finding a small set of individuals in a given graph who could have the greatest effect on the behavior of the rest of the given network.
- The problem of Influence Maximization has important uses in fields such as advertising, public health, and social media.
- Influence Maximization (IM) involves pinpointing a cluster of pivotal nodes capable of amplifying the diffusion of a specific behavior, concept, or product throughout the network.
- Finding these nodes is essential for targeted advertising, viral marketing, and the influence spread of information inside a given social network.

1.2 Problem Statement

The aim is to pinpoint a select group of influential figures within a social network, individuals capable of significantly shaping the actions of others and enhancing the propagation of influence. This endeavor also seeks to ensure equitable representation across diverse network segments, employing adversarial networks and autoencoder models to harness their strengths effectively.

1.3 Influence Maximization

With the exponential surge in global population, social media has emerged as an indispensable platform for enterprises and entities to showcase their products, aiming for extensive outreach and heightened popularity among consumers. The ripple effect of social media, commonly known as word-of-mouth, plays a pivotal role in influencing others' decisions. Many companies resort to strategic marketing maneuvers on social networks, such as conducting influence campaigns and distributing complimentary products to influential individuals (known as seed nodes), in anticipation of their ability to advocate for the products among their social circles. Moreover, certain entities opt to enlist celebrity brand ambassadors, leveraging their widespread popularity to endorse their offerings, thus enticing their followers to make purchases.

Social networks, intricate webs of user relationships, have thus become a focal point for extensive research endeavors by numerous entities and scholars, all aiming to refine and augment the accuracy and other metrics pertinent to the Influence Maximization conundrum (IM). In this study, our focus is specifically on a crucial aspect: identifying a concise seed nodes of set k within a given network, with objective of maximizing their spread propagation—measured by estimated sum of activated nodes post-seed node activation—under specific propagation models.

General graph networks will be present in the following form:

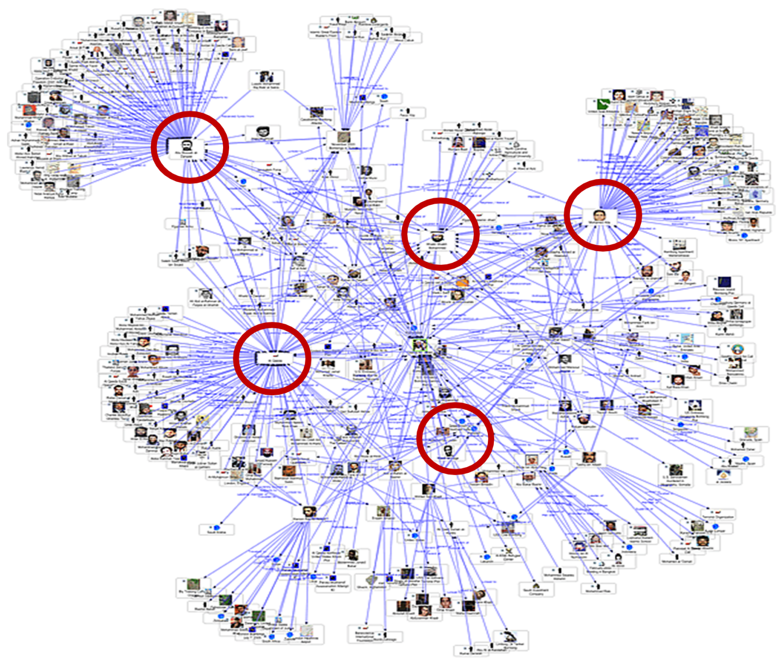


Figure 1.1: General graph networks

1.3.1 Influence Maximization with Fairness

In this work we concentrate on fairness among the influenced nodes of different categories(majority and minority).Fair Influence maximization was first introduced in 2019 and it is a problem of finding k set of influential seed nodes that maximizes the spread of influence and make sure fairness is included in the influenced nodes based on the sensitive attributes. This is a case where there is homophily in the given network, In our work, we concentrate on balancing Influence maximization.
Example : 2 groups (majority , minority)

We have found that greedy algorithm alone without embeddings fail 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 Embedding Generation for Fair Influence Maximization

In this work we focused on generating embeddings which are used to find seed nodes in the network which give maximimum influence and also maintaining categorical ratio between different communities which are formed based on different sensitive attributes. We are first using auto encoder for generating pre-embeddings and then train them using Generative Adversarial Networks (GAN) for fair distribution of nodes in the embedding space.

In pursuit of crafting optimal adversarial network embeddings for a binary community-based input network, we devise an innovative adversarial framework employing Generative Adversarial Networks (GANs). Within this framework, an embedder engages in a strategic confrontation with a discriminator, aimed at achieving desired network embeddings. Subsequently, we employ various methodologies to find the most influential set of k nodes within the network.

1.3.3 Different Methods for Choosing Seed Nodes

After generating the embeddings, we use different approaches to find seed nodes. We in this research used spectral Clustering, CELF, CELF++, CELFIE on the embeddings and compared with standard K-Means algorithm.

The computational aspect of the Greedy algorithm is widely recognized as a foundational method for pinpointing influential nodes within 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.

Determining the k-most influential nodes poses a formidable challenge, classified as an NP-Hard problem due to the necessity of selecting the optimal subset of seed nodes. This task entails exponential time complexity, adding to its complexity.

Chapter 2

Literature Review

Kempe et al. established the NP-hardness of the influence maximization problem, yet provided a ray of hope by showcasing that a simple greedy algorithm could yield the best feasible approximation factor within polynomial time. Building upon this foundation, Chen et al. introduced the CELF algorithm in 2009, leveraging the submodularity property to devise an efficient solution capable of tackling larger problems. This breakthrough led to the development of the CELF++ algorithm by Goyal et al., an enhancement of its predecessor that integrates a priority queue for optimized runtime and expedited computation.

Azaouzi et al. undertook a comprehensive survey exploring emerging trends in influence maximization models. Their study categorized diffusion models into two distinct groups: individual-based and group-based models. Through meticulous analysis, they juxtaposed and evaluated various models within each category, shedding light on their respective strengths and limitations.

Panagopoulos et al. introduced an innovative approach to influence maximization known as CELFIE, which leverages learned influence representations extracted from diffusion cascades. This method bypasses the need for traditional diffusion models, resulting in superior seed set quality and faster execution. Additionally, their IMINFECTOR method utilizes representations derived from diffusion cascades for model-independent influence maximization, surpassing baseline models in both scalability and accuracy.

G. Wu et al. presented a novel perspective on influence maximization, tackling the challenge of multiple information propagation within a single network, each with distinct propagation probabilities. Their proposed greedy framework offers an approximation ratio of 1/3, while parallel algorithms with semaphores enhance efficiency and reduce runtime. Meanwhile, C. Feng et al. addressed the limitations of influence maximization in real-world scenarios, particularly regarding user privacy concerns. They introduced an adaptive approach integrating seeding and diffusion uncertainty to navigate this challenge effectively.

A. Aurora et al. conducted a comprehensive benchmarking study focusing on influence maximization techniques within social networks. Their innovative benchmarking platform facilitates systematic and thorough evaluation and comparison of existing techniques under uniform experimental conditions. Additionally, Singer Y proposed an alternative strategy to address the dissemination of information among influential users within social networks. This adaptive approach strategically targets users based on the influence of their neighbors, aiming for enhanced information propagation.

In the landscape of social network analysis, the pursuit of identifying influential nodes remains paramount for understanding information diffusion and strategic interventions. Traditional metrics like degree centrality and betweenness centrality have long been employed but often fail to capture the nuanced contributions of individual nodes. ”A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks” introduces a novel framework leveraging Shapley values from cooperative game theory to address this gap. By treating influence spread as a cooperative game and quantifying the marginal contribution of each node, the method offers a more nuanced understanding of node influence dynamics. Empirical evaluations on real-world networks demonstrate its superiority over traditional centrality measures, highlighting its potential for enhancing strategic decision-making in network interventions. The paper also addresses computational complexity concerns and proposes efficient algorithms to compute Shapley values, further solidifying its applicability in real-world scenarios.

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.

In a social network context, the finite set symbolizes the users or nodes present within the network. The function assigned to this set assigns a numerical value to each subset of nodes, indicating the extent of influence spread attained by selecting that particular subset as seed nodes.

$$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \quad \forall A \subseteq B \tag{3.1}$$

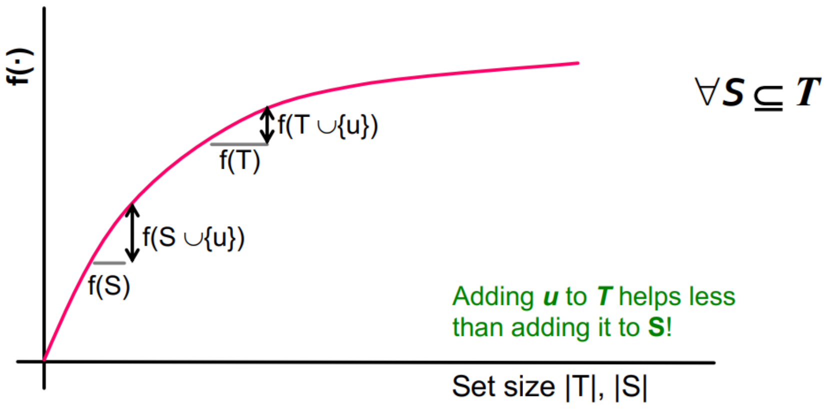


Figure 3.1: Diminishing returns.

The diminishing returns property inherent in submodular functions becomes evident as additional nodes are added to the seed set, resulting in diminishing incremental benefits as the seed set expands. Researchers leverage this property to devise approximation algorithms for influence maximization, striking a balance between computational efficiency and maximizing influence spread within social networks. These algorithms exploit the submodularity of the influence function to efficiently identify near-optimal solutions, thereby facilitating the development of generalized frameworks for approximating the spread of influence in diffusion models.

3.2 Homophily

Homophily describes the inclination of individuals to gravitate towards others who share similar traits or characteristics.(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 edges present in the given network and actual is the sum of all the edges present in between groups

Chapter 4

Diffusion Models

Diffusion models serve as analytical and computational tools aimed at exploring the dynamics of information, innovations, behaviors, or influence propagation within networks or populations across time. These models elucidate the mechanisms by which information spreads within and between the given network. Within the domain of information dissemination or idea propagation in a directed graph *G*, nodes are commonly classified as either active (having embraced the idea) or inactive (yet to embrace it).
There are two basic diffusion models:

1. Linear Threshold Model
2. Independent Cascade Model

4.1 Linear Threshold Model

Linear Threshold Model (LTM) offers insights into how influence disseminates throughout a given network. Operating on the principle of node activation, it determines the spread of influence by assessing nodes' cumulative influence from their network connections and individual activation thresholds. Seed node selection is pivotal in influencing the scope of influence within the network.

In this model, a given network is depicted as a social graph wherein nodes (or also take it as users) are interconnected by the edges, symbolizing their relationships, with every used given a random threshold value between from (0,1). These thresholds denote the minimum influence required for node activation. The initial node selection is based on their perceived potential to influence others, initiating the influence propagation process.

Input: Graph G, Seed set S, k
Output: Activated nodes set in the graph

Algorithm 1 Linear Threshold Model

```
1: Establish a significantly high value for the number of iterations, denoted as $T$.
2: Assign a distinct activation probability $p_u$ to each node $u$ within the set $V$.
3: for $t = 1$ to $T$ do
4:     Activate all nodes within the set $S$.
5:     Begin with the set of activated nodes $A$ being initialized as $S$.
6:     while $A$ is not empty do
7:         Randomly select a node $v$ from the set $A$.
8:         Calculate the activation weight $w_v$ by summing the weights of all neighboring nodes of
   $v$ that are already activated in set $A$.
9:     end while
10:    Store all the activated nodes for this iteration, denoted by $\sigma_t$.
11: end for
12: Find influence spread for each and every node $u$ as average number of the activated nodes over
   $T$ iterations when $u$ is in the seed set, denoted by $\sigma_u$.
13: Initialize the set of selected nodes $U$ to $S$.
14: while $|U| < k$ do
15:     Choose node $u$ that has the highest influence spread $\sigma_u$ among nodes in the $V \setminus U$.
16:     Add $u$ to $U$.
17: end while
18: 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. v is a node that has random threshold $\Theta_v \in [0,1]$
ii. v cab be influenced because of its neighbour who has the weight $b_{v,w}$ such that

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

iii. The node is actived when at least (weighted) Θ_v some of its neighbours are believed to be active

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

Thus, the activation process commences with a selection of seed nodes that are initially activated. Subsequently, at each time step, nodes exhibiting a significant level of influence undergo activation and can subsequently activate their neighboring nodes. This iterative process persists until no further nodes can be activated. The propagation of influence persists through successive iterations, triggering a cascading effect. As influence spreads, additional nodes transition into an active state. The process culminates when no further changes in activation states occur. The resulting set of active nodes signifies the outcome of the influence maximization endeavor.

4.2 Independent Cascade Model

Independent Cascade Model(also known as ICM) serves as a fundamental framework for gauging the scope of influence dissemination stemming from initial seed nodes. This model plays a crucial role in understanding the viral diffusion of information or trends within social networks. By providing a robust analytical tool, it empowers researchers to delve into the intricacies of information propagation and influence maximization within network dynamics. Moreover, it enables the simulation of influence spread across social graphs, offering valuable insights into network theory.
Input: Seed Set S, k, Graph G.
Output: Set of k nodes to maximize influence spread.

Algorithm 2 Independent Cascade Model Algorithm

```
    Input: Seed Set $S$, Graph $G$
    Output: Activated Nodes $A$
 1: Initialize set of activated nodes $A$ to $S$.
 2: Initialize set of newly activated nodes $NewlyActivated$ to $S$.
 3: while $NewlyActivated$ has something(i.e, not empty) do
 4:     Initialize an set $Temp$ that is empty.
 5:     for every node $v$ in $NewlyActivated$ do
 6:         for every neighbor $u$ of $v$ not in $A$ do
 7:             Initialize a random number which is between $r \in [0,1]$.
 8:             if $r \leq$ activation probability of edge $(u,v)$ then
 9:                 Add the $u$ to $Temp$.
10:             end if
11:         end for
12:     end for
13:     Add the $Temp$ to $A$.
14:     Set $NewlyActivated$ to $Temp$.
15: end while
16: 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 finite graph that is directed, G = (V, E)
Select a set of seed nodes at inital and represent it as S. These nodes are initially activated, This can be done randomly, or by selecting a specific set of nodes that are thought to be influential.

Every edge (v,w) will have a probability (weight) P_{vw}.
If it is said to be node 'v' as active, then the node 'v' can get a single chance to make the node 'w' active, with the probability P_{vw}.
P_{vw} is Probability that node v can activate node w.

- In every time step, every activated user or node will have a one time chance to activate one of the inactive neighbours. For a successful activation the probability is given by a propagation probability, which is typically said to be exactly same for all the edges in the given network.
- Each time a node is activated, it will be remained activated forever. Inactive nodes that are successfully activated become active from next time step.
- The activation process ends when there are no new nodes left that can be activated.
- The resulting collection of activated nodes serves as the culmination of the influence maximization process.

In particular, Independent Cascade Model serves as a crucial instrument for comprehending and forecasting the dissemination of information and influence within social networks. Through strategic selection of initial seed nodes and meticulous evaluation of influence transmission probabilities, we can dissect the dynamics of influence propagation. The overarching objective is to pinpoint optimal strategies for amplifying the reach and efficacy of a message, idea, or behavior within a network. Utilizing the knowledge acquired from this model can inform tangible strategies in fields like viral marketing, public health campaigns, and other initiatives where cultivating influence is of utmost significance.

4.2.1 Working of Influence Approximation Algorithm

The Influence Approximation Algorithm is designed to efficiently estimate influence spread within a network, particularly in situations where computing exact influence proves to be computationally challenging. In general, the algorithm progresses by simulating the spread of influence from a predetermined set of initial seed nodes to their adjacent nodes, utilizing probabilistic frameworks such as the Independent Cascade Model or the Linear Threshold Model. Through iterative updates based on these simulations' outcomes, the algorithm progressively hones its predictions of influence spread until meeting a convergence criterion, such as stabilizing estimates or reaching a predefined threshold. Its primary advantage lies in its ability to furnish reasonably accurate influence spread estimates while markedly reducing computational burden compared to exact computation methods.

Nevertheless, this efficiency may come at the cost of introducing approximation errors, especially in large-scale networks or when utilizing simplified influence models. Despite this trade-off, the algorithm's efficiency renders it a valuable tool for swiftly evaluating different seed node selections' potential impact in influence maximization tasks, empowering decision-makers to make informed choices in settings with limited computational resources.

Chapter 5

Methodologies

The goal in fair influence maximization is to achieve a balance. We want to maximize the overall number of people influenced by information diffusion, but also ensure that this influence reaches a similar proportion of individuals from both Group A and Group B (let's call them the "in-group" and "out-group" for now). This approach aims to prevent scenarios where one group benefits significantly more from the information campaign compared to the other.

5.1 Embedding Generation

5.1.1 Auto Encoder: Generating Pre Embeddings

Autoencoders, a class of artificial neural networks utilized primarily in unsupervised learning, excel at learning efficient representations of input data without the need for labeled examples. The fundamental principle behind autoencoders lies in their ability to compress and represent input data effectively, even in the absence of explicit labels. At the heart of this approach lies a two-step process: an encoder and a decoder. The encoder acts like a data simplifier, taking the initial information and condensing it into a more compact form, often called the "latent space" or "code." This code captures the key characteristics of the data. The decoder then takes this condensed code and performs the opposite task - it rebuilds the original data based on the essential features it learned from the code. Through this back-and-forth process of simplification and reconstruction, the system progressively refines its understanding of the data, uncovering the underlying patterns and significant elements.

5.1.1.1 Architecture of Auto Encoder Model

The autoencoder neural network model is designed to learn graph embeddings that capture both the structural attributes and node features of a social network. The model architecture is structured as follows:

Encoder Network: Encoder is a sequential model, which contains 3 layers

1. **Input Layer:** Receives the graph data, including node features and network structure, utilizing the ReLU activation function.

2. **Hidden Layers:** Consists of one hidden layer that gradually transforms the input into a lower-dimensional representation with an embedding size multiplied by 2. This layer also employs the ReLU activation function.

3. **Output Layer:** Produces learned embeddings representing each node in the graph in a lower-dimensional space, aiming to capture meaningful information about node relationships within the network. ReLU activation function is applied here.

Decoder Network:

1. **Input Layer:** Takes the learned embeddings produced by the encoder, utilizing the ReLU activation function.

2. **Hidden Layers:** Similar to the encoder, the decoder contains hidden layers that progressively reconstructs the input graph data that is original from the learned embeddings. ReLU activation function is utilized.

3. **Output Layer:** The Aim to construct again the input graph data that is original, including node features and network structure, employing the sigmoid activation function.

4. **Loss Function:** For the autoencoder model the loss function is the ”Reconstruction Loss”. This function measures disparity between input graph data and the reconstructed data, which is minimized during training to encourage the model to learn meaningful representations of the input graph.

5. **Training Procedure:** The autoencoder model is trained using graph data from the social network. Throughout training, the given model iteratively changes its parameters to reduce the reconstruction loss/error between input and reconstructed data. The Adam optimizer algorithm is utilized for backpropagation.

Through the utilization of the autoencoder network model, we have generated pre-embeddings that capture diverse structural and attribute information of the social network, offering a comprehensive representation of the network topology and node characteristics.

5.1.2 GAN: Generating FairEmbeddings

The Generative Adversarial Networks (GANs) have demonstrated effectiveness across various data modalities, including images, text, and audio. Recently, GANs have also been leveraged for generating graph or node embeddings, which serve as compact representations of graph nodes, encapsulating both structural features and relational attributes within the network. These embeddings, produced by GANs, uphold the underlying graph topology and find application in tasks like node classification, link prediction, and community detection. Within the GAN framework, two core components exist: the Generator and the Discriminator.

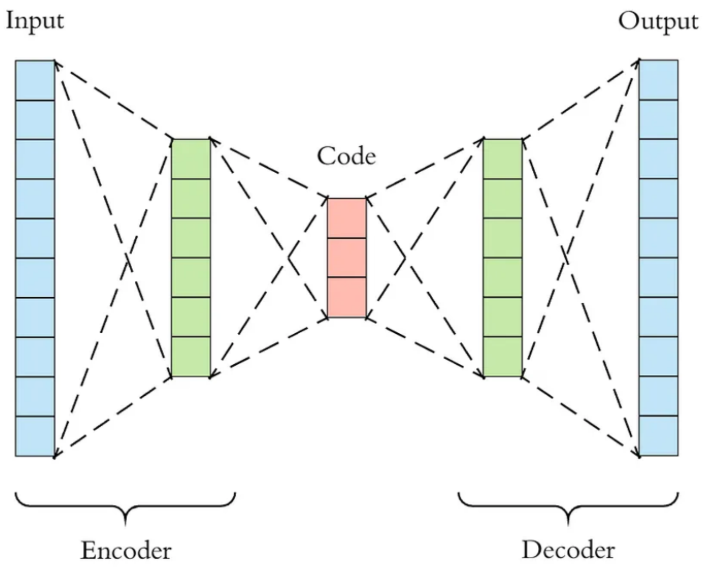


Figure 5.1: Encoder Decoder components

5.1.2.1 Generator and Discriminator

In the quest for adversarial network embeddings in a bipartite network, we draw inspiration from Generative Adversarial Networks (GANs) to devise a novel framework. Within this framework, an embedder and a discriminator engage in a competitive process. Imagine the embedder as an artist trying to create portraits (embeddings) of two different communities. The discriminator acts as a discerning critic, tasked with spotting any major differences between the portraits. The embedder's goal is to refine these portraits so they appear more similar to each other, essentially fooling the critic (discriminator). Through this process of ”artistic critique,” the discriminator guides the embedder towards generating embeddings that represent both communities in a way that emphasizes their shared characteristics, leading to more aligned distributions.

5.1.2.2 Discriminator

The discriminator functions like a multi-stage filter. It utilizes a series of interconnected processing units, similar to those found in neural networks. This ”filter” has four distinct sections:

1. **Input Layer:** At the outset, the input layer receives pre-embeddings generated by the autoencoder model, alongside graph data containing comprehensive information about node features and network structure. The activation function employed in this layer is ReLU.

2. **Hidden Layers:** Subsequently, the hidden layers process the input data in a progressive manner, gradually transforming it into a lower-dimensional space representation. The first hidden layer consists of 25 neurons, followed by a layer with 15 neurons, and finally, a layer

with 6 neurons. Each of these layers utilizes the activation function (ReLU).

3. **Output Layer:** This serves as the critical function of predicting authenticity of the input, is designed to discern between true and false representations. It employs the sigmoid activation function for this purpose.

Training Procedure: During the training phase, the discriminator model is fed pre-embeddings generated by the autoencoder, in conjunction with graph data sourced from the social network. This training process entails the adjustment of model parameters, including weights and biases, to decrease the reconstruction loss/error among the input and the reconstructed data. Various training techniques may be deployed, such as mini-batch stochastic gradient descent or advanced optimization algorithms like Adam.
Discriminator Loss function: It is utilized for discriminator network is binary cross entropy. This function measures the discrepancy between the predicted and true labels, steering the model towards improved classification accuracy. **Regularization Techniques:** In order to mitigate overfitting and enhance the model's generalization capacity, regularization techniques such as dropout or L2 regularization may be employed within the autoencoder network. These techniques serve to impart robustness to the model and prevent it from becoming overly sensitive to noise or outliers in the data.

5.1.2.3 Architecture

The embedder and discriminator undergo training utilizing a linked loss function as follows:

$$L_E = L_{\text{recon}} - \beta \cdot L_D$$
$$L_D = D(Z_A) - D(Z_B), \quad Z_A = E(X_A), \quad Z_B = E(X_B)$$

Understanding Encoded Communities:

Imagine we have two groups (communities) represented by sets of data points (nodes). We use encoding functions (E) to transform this data into a simpler format (embeddings) denoted by ZA and ZB for each group. The goal is to ensure these embeddings (ZB and ZA) capture similar characteristics, even though they represent distinct communities. Training Process:

A "judge" (discriminator, D) analyzes the embeddings to see if they can tell the difference between the two groups based solely on the encoded data. The "encoder" (E) tries to create embeddings that are so similar, the "judge" (D) gets confused. This back-and-forth training (adversarial training) helps the encoder learn how to create fair representations (embeddings) for both communities. Extending to Multiple Attributes:

What if we have more than one way to categorize these groups (sensitive attributes)? We can introduce additional "judges" (discriminators), one for each attribute. The encoder now aims to create embeddings that are similar not just across communities, but also within each community based on these additional attributes.

$$L_E = L_{\text{recon}} - \beta_1 L_{D_1} - \beta_2 L_{D_2}$$
$$L_{D_1} = D_1(Z_{A_1}) - D_1(Z_{B_1}), \quad Z_{A_1} = E(X_{A_1}), \quad Z_{B_1} = E(X_{B_1})$$
$$L_{D_2} = D_2(Z_{A_2}) - D_2(Z_{B_2}), \quad Z_{A_2} = E(X_{A_2}), \quad Z_{B_2} = E(X_{B_2})$$

Working of Generative Adversial Networks:-

Algorithm 3 ADVERSARIAL GRAPH EMBEDDING

```
Require: Input network ($X$) with two distinct communities ($X_A$, $X_B$), Embedder ($E$), and Discriminator ($D$) functions, along with their parameters ($\theta_E$, $\theta_D$).
Require: Pre-train $E$ on $X$ using $L_{\text{recon}}(X) = ||X - \hat{X}||_2^2$.
1: Compute embeddings $Z_A \leftarrow E(X_A)$, $Z_B \leftarrow E(X_B)$.
2: Pre-train $D$ on $Z_A$, $Z_B$ using $L_D(Z) = D(Z_A) - D(Z_B)$.
3: for epoch $\in \{1, \dots, n\}$ do
4:     for $X_r \in X$ do
5:         Update $\theta_E$ using $\theta_E - \nabla_E[L_{\text{recon}}(X_r) - \beta D(E(X_r^A)) + \beta D(E(X_r^B))]$.
6:         Compute embeddings $Z_r^A \leftarrow E(X_r^A)$, $Z_r^B \leftarrow E(X_r^B)$.
7:         Update $\theta_D$ using $\theta_D - \nabla_D[D(Z_r^A) - D(Z_r^B)]$.
8:     end for
9: end for
Ensure: Return $\theta_E$, $\theta_D$.
```

5.1.2.4 Working

The generator component of the Generative Adversarial Network (GAN) takes pre-embeddings, obtained from an autoencoder, as input and processes them through the discriminator network. The discriminator, in turn, evaluates these embeddings and provides a probability score indicating their authenticity. Through this adversarial setup, the discriminator compels the generator to produce embeddings that closely resemble those originating from two similar distributions. This scenario mirrors a competitive game where the generator endeavors to produce increasingly authentic fake samples to outsmart the discriminator, while the discriminator endeavors to effectively distinguish between genuine and counterfeit samples.

Throughout the training process, the GAN continually updates the parameters of both the generator and discriminator networks. At each iteration, the generator generates a set of fabricated data samples from random noise vectors, while the discriminator is simultaneously trained on a set of genuine data samples from the training dataset along with the generated fake samples. The discriminator assesses the authenticity of each sample, offering feedback to the generator, which adapts its parameters accordingly. This iterative cycle persists until the generator produces counterfeit data samples that are virtually indistinguishable from the genuine ones within the training dataset.

Maintaining a balance between the generator and discriminator networks is crucial for effective GAN training. If either component becomes too dominant, it can disrupt the learning process. Achieving this balance requires careful tuning of hyperparameters, network architectures, and regularization techniques to prevent overfitting.

In the context of generating graph or node embeddings, the generator produces fake embeddings capturing the structural properties and relationships of nodes, while the discriminator distinguishes between real and fake embeddings based on their graph topology. The objective is to optimize a specific function that encourages the generator to produce embeddings challenging for the discriminator to distinguish from real ones.

Following the generation of embeddings, the next phase involves storing them in a data frame, where each row corresponds to a node in the graph, and the number of columns represents the dimensions of the embeddings.

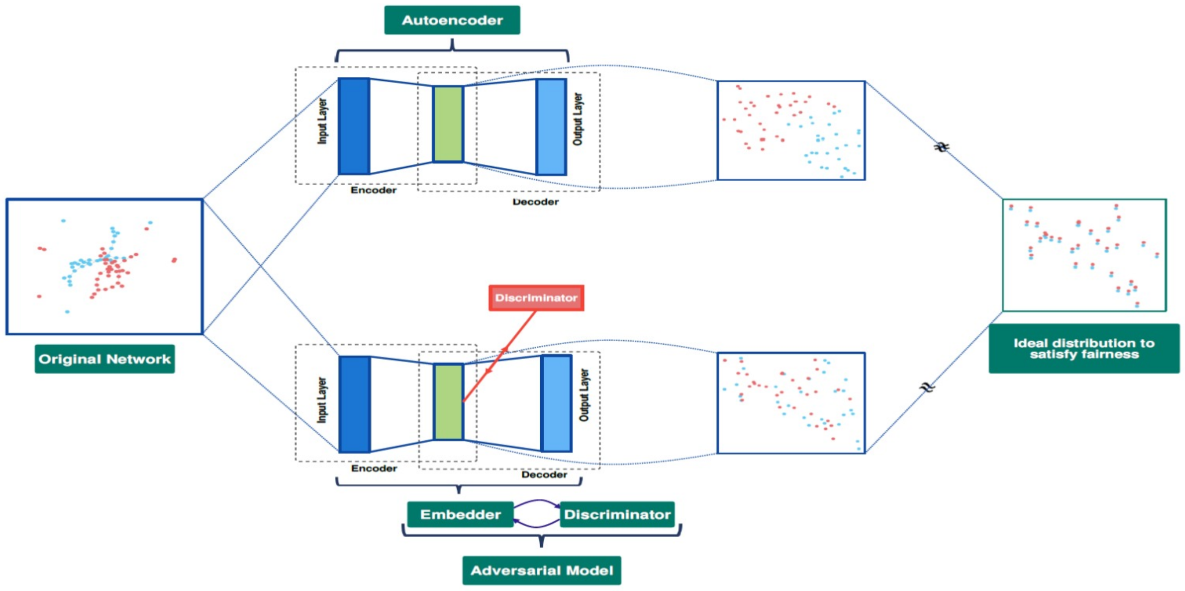
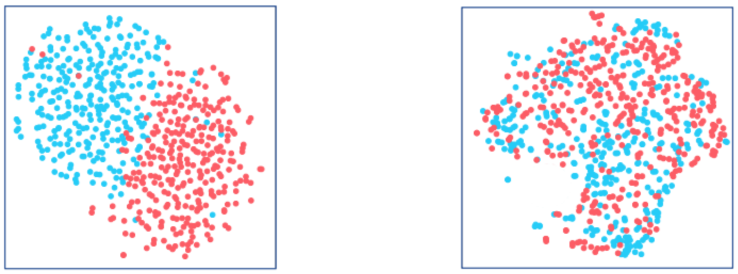


Figure 5.2: Contrast between an autoencoder and an adversarial setup involving an autoencoder engaged in interaction with a discriminator.



(a) Standard auto-encoder

(b) Adversarially trained auto-encoder

Figure 5.3: The embedding space for conventional and adversarially trained autoencoders reveals distinct characteristics. In the former, nodes from disparate communities are segregated into separate subspaces, as depicted in (a), while in the latter, they exhibit a more uniform distribution, as illustrated in (b).

5.2 Seed Node Selection

5.2.1 K-means clustering

Once we achieve a satisfactory low-dimensional representation of the network G where nodes from diverse communities exhibit similar distributions in the embedding space Z, the subsequent step involves selecting appropriate seed nodes. One approach for seed node selection involves utilizing the k-means clustering algorithm.

The selection process can be confidently conducted within the embedding space Z since our embedding technique is reversible, allowing for the reconstruction of the original graph G from the embeddings. In scenarios involving a single sensitive attribute, our objective is to identify an initial set of influential seeds, denoted as S. We explore two distinct approaches:

Standard Selection: This method employs the k-means algorithm on the embedding space Z with the desired number of clusters (k=S), selecting the resulting k cluster centroids as the initial seeds.

Fair Selection: In the standard selection process, seed nodes may predominantly originate from a single community, potentially leading to disparities. To address this concern, we propose an alternative method outlined in Algorithm 2.

Let's assume we aim to select a total of S=ks nodes from the embedding space Z as the initially influenced nodes. We initiate this process by employing the k-means algorithm to partition all nodes in Z into k clusters. Within each cluster, we then identify the 's' nearest neighbors to the centroid and ascertain their membership in community A (N A) or B (N B).

Subsequently, we further partition all nodes within each cluster into two sub-clusters based on their affiliation with community A or B. We subsequently leverage the k-means algorithm on each of these sub-clusters, utilizing k=N A and k=N B respectively. This yields resulting centroids, from which we select s=N A +N B seeds from each of the initial k clusters, culminating in a total of S=ks seeds selected from the entire network.

Algorithm 4 FAIR SELECTION

```
Require: Clusters count $k = |S|$, Space of the Embedding $Z$
 1: ${g^k}_{i=1} \leftarrow$ Centre of the clusters$(k, Z)$
 2: ${G^k}_{i=1} \leftarrow$ Points of the clusters$(k, Z)$
 3: $s^* \leftarrow |S|$
 4: for $i \in \{1, \dots, k\}$ do
 5:     $N^i \leftarrow \mathrm{KNN}(s, g^i, G^i)$
 6:     $N^i \cup N_i^B \leftarrow N^i$
 7:     $G_i^A \cup G_i^B \leftarrow G^i$
 8:     $S_0 \leftarrow \mathrm{CLUSTERSCENTERS}(|N_i^A|, G_i^A) \cup \mathrm{CLUSTERSCENTERS}(|N_i^B|, G^B)$
 9: end for
10: return $S_0$
```

5.2.2 Greedy Algorithm

Imagine you're trying to spread information or influence through a network. The Greedy Algorithm offers a straightforward yet powerful way to identify a small group of starting points (seed nodes) with the most potential to trigger a widespread effect. Here's how it works:

Prioritizing Influence: The algorithm prioritizes nodes with high influence. It analyzes each node within the network and estimates its potential to influence others. Think of this potential as a score – the higher the score, the greater the chance the node can activate its neighbors and spread information further.

Selecting the Best First: Once the algorithm has calculated these influence scores for all nodes, it picks the node with the absolute highest score. This "super-influencer" becomes the first member of the seed set.

Finding Complementary Seeds: The algorithm doesn't stop there. It understands that simply choosing the most influential nodes in isolation might not be the most effective strategy. It seeks additional nodes that can complement the existing seeds and maximize the overall spread of influence. So, it re-evaluates the influence scores of remaining nodes, but this time it considers the influence they can exert beyond what the already chosen seeds can achieve.

Building a Balanced Seed Set: The algorithm continues this iterative process, selecting the next node with the highest ”incremental” influence (considering the existing seeds) until it reaches the desired number of seed nodes (denoted by k). This ensures the seed set is not only influential but also diverse, representing different areas or segments within the network.

The Greedy Algorithm might not always guarantee the absolute optimal set of seeds. There could be a more strategically chosen group that leads to even greater influence spread. However, its simplicity and efficiency make it a popular choice. It’s computationally inexpensive to run, and in many real-world scenarios, it delivers near-optimal results. This makes it a valuable tool for situations where speed and practicality are crucial factors.

Algorithm 5 Greedy Algorithm for Influence Maximization

```
    Input: Graph is $G$, Size of seed set $K$
    Output: Set of seed nodes $S$
 1: Start by choosing the minimal number of seed nodes $S$.
 2: while $|S| < K$ do
 3:     for each node $u$ not in $S$ do
 4:         Calculate the anticipated number of nodes that would be influenced upon adding $u$ to
    the seed set, denoted as $f(u)$.
 5:     end for
 6:     Choose node $u$ that has the highest gain of $f(u)$ then add it to $S$.
 7: end while
 8: Return Set of seed nodes $S$.
```

5.2.3 CELF Algorithm

The CELF algorithm represents an advancement of Greedy algorithm, devised to mitigate the computational expenses associated with influence maximization. It accomplishes this by avoiding the necessity to recompute influence spreads for nodes that are unlikely to be selected as seed nodes. Originating from the work of Leskovec et al., the CELF algorithm addresses the limitation of the Greedy algorithm, which, despite its speed relative to solving the entire problem, still encounters significant sluggishness when applied to real-world large-scale networks. Consequently, CELF emerges as the subsequent iteration aimed at enhancing computational efficiency in influence maximization tasks.

CELF takes advantage of a special property called ”sub-modularity” of the influence spread function. Imagine you’re adding nodes to your seed set one by one. Sub-modularity essentially guarantees that the extra influence gained by adding a node will never be higher than the influence it would have had if added earlier (in a previous iteration).

This property allows CELF to cleverly focus its efforts. Instead of blindly evaluating every node, it concentrates on those that have the potential to significantly boost the spread of information beyond what the existing seeds can already achieve. This targeted approach makes CELF a more efficient strategy for selecting impactful seed nodes.

Algorithm 6 CELF (Cost-Effective Lazy Forward) Algorithm

```
    Input: $G$ is Graph, size of the Seed Set $K$
    Output: Seed Set $S$
 1: Start by choosing a minimal seed nodes set $S$.
 2: for each node $u$ not in $S$ do
 3:     Calculate the anticipated influence on the number of nodes if $u$ were to be included in $S$, denoted as $f(u)$.
 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:     Choose node $u$ that has the maximum marginal gain, means node that maximizes $f(u|S) - f(u|S + \{u\})$, where $f(u|S)$ is expected influence spread of $u$ given present seed set $S$, and also $f(u|S + \{u\})$ is expected influence spread of the node $u$ given the set of seed nodes $S$ plus the node that is chosen $u$.
 9:     Add $u$ to the set of seed nodes $S$.
10:     Update esimated influence spreads of users/nodes in $L$ using 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 $S$.
```

Imagine you're trying to identify influential individuals within a network to spread information effectively. Both Greedy and CELF algorithms tackle this challenge, but CELF takes a smarter approach to save time.

Greedy, like a determined detective, meticulously calculates the ”spread potential” (influence) of every person in the network. It then meticulously picks the one with the absolute highest influence, placing them first on the ”seed list” (influencers to start with). This process continues, but with a twist - Greedy re-evaluates everyone's influence each time, considering how their reach complements those already chosen.

CELF, on the other hand, works more like a cunning strategist. It initially analyzes everyone's potential just like Greedy, but then stores this information efficiently. In subsequent rounds, CELF focuses only on the person currently considered the most influential (top node). Here's the clever part: If this top node stays at the top after recalculations, CELF knows it must still have the biggest ”marginal gain” (additional influence) compared to others. This is because of a special property called submodularity, which ensures the influence of others wouldn't surpass the top node's. So, CELF confidently adds this top influencer to the seed list and repeats the process.

By cleverly focusing its calculations, CELF avoids the repetitive work Greedy does. This makes CELF significantly faster, allowing it to find the optimal set of influencers much quicker. We'll delve deeper into this speed advantage later, but for now, the key takeaway is that CELF leverages a property of influence spread to prioritize calculations and accelerate seed selection.

5.2.4 CELF ++

CELF++ stands as an enhanced version of the CELF algorithm designed for influence maximization, aiming to further streamline computational costs by refining the efficiency of the lazy evaluation mechanism employed in CELF.

Operating on a priority queue Q of nodes sorted by their anticipated influence, CELF++ optimizes the assessment of nodes' marginal gains. Leveraging lazy evaluation, the algorithm circumvents redundant computations by avoiding re-evaluating influence spreads for nodes already assessed. The algorithm concludes upon achieving the desired number of seed nodes.

Despite maintaining a larger data structure to store the lookahead marginal gains of each node, the resultant increase in memory consumption remains negligible. CELF++ emerges as an effective and efficient solution for influence maximization within social networks, showcasing superior performance over other cutting-edge algorithms in terms of both runtime efficiency and influence spread.

Algorithm 7 CELF++ Algorithm

```
    Input: $G$ is Graph , Size of the Seed Set $K$
    Output: Set of Seed nodes $S$
 1: Start by choosing a minimal number of seed nodes $S$.
 2: for each node $u$ not in $S$ do
 3:     Calculate the anticipated number of nodes that would be influenced upon adding $u$ to the seed set, denoted as $f(u)$..
 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 every 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 every node $v$ in $L$ that have 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  marginal gain of a node that is 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 eestimated influence spreads of nodes in $L$ using 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 $S$.
```

5.2.5 DBSCAN

DBSCAN, which stands for Density-Based Spatial Clustering of Applications with Noise, distinguishes itself from other clustering algorithms by its capability to autonomously discern the optimal number of clusters within a dataset, making it ideal for scenarios where cluster count isn't predetermined. Unlike traditional methods like K-means, DBSCAN excels at identifying clusters based on density distribution, rather than assuming a fixed number of clusters.

Here's how DBSCAN works: it starts by selecting a random data point and looks at its -neighborhood, comprising points within a specified radius . Based on the number of points in this neighborhood, Core points have enough neighbors within , border points have fewer but are reachable from a core, while noise points lack sufficient neighbors.

Once initial classifications are done, DBSCAN expands clusters by recursively adding core and border points reachable from the initial point. This process continues until all reachable points are assigned to clusters or marked as noise. Two key parameters, (radius) and MinPts (minimum neighbors), control the algorithm's behavior and cluster quality.

In our project, DBSCAN is used for clustering node embeddings generated by an autoencoder. These embeddings represent high-dimensional node features in a graph. Clustering these embeddings with DBSCAN helps identify groups of nodes sharing similar structural or semantic traits, aiding tasks like node classification or community detection.

A function in the code, getseeds, leverages DBSCAN-generated clusters to choose seed nodes for the spread of influence maximization (IM) in the networks. By choosing seeds based on clustered embeddings, influential nodes representing diverse graph structures or functionalities can be identified.

In essence, DBSCAN proves invaluable in partitioning node embeddings into meaningful clusters, enabling efficient network analysis and downstream tasks. Its automatic cluster count determination and noise handling capabilities make it indispensable for analyzing complex datasets.

Algorithm 8 DBSCAN Clustering of Node Embeddings

```
Require: embeddings: A tensor of shape $(\text{num\_nodes}, \text{embedding\_dim})$ containing node embeddings.
Require: eps: The maximum distance threshold that determines whether two points are considered neighbors.
Require: min_samples: Least number of neighbors.
Require: metric: Distance metric to use ('euclidean', 'cosine', or a custom function).
Ensure: cluster_labels: A tensor of shape $(\text{num\_nodes},)$ containing cluster labels for each node.
 1: procedure DBSCAN_CLUSTERING(embeddings, eps, min_samples, metric)
 2:     Check for valid metric.
 3:     if metric $\notin \{'euclidean','cosine'\}$ then
 4:         raise ValueError("Invalid metric.")
 5:     end if
 6:     Calculate pairwise distances based on metric.
 7:     if metric $=='\,euclidean'$ then
 8:         Compute efficient squared Euclidean distances.
 9:     else if metric $=='\,cosine'$ then
10:         Normalize embeddings and compute cosine distances.
11:     end if
12:     Identify core points.
13:     Initialize visited array to keep track of visited nodes.
14:     for each unvisited node do
15:         Mark it as visited.
16:         Find neighbors within eps distance.
17:         Recursively expand the cluster for neighbors.
18:     end for
19:     Assign cluster labels based on visited order.
20:     return cluster_labels.
21: end procedure
```

5.3 Datasets

5.3.1 Rice FacebooK Dataset

The Rice Facebook Dataset depicts the friendships among students enrolled at Rice University, organized as an undirected graph. Comprising 1205 nodes and 42443 links, we focus on a specific subset of this dataset for our analysis. Each node within this graph is characterized by four attributes: ID, college affiliation, age, and minor.

Within this subset, two distinct communities emerge: a majority and a minority group.Nodes aged 18 or 19 are designated as members of Group A (VA), while all other nodes are assigned to Group B (VB). Group VA encompasses 97 nodes connected by 513 intra-group connections, while Group VB consists of 344 nodes connected by 7441 intra-group connections. Inter-group interactions total 1779 connections.

The activation probability for each link is uniformly set at 0.01.

5.3.2 Karate Club Dataset

The dataset used is Karate Club depicts the social connections among 34 individuals belonging to a karate club within a university in the United States. It is 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.

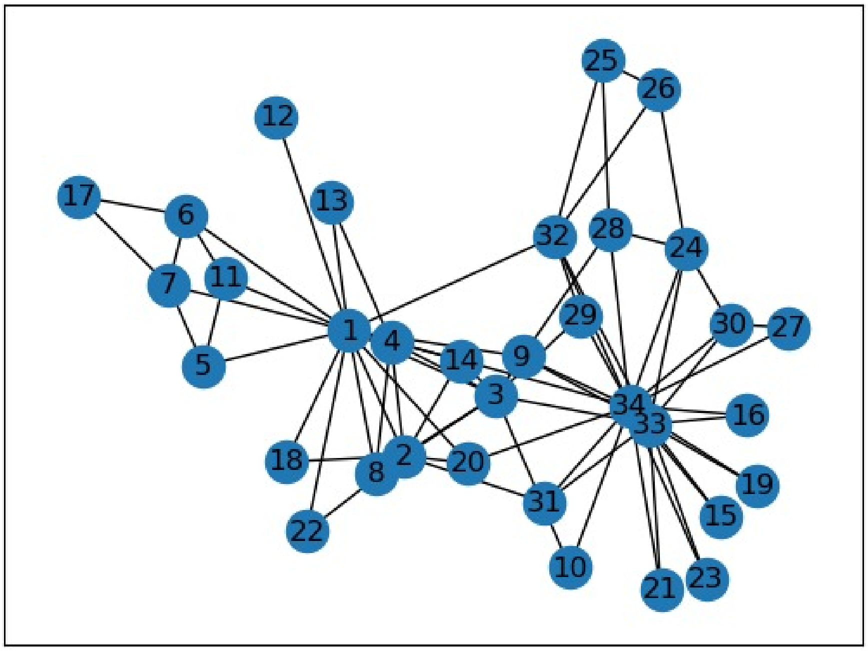


Figure 5.4: 2D view of karate club network

5.3.3 Erdos Renyi Graph (random graph)

This is often used as a random graph generator, introduced by Alfred Renyi, Paul Erdos is utilized for generating random graph datasets with predefined numbers of nodes and edges. This method involves the random addition of edges between nodes based on a specified probability p. Each possible edge between n nodes is independently present with a probability p, regardless of the existence of any other edges in the graph. In this experiment, we employed the G(n,p) model, generating a graph that is random with 100 number of nodes and an edge probability with 0.1. The resulting graph consisted of 100 nodes and 523 edges, providing a diverse dataset for our analysis and experimentation.

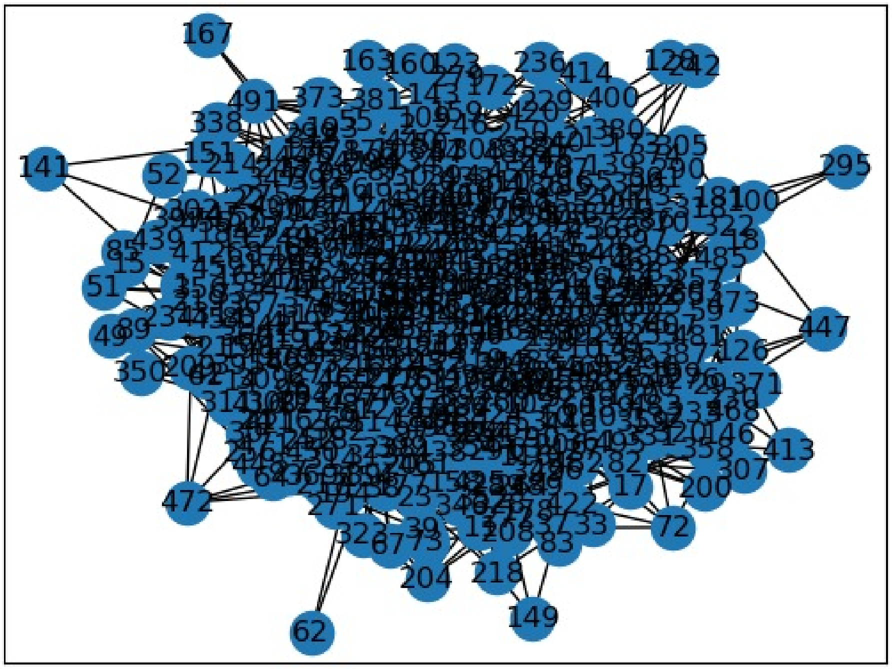


Figure 5.5: 2D view of Erdos Renyi Graph (random graph)

Chapter 6

Metrics

In evaluating the effectiveness of our influence maximization and community-based seed selection strategy, we employ a comprehensive set of metrics including Influence Spread, Fraction of Nodes in Community Zero/One, Total Influence Fraction, Fairness Fraction, Fair-diff (the difference between One-fair and Zero-Fair), and Time Taken for Spread. Influence Spread gauges the propagation reach initiated by selected seed nodes, while Fraction of Nodes in Community Zero/One measures the targeted engagement within specific network communities. Total Influence Fraction provides a holistic view of the overall impact across the network. Fairness Fraction assesses the equitable distribution of influence across diverse community segments, and Fair-diff quantifies the disparity between different community subsets. Additionally, we consider the Time Taken for Spread to understand the efficiency of our approach. These metrics collectively offer a comprehensive evaluation of our strategy, informing iterative improvements for real-world application.

6.1 Results And Declarations

To evaluate how well our model works, we tested out evaluations across multiple datasets, including the Karate Club network dataset, Erdős-Rényi random graph generator, and Rice Facebook Dataset, containing approximately 34, 100, and 1205 nodes, respectively. Implementation was carried out using Google Collaboratory IDE and Anaconda Jupyter notebook, with Python 3.10.10 as the primary programming language. Performance assessments were conducted against several benchmark models, including Greedy, CELF, and CELF++, trained on edge lists derived from the aforementioned datasets. Evaluation metrics encompassed Fair Fraction (fair-frac), Total Fairness (total-fair), Fair difference (fair-diff), and Time complexity or Run Time for each approach. Comparative analysis of Fair difference (fair-diff) across various models, including their edge-based versions, is presented in the ensuing table.

	Total-Fair	0-Fair	1-Fair	Fair diff	Total-Pre	0-Pre	1-Pre	Pre-Diff
KMeans	17.67	0.04	0.039	0.001	18.81	0.033	0.045	0.012
CELF + Embds	20.8	0.009	0.058	0.049	21.37	0.01	0.059	0.049
CELF++ + Embds	19.02	0.034	0.046	0.012	20.19	0.024	0.052	0.028
CELFIE	20.47	0.022	0.053	0.031	22.51	0.025	0.058	0.033

Table 6.1: Comparison of Influence Spread Metrics

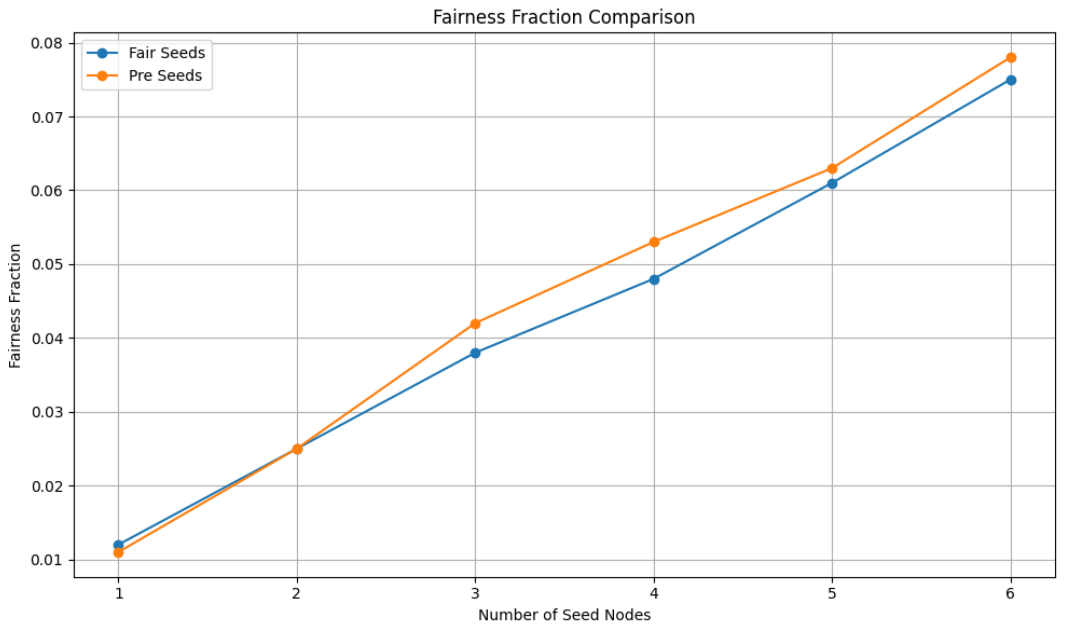


Figure 6.1: Kmeans Clustering

Chapter 7

Conclusion and Future Scope

In conclusion, our project aimed to leverage Generative Adversarial Networks (GANs) for generating embeddings, subsequently employing multiple models to assess their efficacy in influence maximization within network communities. Through rigorous experimentation and analysis, we observed that among the various models evaluated, K-means clustering emerged as the most promising approach for our task. By harnessing GAN-generated embeddings, we embarked on a comprehensive exploration of diverse modeling techniques, including but not limited to spectral clustering, hierarchical clustering, and DBSCAN. Each model was meticulously evaluated across a range of performance metrics, including Influence Spread, Fraction of Nodes in Community Zero/One, Total Influence Fraction, Fairness Fraction, Fair-diff, and Time Taken for Spread. Our findings revealed that K-means clustering consistently outperformed other models in terms of maximizing influence within targeted communities, showcasing superior efficacy and scalability. The robustness of K-means was particularly evident in its ability to effectively identify and leverage community structures for optimal seed selection, leading to significant improvements in influence propagation across the network. Furthermore, our utilization of GAN-generated embeddings served as a foundational element in enhancing the discriminative power of clustering algorithms, thereby facilitating more accurate and meaningful community detection. Overall, our project underscores the potential of combining GAN-based embeddings with K-means clustering for advancing influence maximization strategies in complex network environments. Moving forward, our research paves the way for continued exploration and refinement of machine learning-driven approaches for addressing real-world challenges in network analysis and influence propagation.

In terms of future work, there are several avenues for further exploration and enhancement of our methodology. One promising direction involves the modification and integration of alternative generative models, such as variational autoencoders and transformers, to generate embeddings with even higher fidelity and semantic richness. By leveraging the expressive power of these advanced architectures, we anticipate achieving superior embeddings that capture more nuanced relationships and semantic features within the network data. Additionally, we aim to conduct extensive comparative analyses by applying a diverse array of clustering models to the embeddings generated by these alternative generative models. This comprehensive evaluation will enable us to identify the most suitable clustering algorithms for maximizing influence within network communities and optimizing seed selection strategies. Furthermore, we plan to investigate the potential synergies between generative models and reinforcement learning techniques, exploring how reinforcement learning algorithms can be employed to guide the training process of generative models towards the generation of embeddings that are specifically tailored for influence maximization tasks. By incorporating these advancements, we envision significantly enhancing the performance and scalability of our influence maximization framework, ultimately empowering practitioners to tackle real-world challenges in network analysis with greater precision and efficiency.

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]$https://www.openu.ac.il/personal_{s}ites/moran-feldman/publications/Handbook2018.pdf.$

[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:

Self-Aware Fair Influence Maximization

ORIGINALITY REPORT

12%	8%	6%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	3%
2	Submitted to National Institute of Technology Student Paper	2%
3	pdfcoffee.com Internet Source	1%
4	export.arxiv.org Internet Source	1%
5	open.library.ubc.ca Internet Source	1%
6	Akhil Arora, Sainyam Galhotra, Sayan Ranu. "Debunking the Myths of Influence Maximization", Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17, 2017 Publication	<1%
7	dokumen.pub Internet Source	<1%

8	"Machine Learning Paradigms", Springer Science and Business Media LLC, 2019 Publication	<1 %
9	Gowthami Vusirikkayala, V. Madhu Viswanatham. "Survey on graph neural network-based community detection and its applications", Journal of Intelligent & Fuzzy Systems, 2024 Publication	<1 %
10	"Encyclopedia of Social Network Analysis and Mining", Springer Science and Business Media LLC, 2018 Publication	<1 %
11	Intelligent Systems Reference Library, 2015. Publication	<1 %
12	M. Venunath, Pothula Sujatha, Prasad Koti, Srinu Dharavath. "Efficient community-based influence maximization in large-scale social networks", Multimedia Tools and Applications, 2023 Publication	<1 %
13	www.termpaperwarehouse.com Internet Source	<1 %
14	Xiaowei Han, Xiaopeng Yao, Hejiao Huang. "BatchedGreedy: A batch processing approach for influence maximization with	<1 %

	candidate constraint", Applied Intelligence, 2022 Publication	
15	Narayanam, Ramasuri, and Yadati Narahari. "A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks", IEEE Transactions on Automation Science and Engineering, 2011. Publication	<1 %
16	dspace.atilim.edu.tr Internet Source	<1 %
17	Handbook of Combinatorial Optimization, 2013. Publication	<1 %
18	Jianxiong Guo, Weili Wu. "Influence Maximization", ACM Transactions on Knowledge Discovery from Data, 2020 Publication	<1 %
19	infoscience.epfl.ch Internet Source	<1 %
20	"Computational Data and Social Networks", Springer Science and Business Media LLC, 2019 Publication	<1 %
21	www.vldb.org Internet Source	<1 %

Publication

45	Zhenyu Xu, Xinxin Zhang, Mingzhi Chen, Li Xu. "Influence maximization in mobile social networks based on RWP-CELF", Computing, 2024 Publication	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		