People's Democratic Republic of Algeria
Ministery of Higher Education and Scientific Research
Mohamed El Bachir El Ibrahimi University of Borj Bou Arréridj
Faculty of Mathematics and Informatics
Informatics Department

DISSERTATION

Presented in fulfillment of the requirements of obtaining the degree

Master in Informatics

Specialty: Information Technologies and Communications

THEME

User Similarity Measures in Online Social Networks

Presented by:

Abedelaziz BELABACI

Khalil SAIDANI

Publicly defended on: 11th June 2025 In front of the jury composed of:

President: Dr. Safa ATTIA

Examiner: Dr. Meriem REGOUID

Supervisor: Dr. Meriem LAIFA

CO-Supervisor: Ms. Aya ZOUAOUI

2024/2025

Acknowledgment

Above all, we thank Allah, the Most Gracious and Most Merciful, for giving us the strength, patience, and guidance to fulfill this project.

We express our gratitude to our supervisor, Dr. Mariem Laifa, for her essential guidance, encouragement, and support throughout the project's progress. The expertise and mentoring she provided were extremely important to the success of our work.

Also, we send our thanks to her PhD student, Aya Zouaoui, for her patient assistance, helpful suggestions, and willing help and support throughout all phases of the project. We also appreciate the members of the jury for their time, feedback, and judgment of our work.

Without forgetting our families for their support, encouragement, and understanding for all we have been through.

We express our sincere thanks to all the professors of our university for the great knowledge, guidance, and academic support during our studies.

Abstract

With the growing use of Online Social Networks (OSNs), these platforms have become central to engagement, participation, and community formation. As user bases expand, understanding user similarity becomes increasingly important for applications such as friend recommendation, community detection, and spam detection. Measuring user similarity includes determining behavioral patterns in addition to profile similarities and behavioral differences. In this research, we examine some classical similarity measures (Jaccard, Cosine, and Euclidean) versus deep learning-based measures, including Node Embeddings, Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE with a synthetic Twitter dataset. User interactions are modeled as graphs and transformed into low-dimensional embeddings. Using the Deep Graph Infomax framework for unsupervised learning, clustering is applied and assessed through internal evaluation metrics: Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Score. The objective is to identify the most effective method for discovering meaningful user clusters in OSNs.

Keywords: User Similarity, Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, Graph Infomax (DGI), Cosine Similarity, Jaccard Similarity, Euclidean Distance.

Résumé

Avec l'utilisation croissante des réseaux sociaux en ligne, ces plateformes sont devenues essentielles à l'engagement, à la participation et au développement de communautés. À mesure que le nombre d'utilisateurs s'accroît, la compréhension de la similarité utilisateur devient de plus en plus importante pour des applications telles que la recommandation d'amis, la détection de communautés et la détection de spam. La mesure de la similarité utilisateur inclut la détermination de schémas comportementaux, ainsi que des similarités et des différences de profils. Dans cette recherche, nous examinons certaines mesures de similarité classiques (Jaccard, Cosinus et Euclidienne) par rapport à des méthodes d'apprentissage profond, notamment les Incorporations de nœuds, les réseaux convolutifs de graphes (GCN), les réseaux d'attention de graphes (GAT) et GraphSAGE avec un jeu de données Twitter synthétique. Les interactions utilisateur sont modélisées sous forme de graphes et transformées en Incorporations de nœuds de faible dimension. À l'aide du framework Deep Graph Infomax pour l'apprentissage non supervisé, le regroupement est appliqué et évalué à l'aide de mesures d'évaluation internes : le score Silhouette, l'indice Davies-Bouldin et le score Calinski-Harabasz. L'objectif est d'identifier la méthode la plus efficace pour découvrir des groupes d'utilisateurs significatifs dans les réseaux sociaux en ligne .

Mots-clés : la similarité utilisateur , Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, Graph Infomax (DGI), , Cosine Similarity, Jaccard Similarity, Euclidean Distance.

ملخص

مع الاستخدام المتزايد لشبكات التواصل الاجتماعي عبر الإنترنت (OSNs)، أصبحت هذه المنصات محورية لمشاركة المجتمع وتكوينه. مع توسع قاعدة المستخدمين، يصبح فهم تشابه المستخدم ذا أهمية متزايدة لتطبيقات مثل توصية الأصدقاء واكتشاف المجتمع واكتشاف البريد العشوائي. يتضمن قياس تشابه المستخدم تحديد الأنماط السلوكية بالإضافة إلى أوجه التشابه في الملف الشخصي والاختلافات السلوكية.

في هذا البحث، ننظر إلى بعض مقاييس التشابه الكلاسيكية (جاكارد وجيب التمام والإقليدية) مقارنة بالطرق القائمة على التعلم العميق، بما في ذلك تضمينات العقد وشبكات الالتفاف البياني (GCN) وشبكات الانتباه البياني (GAT) وGraphSAGE مع مجموعة بيانات تويتر الاصطناعية. يتم تصميم تفاعلات المستخدم على شكل رسوم بيانية وتحويلها إلى تضمينات عقد صغيرة الحجم.

باستخدام إطار عمل Deep Graph Infomax للتعلم غير الخاضع للإشراف، يتم تطبيق التجميع وتقييمه باستخدام مقاييس التقييم الداخلية: Silhouette Score وDavies-Bouldin Index وCalinski-Harabasz Score. الهدف هو تحديد الطريقة الأكثر فعالية لاكتشاف مجموعات المستخدمين المهمة في شبكات OSN.

Contents

Chapter 3: Methodology

Chapter 4: Experiments, Results, and Discussion

Abbreviations list

User Similarity,

Graph Convolutional

Networks (GCN),

Graph Attention Networks (GAT),

GraphSAGE, Graph Infomax (DGI), ,

Cosine Similarity,

Jaccard Similarity,

Euclidean Distance.

List of Figures

List of Tables

Introduction General

With the rapid growth in the use of Online Social Networks (OSNs), they have become a base for engagement, participation, and community development. OSNs such as X (Formally Twitter), Facebook, and LinkedIn enable users to post, share, and connect with each other, leading to massive volumes of user-generated content and dynamic social structures. Understanding the relationships and similarities among users within these networks has gained importance in applications such as friend recommendation, community detection, and spam detection.

As OSNs continue to grow, the measurement of user similarity involves more than a profile comparison for a single feature but also user behavioral patterns, and an examination of differences between users. This study aims to evaluate different methods for computing user similarity by using classical similarity measures and graph-based deep learning feature extraction approaches. The key objectives of this work are:

- To extract and engineer features from a synthetic X dataset that represents user profiles and interactions.
- To calculate traditional similarity measures (Jaccard, Cosine, and Euclidean) along with more advanced methods from deep learning (Node Embeddings, Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE).
- To assess the effectiveness of clustering users based on the resulting similarity matrices and embedding vectors using its evaluation metrics.

To achieve these objectives, the study follows a multi-step methodology. It begins with data preprocessing and feature engineering to transform raw user data into structured numerical representations. A graph is then constructed to model user interactions, upon which unsupervised Graph Neural Network models are trained using Deep Graph Infomax (DGI). The embeddings produced by GCN, GAT, and GraphSAGE are subsequently used to compute user similarity and perform clustering. Finally, clustering quality is evaluated using metrics such as Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index.

To better define our study theme, we have organized the work into four chapters: the first chapter will define the basic domains and concepts around which the work is structured and will define OSNs and user similarity.

- The second chapter will define node embeddings and their types, concluding with their challenges and limitations.
- The third chapter presents the methodology, including feature extraction, graph construction, node embedding techniques, and similarity calculation.
- As for the fourth and final chapter, which occupies a key part of this work, we will discuss the results obtained and then evaluate them.

- The conclusion will summarize the power of graph-based deep learning methods for embeddings in capturing user similarity, present the limitations of using synthetic datasets, and provide future research directions.

Chapter 1:

An overview of Online Social Networks and User Similarity

1.1 Introduction

This chapter explains Online Social Networks (OSNs) and user similarity, describing their functionality, use, and applications. We also review a number of similarity measures based on profile and behavior attributes, to set the context of our research study.

1.2 Online social network (OSN)

1.2.1 OSN's Definition

Social network sites are web-based services designed to facilitate social interaction and contact among users. OSN allows users to create a public or semi-public profile in a bounded system of the site, which is an online version of their identity. The profile may contain information that is personal, photographs, and multimedia. Additionally, users may display and represent other users that they have relationships with, generally referred to as "Friends", "Contacts", or "Followers"-depending on the website. These relationships are the social relationships that users have, whether they are already existing relationships or relationships that they have constructed. Finally, users can view and navigate through their list of connections, as well as navigate through other people's connections within the system. This enables the functionality of users to browse the social network, discover overlapping connections, and expand their network in a visible and interactive way. As a whole, these functionalities provide a formalized framework where users can maintain, display, and study their social relations within a virtual environment (Boyd & Ellison, 2007).

1.2.2 Functionalities and usage of OSNs

In general, OSNs are considered essential tools in the modern world to perform both social and professional activities and entertainment purposes. They offer quite a number of features that help in the provision of communication, interaction, and sharing of content among users. The platforms provide services like messaging, video calls, and group chat, which help people communicate with their friends, family, or colleagues from anywhere in the world (Boyd & Ellison, 2007). OSNs also help in sharing different types of content, such as pictures, videos, articles, to mention but a few, in order to create a sense of belonging. Also, they provide features such as updates, stories, and live videos, which help people to post their current events.

Moreover, OSNs are used as professional networking sites where people can create relationships, post industry information, and work on group projects[2](Kaplan & Haenlein, 2010). Furthermore, these platforms have a way of recommending content through algorithms, which suggest posts, news, and advertisements to the users based on their interests and activities.

OSNs have multiple goals, and they provide the ability to quickly and easily communicate and stay in touch no matter the distance. OSNs are also important for information sharing and changing how users think, and shaping public opinion. OSNs provide an opportunity for social network analysis, where researchers can analyze human behavior, quantify relationships, and also track how ideas or trends are diffused throughout society (Zafarani et al., 2014). Besides, they assist in cybersecurity by detecting fake profiles, preventing fraud, and combating malicious activities such as spreading misinformation and rumors. OSNs are utilized by companies for targeted marketing, brand promotion, and customer engagement, while governments and institutions use them to establish public awareness and disseminate critical information during crises.

1.2.3 Applications of OSNs

Online Social Networks (OSNs) enable a variety of applications across a range of domains, through the use of user similarity measures. The applications can range from improving user experience, to securing digital environments. Similarity measures (both structural and behavioral) can act as the lifeblood of these systems(Zafarani et al., 2014). **Table 1.1** is a summary of the key applications' types, their definitions, and methods that are typically utilized in their realization.

Table 1.1 Applications of OSNs with Techniques and Similarity Measures

Area of Application	Definition	Methods
Recommendation Systems	Remake an impression of a user, content, or link, based on similarities to other accounts, behaviors, or types of interaction.	Link prediction, user-user/item-item similarity, collaborative filtering.
Behavior Analysis	Reconstruct user personality, preferences, or participation over time.	Temporal activity modelling, frequency analysis and tracer metrics or engagement metrics.
Cybersecurity	Identify fraudulent activities, such as bots, fake accounts, or rumour amplifiers.	Profile similarity checking, anomalies, or machine learning classifiers.
Content Personalization and Information Retrieval	Adapt content to user preferences in personalized feeds or suggested pieces.	Topic modelling, deep learning embeddings, or similarity metrics.
Targeted Advertisement and Social Influence	Design ads based on user behaviours, interests, attributes, and dynamic social interaction.	Social influence modelling, clustering methods, or sentiment analysis.
Community Detection and Social Structure Analysis	To find out or describe communities and their required clusters, to understand the group formations and conceivable social structures.	Use graph clustering (i.e, Louvain, Girvan-Newman), maximization of modularity.

1.3 User similarity concept

1.3.1 Similarity definition

In the context of OSNs, similarity refers to how similar two users or profiles are in a number of areas, including activity patterns, social connections, and shared content. It forms the basis of identifying trends, suggesting connections, and identifying spammers or duplicate accounts (Ferres et al., 2014).

1.3.2 Similarity Classification

User similarity in online social networks (OSNs) can be classified into different kinds based on the types of data and analytical perspective. Classifications will assist in customizing algorithms in recommendation, link prediction, community detection, user profiling processes, and so on. **Table 1.2** shows the main types of similarity classification, the main focus, and some common methods.

Table 1.2 Classification of User Similarity Types in OSNs

Similarity Classification Type	Key Focus	Common Techniques
Profile	User attributes	Jaccard, Cosine
Behavioral	Actions & interactions	Sequence analysis, frequency vectors
Structural	Network connections	Graph-based metrics
Content-Based	Shared content	TF-IDF, LDA
Temporal	Timing of activity	Time-series analysis
Interest-Based	User preferences	Hashtag/topic modeling
Semantic	Meaning of content	NLP embeddings
Embedding-Based	Learned vector representation	Node2Vec, BERT
Interaction	Shared engagements	Interaction graphs
Heterogeneous	Multi-source similarity	Hybrid models

1.3.3 User Similarity Measures

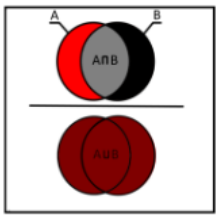
1.3.3.1 Jaccard Similarity

Jaccard similarity is a classic measure of similarity between two sets. When applied to Twitter accounts, it can measure similarity between users on the basis of how many people they share an audience or interact with. The Jaccard coefficient is defined as the number of elements in common between two sets, divided by the number of unique elements.. It has the following formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- A and B represent sets of followers, interactions, or interests.

- $|A \cap B|$ is the number of mutual followers or shared interactions.
- $|A \cup B|$ is the total number of unique elements in both sets (Singh et al., 2019).



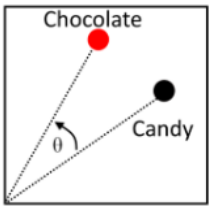
1.3.3.2 Cosine Similarity

Cosine similarity measures the directional similarity of two non-zero vectors, rather than how close the two vectors are to each other in magnitude. Cosine similarity is useful for high-dimensional and sparse datasets like text-based user profiles and tweet embeddings. The formula for cosine similarity is the following:

$$cos\ cos\ (\theta) = \frac{A \cdot B}{||A||||B||}$$

where:

- A and B are vectors representing text features (e.g., from tweets or bios).
- $A \cdot B$ is the dot product of the two vectors.
- $|A|$ and $|B|$ are the magnitudes (norms) of the vectors (Singh et al., 2019).



1.3.3.3 Euclidean Similarity

Euclidean similarity is described in terms of Euclidean or geometric distance , that is, how far it is from an origin to a point in space. With user data (which is user profiles or interaction vectors) we can use this type of distance to show how close or similar two users are with respect to their only features or behaviors, depending on the relationship chosen.. This is a good approach when the absolute differences of numerical features have some significance. The Euclidean distance between two vectors A and B is computed using the formula:

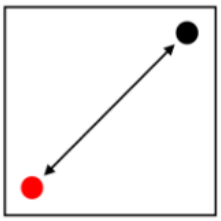
$$dist(A, B) = \sqrt{\sum_i \ (A_i - B_i)^2}$$

To turn this into a similarity measure (because smaller distances mean more similarity), one common choice is to carry out an inverse transformation:

$$similarity(A,B) = \frac{1}{1+distA,B}$$

Where

- A and B are feature vectors representing the users.
- The denominator bounds the similarity score between 0 and 1.



This is quite effective for continuous data and feels intuitive for representing magnitude-based closeness between users (Itkin & Soleymani, 2021).

1.3.3.4 Deep Learning Techniques

Traditional similarity metrics mentioned earlier employ handcrafted features and elementary statistical techniques. However, these approaches often fail to learn complex user relations in OSNs. Deep learning techniques are more scalable and flexible for computing similarity by directly learning representations from data(Hamilton et al., 2017b).

The main deep learning models are the following:

A. Graph Neural Networks (GNNs)

GNNs extend deep learning to graph-structured data and are very convenient in analyzing user similarity in OSNs. They learn node representations called node embeddings that capture network structure, and similarity is computed using Cosine Similarity or Euclidean Distance of user embeddings.

And have been used especially for friend recommendations, where similar users are suggested based on network embeddings (Zhang et al., 2022). They have also been used for community detection, in particular to cluster similar user communities with similar interests.

Community Detection: Clustering similar user communities with similar interests. The main architectures used for such neural networks are:

Graph Convolutional Networks (GCN): it aggregates information from user relations (Kipf & Welling, 2016).

Graph Attention Networks (GAT): it uses an attention mechanism to pay attention to essential relations (Veličković et al., 2017).

B. Node2Vec and DeepWalk (Graph Embeddings)

In OSNs, relationships and similarities between users play a major role in friend recommendations, community detection, and identification of fake or spurious accounts. The two techniques-a known graph embedding technique-known as Node2Vec and DeepWalk-equally map user entities within the social network to a low-dimensional space and assign their similarity in vector form through metrics like Cosine Similarity. Such embeddings encode the relational and structural information of users and facilitate efficient and precise similarity calculation.

DeepWalk: Uniform Random Walks for Scalable Embedding

One of the earliest algorithms to learn node embeddings treats the network as a "document" and does uniform random walks to create node sequences . In NLP, these are like sentences where the nodes are like words. Then the Skip-gram model (which was originally employed for word embeddings) can be employed to learn the embeddings which maximize the likelihood to predict the neighbors of a node from the sequences(Perozzi et al., 2014).

That core idea is that nodes which share the same patterns of random walks are likely to be similar. It is particularly effective at identifying homophily, the tendency of people in social networks to be around other people who share similar properties or interests.

Node2Vec: Flexible Biased Random Walks for Richer Embeddings

Node2Vec generalizes DeepWalk with a more generic random walk policy with controllable exploration of the graph. It uses two parameters, p (return parameter) and q (in-out parameter), that limit the walk's behavior:

Breadth-First Sampling (BFS): When q is big, the walk stays near the starting node, and structural equivalence is adequately picked up. This proves to be helpful in searching for participants with similar roles within the network (e.g., influencers or bridge users) despite them not directly being connected.

Depth-First Sampling (DFS): Because q is limited, the walk moves deeper from the starting node, seeing homophily same community users or the same interest users.

Node2Vec generalizes between BFS and DFS and learns representations observing homophily as well as structural equivalence more broadly applicable for calculating similarities in OSNs (Grover & Leskovec, 2016).

Both DeepWalk and Node2Vec can be used to measure similarities between users in OSNs by computing the Cosine Similarity of their embeddings. This similarity metric can then be leveraged for several tasks:

• **Friend Recommendation:** Users whose similarity values arFriend Recommendatione high tend to share similar interests or belong to the same community, making them suitable friend recommendation targets.

• **Community Detection:** Similar users can be grouped into communities, helping identify clusters of users with shared behaviors or properties.

• **Fake Account Detection:** Imposter accounts typically exhibit unusual connectivity patterns. By comparing their embeddings to those of genuine users, anomalies can be identified (Grover & Leskovec, 2016).

Table 1.3 summarizes the differences between DeepWalk and Node2Vec, two widely-used random walk-based embedding techniques.

Table 1.3 Comparison of DeepWalk and Node2Vec

Feature	**DeepWalk**	**Node2Vec**
Walk Type	Purely Random	Biased Random Walk
Flexibility	Limited	Can balance local/global exploration
Control Over Walks	No control	Uses parameters p and q to adjust walk behavior
Best For	Community detection	Both communities & role-based similarity

C. CNN-LSTM Hybrid Models for Text-Based Similarity

A combination of Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) is suitable for the estimation of content similarity between users. CNN captures local features of the text, while LSTM captures long-range dependencies in the content of the users. This is best suited to capture semantic similarity in tweets, user profiles, and shared items.

Such hybrid models are usually employed for detecting fake accounts by identifying users with copied content. Additionally, they are used for content-based user matching to retrieve users having common interests based on their tweet analysis or their bios (Grover & Leskovec, 2016).

Table 1.4 provides a comparative overview of popular user similarity techniques, highlighting their characteristics across several dimensions.

Table 1.4 The differences between this methods for user similarity in online social networks

Aspect	**Jaccard Similarity**	**Cosine Similarity**	**Graph-Based Methods**	**Deep Learning Methods**
General	Basic, good for binary features	Suited for high-dimensional data	Captures structural similarity well	High accuracy if trained well on rich data
Data Type	Binary/set-based	Continuous or binary vectors	Graph/network structures	Multi-modal (text, graph, profile, behavior)
Scalability	Fast for small-medium datasets	Efficient for sparse, high-dim vectors	Slower on large, dense graphs unless optimized	Resource-heavy, but scalable with proper frameworks
Interpretability	High	Intuitive	Medium, varies by method	Low (black-box models)
Computational Cost	Low	Low to Medium	Medium to High	High
Implementation	Easy	Easy	Moderate (requires graph tools)	Complex (requires ML/DL knowledge)
Examples	Jaccard Index	Cosine on TF-IDF vectors	Common Neighbors, Katz, SimRank	GNNs (GraphSAGE, GAT), Transformers (BERT, DistilBERT)

2.4 Conclusion

This chapter provides an introduction to Online Social Networks (OSNs), their characteristics, and their applications. We introduced user similarity as one of the main topics, and outlined the similarities between users by their usages of dynamic networks using different measures including Jaccard Similarity, Cosine Similarity, Euclidean Similarity, and Deep Learning methods.

Chapter 2:

Node Embeddings

2.1 Introduction

This chapter examines node embeddings as dense vector representations of users in online social networks (OSNs), where users constitute nodes within complex graph structures characterized by both structural and semantic relationships. While traditional feature engineering approaches remain applicable to OSN analysis, node embeddings demonstrate superior scalability, flexibility in modeling diverse relationships, and the capacity to learn directly from graph adjacency structures. The chapter presents key embedding methodologies including DeepWalk, Node2Vec, GraphSAGE, Graph Convolutional Networks (GCNs), and Graph Attention Networks (GATs), while also addressing practical implementation challenges such as scalability constraints, interpretability limitations, and data quality issues encountered in real-world OSN deployments.

2.2 The concept of node embeddings

2.2.1 Definition of node embeddings

Node embeddings are low-dimensional vector representations of the nodes (users) in a network. Node embeddings are learned so that they encode the structural properties and relationships of nodes in the graph. The goal is to retain the aspects of the network topology, like proximity and neighborhood, in the learned embedding space.

Formally, graph $G = (V, E)$ with nodes V and edges E, a node embedding is a function $f: V \rightarrow R^d$ that maps each node to a ddd-dimensional vector space (Perozzi et al., 2014). The goal of these vectors is to preserve the graph's topology, so that nodes with similar roles or positions in the graph have similar embeddings.

2.2.2 Importance of Node Embeddings in OSNs

Node embeddings are valuable representations in OSNs, availing the ability to discover hidden structural and semantic relationships that exist in a dynamic and complex network environment:

Hidden Structural Patterns: Node embeddings retain the topological features of a graph at both the local and global level, thus helping in discovering hidden information like communities, hierarchy, and influential nodes. Also, provide a means to map nodes into continuous vector spaces representing their relationship in the network. Often, the relative position and role of that node in the network is encoded implicitly, beyond the links present in the graph.

Semantic Proximity: Embeddings can capture explicit connections (e.g., “follows” or “friends”) and they can also represent hidden semantic similarities such as shared interests, behaviours or interaction. This allows a system to infer the relationships that are not visible in the network representation of raw data.

Adaptation to Dynamics: In dynamic OSNs with continuous evolution of the structural network, node embeddings allow maintenance of up-to-date representations through dynamic or temporal models to look for both past and present relationships (Bhaskaran et al., 2021).

2.2.3 The Differences between node embeddings and traditional feature engineering

Node embeddings and traditional feature engineering represent two different methods of representing nodes in networks. Traditional feature engineering relies on manual entry and domain knowledge related to the manually constructed features, while in node embeddings, using data-driven algorithms allows for the automatic learning of the vector representation. To summarize the key differences between our approach to classifying node embeddings (Hamilton et al., 2017b), **Table 5** presents the specific attributes of these approaches in four key areas: creation, representation, flexibility, and scalability.

Table 2.5 The differences between node embeddings and traditional feature engineering

Aspect	Traditional Feature Engineering	Node Embeddings
Creation Process	Requires manual feature creation with a reliance on subject matter expertise. It can take a long time to create and can only be reflective of a limited number of concepts.	Automatically learned from data from algorithms, like random walks or neural networks. Can capture local patterns and global patterns.
Representation Characteristic	Create sparse and high-dimensional vectors (ex., one-hot encoding).	Generate dense, low-dimensional vectors that encode semantic and structural knowledge.
Adaptability and Learning	Static features that need to be updated with manual input when patterns in the data change.	Dynamic and Adjustable. Embeddings can incorporate new data as it arises, making them better equipped for dynamic networks.
Implicitly Capturing Relationships	Obtaining explicit relationships with higher-order or more complex relationships is slightly more limited, even more so when we consider complex graph structures.	Effectively capture and preserve the original direct and indirect relationships while maintaining the original graph topology.
Scalability and Efficiency	Can be inefficient and tricky to scale. More complex structures may become even harder to work and reason with.	Embeddings can be scaled easily and efficiently, after training speed is a function of similarities and comparison and only requires a few vector operations to complete.

2.3 Types of Node Embedding

2.3.1 Random Walk-Based Embeddings

2.3.1.1 DeepWalk

DeepWalk is one of the pioneering approaches in the field of graph representation learning. It formulates the problem of learning latent node representations by treating truncated random walks in graphs as analogous to sentences in natural language processing. This enables the application of Word2Vec's Skip-Gram model to develop embeddings from node co-occurrence in sampled walks. The steps are :

Random Walk Generation: For every node in the graph, generate several uniform random walks. Each random walk produces a sequence of nodes, where the transitions between any neighboring nodes are made uniformly at random, thus allowing the representation of the local neighborhood structure.

Contextual Embedding via Skip-Gram: Treat the generated node sequences as sentences and train a Skip-Gram model to predict a node's context (the neighboring nodes within a specified window size). Each resulting node embedding thus captures local structural similarities.

Selecting and Optimizing: Model the generative process as maximizing the probability of the contexts, given a node's embedding. Stochastic Gradient descent is employed with hierarchical softmax or negative sampling to facilitate an understanding of latent representations, and hopefully give certain advice about working with biases (Perozzi et al., 2014).

The DeepWalk algorithm effectively captures homophily and community structure, but does not have control over the type of embedding similarity, like a scenario of Capturing social context from truncated random walks; useful for community detection and role-based similarities.

2.3.1.2 Node2Vec

Node2Vec improves on DeepWalk by allowing for biased second-order random walks, and thus a flexible way of sampling the local and global structure of a graph. Two hyperparameters, p and q, control the walk behavior:

Small value for p encourages returning to previously visited nodes, which simulates depth-first search (DFS) favored for capturing homophily.

Small value for q encourages exploration of nodes further away from the starting node, which simulates breadth-first search (BFS) favored for detecting structural equivalences

This bias is adjusted to create a second-order dependency in walks, which achieves a balance between homophily and structural roles. The same Skip-Gram model of DeepWalk is used to generate embeddings directly from the sequences of nodes (Grover & Leskovec, 2016).

Empirical results of Node2Vec suggest it generally outperforms DeepWalk across a variety of tasks, especially in networks where structural roles are important.

2.3.2 Random Walks and Skip-Gram in Graph Embedding

Random walks and the Skip-Gram model used together form a powerful framework through which to learn node embeddings:

Random Walks: Provide a way to sample from the structure of the graph. DeepWalk uses uniform sampling while Node2Vec creates biased sampling in order to vary the exploration strategy.

Skip-Gram Model: given a node u, the goal is to maximize the log-probability of the context nodes $Ns\ (u)$, defined in the context of a sliding window based on .

$$\sum_{u \in V} \quad log\ Pr(Ns(u) \mid f(u)) \text{ [5]}$$

Improving computational efficiency on the scale of large graphs is accomplished through hierarchical softmax or negative sampling. The end result of these embeddings signifies node proximity and structural similarity based on the random walk strategy used.

2.3.3 Graph Neural Network (GNN)-Based Embeddings

2.3.3.1 GraphSAGE

GraphSAGE (Graph Sample and Aggregate) is an inductive GNN framework that discovers node embeddings by sampling and aggregating features from neighbors. GraphSAGE is the first inductive framework to generalize to unseen nodes while retaining the semantics derived from features (Hamilton et al., 2017a).

In this framework operating on k layers, a node's representation is updated by aggregating its neighbors' node embeddings, and some neighbor aggregation functions include mean, LSTM, and pooling aggregations. After k iterations, a node embedding will have incorporated information from at most a k-hop neighborhood. GraphSAGE encourages relatively fast time complexity, flexible online learning, and linear time complexity with respect to the number of edges (E). These factors make this model particularly applicable to large, dynamic systems, such as social networks and IoT networks.Use case: Aggregation of neighbour information in an inductive manner; with large-scale and dynamic graphs, this model is suitable as it allows for new nodes to be encountered in the GraghSAGE process.

2.3.3.2 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks take the idea of convolutions to graph-structured data, enabling semi-supervised learning. GCNs take the normalized adjacency matrix and include self-loops, making for a way to propagate and transform the features of the node, use it in combination with node features and topology for semi-supervised learning for node classification and recommendation in attributed OSNs.

$$H^{(l+1)} = \sigma(\widehat{D}^{-\frac{1}{2}}\ \hat{A}\widehat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$$

- $H^{(l)}$ Is the input feature matrix at layer l
- $W^{(l)}$ the trainable weight matrix at layer l
- $\hat{A}\ = A + I$ Is the adjacency matrix with added self-loops
- $\widehat{D}$ is the diagonal degree matrix of $\hat{A}$
- σ is a non-linear activation function (Kipf & Welling, 2016).

2.3.3.3 Graph Attention Networks (GATs)

Graph Attention Networks improve on GATs by using an attention mechanism that gives each neighbor a learnable weight when aggregating. Attention coefficients are computed based on source and destination feature similarity, then normalized with softmax.

$$h_i' = \sigma\left(\sum_{j \in N} \quad \alpha_{ij} W h_j\right)$$

The attention mechanism affords the model the ability to dynamically attend to more informative neighbors, which can improve performance on heterogeneous graphs. Using a multi-head attention mechanism increases robustness and expressiveness, which makes GATs applicable to tasks that require understanding nuanced relational reasoning (Veličković et al., 2017).

GATs provide a high degree of interpretability and consistently demonstrate good performance on tasks including node classification and anomaly detection. Gives attention weights to neighbors on aggregation; gives advantage when learning from heterogeneous and noisy networks .

2.4 Challenges and Limitations of Node Embeddings

While node embedding methods derive powerful ways to analyze online social networks (OSNs), they face a number of challenges.

Scaling to Large Networks when working with large OSNs with millions of users and user-user interactions, embedding algorithms may face scaling issues. Many current methods in embedding learning are based on global computations, which can be a computational bottleneck, and potentially one that requires significant memory and processing power, such as full-graph training and dense similarity matrices. While methods such as GraphSAGE and sampling-based approaches relieve some of the scalability issues, further tradeoffs arise between scaling and accuracy.

Interpretability of Embeddings. Another limitation of node embeddings is interpretability. Traditional features may have the potential for semantics (for example, number of friends or number of times they have been interacted with), but because embedding dimensions are

abstract and uninterpretable after training and optimization, we still need better methods for determining semantics. It is seldom obvious why two users would be grouped as similar, which can be a great issue in contexts that seek transparency or explainable results.

Reliant on the Quality of the Graph. Finally, the success of the embeddings is dependent on the structure and completeness of the input graph. A graph can have errors, be incomplete, have fragmented information, or be biased, and as such, the resulting embeddings will not represent the user user behavior or relationships captured in the graph (Rodrigues & Rocha, 2021). For example, the absence of an edge or wrong edges will misrepresent proximity measures of nodes and lead to inaccurate recommendation predictions, clustering, and other downstream tasks.

2.5 Conclusion

Node embeddings offer a flexible and scalable way of capturing both structural and semantic relationships to represent users in OSNs, which go beyond the conventional features that can lead to more accurate similarity comparisons. However, there were still some challenges to node embeddings for OSN users, including scalability, interpretability, and the quality of data used to create the embeddings. Node embeddings can represent users in OSNs due to modern embedding techniques, specifically embedding methods based on graph neural networks (GNN), that can provide a more basic framework to model complex or mobile-based social networks.

Chapter 3:

Methodology

3.1 Introduction

This chapter presents the Development and Programming Environment and the methodology used to compute user similarity in Online Social Networks (OSNs), contrasting traditional feature-based approaches with modern graph-based methods. It presents the complete experimental process: from the extraction of profile and behavioral features, to the creation of an embedding of users via Graph Neural Networks (with Deep Graph Infomax with GCN, GAT, and GraphSAGE), then clustering and evaluating the outputs via internal measures. We were interested in assessing the success of the approaches we used to identify user communities through quantitative scoring and also visually.

3.2 Development and Programming Environment

For the implementation, we used a Lenovo computer with the following features:

- Processor: AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00 GHz.

- Installed memory (RAM): 8 GB.

- System Type: operating system 64-bit.

- Microsoft Windows 10.

In preparing the implementation environment, we used **Google Colab**, a cloud-based platform that provides access to powerful computing resources, including free GPUs. This environment enabled the efficient training of deep learning models.

- **Google Colaboratory (Google Colab)** is a free, cloud-based Jupyter notebook environment provided by Google that allows users to write and execute Python code through the browser. It supports seamless integration with Google Drive, offers access to hardware accelerators like GPUs and TPUs, and is widely used for machine learning, data analysis, and education(Bisong, 2019).
- **Python Language** An advanced programming language with a simple structure. It is an effective language for scripting and rapid application development. Notable libraries in this domain include Pandas, Numpy, and Scikit-Learn, which are commonly employed for various data-related operations (loading, reorganization and processing)(Najadat et al., 2019).
- **PyTorch** is an open-source deep learning framework developed by Facebook's AI Research lab. It provides a dynamic computation graph and tensor-based automatic

differentiation, making it highly flexible for building neural networks. PyTorch is widely used for research and production in computer vision, natural language processing, and graph-based deep learning(Paszke et al., 2019).

Libraries Importation Phase

Pandas: it is a Python library used for analyzing, cleaning, exploring, and manipulating datasets. It has a reference to both "Panel Data", and "Python Data Analysis(W3Schools, n.d.).

NumPy: an open source library for t Python, it is an acronym for Numerical and Python and used for scientific programming in Python, particularly for data science, engineering, mathematics or science(Harris et al., 2020).

Scikit-learn: For the Python programming language. Generally used in machine learning projects and offering different algorithms, it is created for the Python numerical and scientific libraries Numpy and SciPy(Pedregosa et al., 2018).

PyTorch Geometric (PyG) is a library built on top of PyTorch, designed specifically for deep learning on irregular structures like graphs and manifolds. It provides efficient tools and GPU-accelerated methods for implementing Graph Neural Networks (GNNs) such as GCN, GAT, and GraphSAGE(Fey & Lenssen, 2019).

Matplotlib: A complete library for producing Php visualizations, called Matplotlib makes simple and difficult things possible(Hunter, 2007).

3.3 Work process

Our research focuses on computing user similarity in OSNs based on profile features and behavioral features. We compute the similarity between users using traditional similarity measures, such as Cosine similarity, Jaccard similarity, and Euclidean similarity, as well as newer methods based on deep learning and Graph Neural Networks (GNNs). To this end, we aim to produce meaningful user embeddings, using unsupervised learning with Deep Graph Infomax (DGI), then cluster them and evaluate the quality of the embeddings. After that, we compare the results. We applied our methods to a Synthetic Twitter dataset of user interactions. **Figure 3.1** below shows the architecture of the research workflow

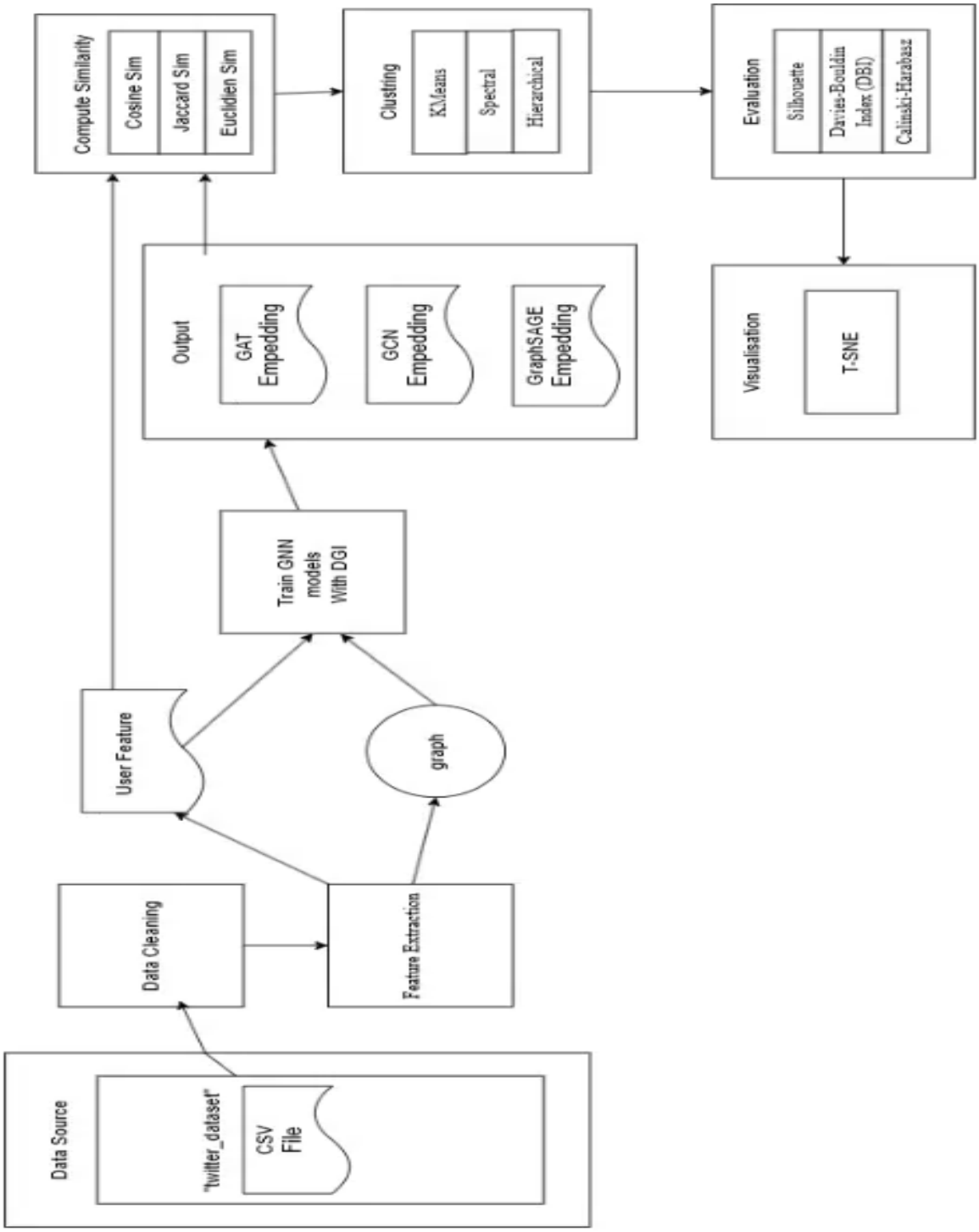


Figure 3.1 Architecture of our research

3.4 Classical Similarity Measures

A design of a comprehensive feature extraction pipeline allows us to exercise classical clustering based on similarity measures such as Jaccard, Cosine, and Euclidean distances. This process transforms raw user data into structured vector representations, catching semantic, categorical, and structural aspects of user profiles.

- Jaccard Similarity is computed by binarizing the feature vectors and evaluating the overlap ratio between feature sets (Singh et al., 2019). It quantifies how much two users have in common (or overlap) compared to the totality of their features, making it an easy way to represent the shared interactions or interests between two users.

- Cosine Similarity is calculated as the cosine of the angle between user feature vectors, highlighting direction-based similarity (Singh et al., 2019). Look at the angle of the feature vectors, not the relative magnitude, which means it is useful when we want to compare user profiles (i.e., the user interactions) but the values may vary considerably across users (for example, comparing user profiles from a text perspective or user engagement patterns).

- Euclidean Similarity is derived by computing the pairwise Euclidean distances and transforming them into similarity scores using the inverse distance formula. It is based only on the actual distances with the feature space, and it accounts for the overall difference when comparing user profiles, so it works well when the actual difference in features is meaningful (for example with continuous or numeric features) (Itkin & Soleymani, 2021).

3.4.1 Feature Extraction

To create a single representation for each user, the feature construction process includes various heterogeneous attributes. For textual data in the cleaned_bio field, semantic information was retained by utilizing the Term Frequency-Inverse Document Frequency (TF-IDF) technique. The user engagement patterns represented in the users_interaction column were tokenized and transformed with the Bag-of-Words (BoW) model to create an organized numerical representation of the users' engagement behavior. The categorical domain attribute was encoded with one-hot encoding in binary vector format, and the total_interactions field representing total user activity was reshaped as a numerical feature. The new feature vectors were all horizontally joined into a single user feature matrix X. **Figure 3.2** explains the architecture of Feature Extraction:

Figure 3.2 Architecture of User Feature Extraction

3.5 Node embeddings

This explanation will justify the use of this work employing node embeddings which is employed to represent the structural and semantic characteristics of the social graph users. Unlike the conventional feature engineering that commonly relies on manually crafted features (like term frequencies or categorical one-hot encodings), node embedding techniques automatically learn low-dimensional, dense representations of the nodes, conveying the node position and role in the graph (Grover & Leskovec, 2016).

This kind of usage is particularly justified for Online Social Networks (OSNs), in which similarity and behavior are determined by the network structure. Vector-based representation fails to adequately represent these structural relationships (Hamilton et al., 2017a).

Using unsupervised graph neural network techniques such as Deep Graph Infomax (DGI) coupled with encoders such as GCN, GAT, or GraphSAGE, the model learns to produce embeddings that reflect local neighborhoods and global structural patterns (Veličković et al., 2018). These representations are more expressive and best suited for tasks like clustering, where network context preservation is essential.

Finally, node embeddings enable more accurate similarity calculation, to obtain higher cluster quality and an accurate representation of user communities in social networks.

3.5.1 Graph Construction

The first step to construct the user interaction graph is by extracting all unique users from the dataset , including both the primary usernames and their associated interaction partners. then Each unique user was assigned a distinct integer identifier to serve as a node within the graph. We established directed edges from source nodes to their corresponding target nodes to represent user interactions. These edges were converted into a tensor format compatible with the PyTorch Geometric framework. The graph output was compact in a Data object containing the edge index tensor along with the total number of nodes.

We initialized the final graph structure by incorporating both node features and the graph topology into a PyTorch Geometric Data object. The latest form of the graph is shown in the **Figure 3.3** below

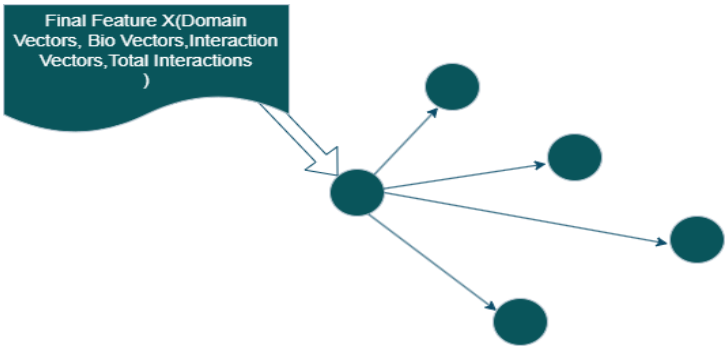


Figure 3.3 Feature Graph Final Feature Composition

3.5.2 GNN-Based Feature Extraction

Once the graph is built with user features and connections based on their interactions, the next step will be using Graph Neural Networks (GNNs) to learn useful representations (embeddings) for each user in the graph

3.5.2.1 Graph Convolutional Network (GCN)

Uses a layer-wise propagation rule to aggregate features from each node's neighbors through its normalized adjacency matrix (Kipf & Welling, 2016). In this work, a standard two-layer GCN encoder with two steps was used to convert raw features into structurally informed embeddings by accounting for both local features, and the topology of the graph.

GCNs are characterized by their ability to preserve the smoothness assumption nodes with similar connectivity are expected to share similar features which works well with dense, homophilic networks like OSNs .

3.5.2.2 Graph Attention Network (GAT)

Employs an attention mechanism to learn weights of each neighbor so the model focuses on the strongest connections for aggregating features from neighboring nodes. In this work, a multi-head attention GAT encoder was utilized to create embeddings that emphasize salient social ties in the network (Veličković et al., 2017).

This metric learns the weights for the neighborhood and provides adaptability, which in social graphs is important because not all user interactions are equally valuable.

3.5.2.3 GraphSAGE

learns embeddings of nodes by sampling and aggregating features from fixed neighbor size. It has several possible aggregation methods to consider. In this case, the aggregation method was mean. Additionally, GraphSAGE generates embeddings in a scalable and inductive manner (Hamilton et al., 2017a).

We used GraphSAGE because it scales well for large data with evolving users, and also because it is able to generalize to unseen nodes.

3.5.2.4 Deep Graph Infomax (DGI) implementation

To train the Graph Neural Network in an unsupervised way without needing labels we use the Deep Graph Infomax (DGI) framework .The model learns via maximizing mutual information between the local node embeddings and a global summary of the graph. Multiple

variations of the encoder architecture are experimented for local node embeddings, including GCN, GAT and GraphSAGE. All types of encoders are capable of learning low-dimensional embeddings for all nodes (Zhang et al., 2022).

The output from each of the three models is a matrix of user embeddings $Z \in R\ n\times$, where rows of the embedding correspond to users and d is the learnt embedding dimension

3.6 Evaluation

3.6.1 Clustering Metrics

The next step is to analyze patterns of similarity among users through clustering. In this research, three popular clustering algorithms are used on both classical similarity and embedding-based similarity matrices to cluster users according to behavioral and structural similarity. Each type has unique strengths with respect to the underlying data distribution and properties of the graph.

KMeans Clustering: is a centroid-based algorithm of partitioning data into k clusters such that intra-cluster variance is minimized, i.e., the sum of squared distances of each point from its corresponding cluster centroid. It is considered to have spherical clusters with equal variance and hence is extremely appropriate for linearly separable datasets (Oti et al., 2021).

Algorithm 1: K-Means Clustering Evaluation

Input:

- sim_matrix – similarity matrix (cosine, jaccard, or euclidean)
- k_range (2 to 10)

Output:

- clustering scores

Convert similarity to distance:
If metric in {cosine, jaccard}:
    dist_matrix = 1 - sim_matrix
Else if metric is euclidean:
    dist_matrix = sim_matrix
For each k in k_range:
    Apply KMeans clustering on dist_matrix
If number of unique labels < 2:
Continue to next k
Compute:
        Silhouette score (precomputed distance)
        Davies-Bouldin index
        Calinski-Harabasz index
    Store scores for k
Return clustering scores

Spectral Clustering: employs graph Laplacian eigenvectors, derived from a similarity matrix, to reduce the dimensionality before clustering. Being capable of discovering intricate, non-convex shapes of clusters, the algorithm performs better than traditional algorithms in discovering such complicated cluster shapes (Jia et al., 2014).

Algorithm 2: Spectral Clustering Evaluation

Input:

- sim_matrix – similarity matrix (cosine, jaccard, or euclidean)
- k_range (2 to 10)

Output:

- clustering score

Construct affinity matrix

If metric in {cosine, jaccard}:

    Set diagonal to 1

Else if metric is euclidean:

    Convert to affinity using RBF kernel:

        gamma = 1 / mean(similarity)

        affinity = exp(-gamma * sim^2)

        Set diagonal to 1

For each k:

  Run Spectral Clustering with affinity_matrix, get labels

  If only one cluster is found, skip

  Compute distance matrix as 1 - affinity_matrix, zero diagonal

  Compute:

    sil = silhouette_score(dist_matrix, labels)

    db = davies_bouldin_score(dist_matrix, labels)

    ch = calinski_harabasz_score(dist_matrix, labels)

    Store scores for k

Return clustering score

Hierarchical Agglomerative Clustering is a bottom-up method where every data point is treated as a single cluster to begin with and then successively merges the closest pairs based on the chosen linkage measure (Murtagh & Contreras, 2012). The process provides a dendrogram of the nested hierarchy of clusters without specifying a number of clusters in advance

Algorithm 3: Hierarchical Clustering Evaluation

Input:

- sim_matrix – similarity matrix (cosine, jaccard, or euclidean)
- k_range (2 to 10)

Output:

- clustering scores

For each metric ∈ {jaccard, euclidean, cosine}:

1. If metric is similarity (jaccard or cosine):
   dist_matrix = 1 - sim_matrix
   Clip negatives, set diagonal to 0
2. Standardize distance matrix using StandardScaler
3. For each k ∈ k_range:
   Apply AgglomerativeClustering(n_clusters=k, metric='precomputed', linkage='average')
   Get labels for data points
   If only one cluster: skip
   Compute:
   - Silhouette Score (precomputed)
   - Davies-Bouldin Index
   - Calinski-Harabasz Index
   Store scores and labels

Return clustering scores

3.6.2 Evaluation Strategy

For the determination of the quality and efficacy of the user groupings built using clustering methods, an evaluation procedure is created, applied on classical feature-based methods as well as graph-based embedding methods from GCN, GAT, and GraphSAGE encoders. The objective is to assess the cohesion, compactness, and separation of clusters in an unsupervised style.

1. Silhouette Score measures how close one point is to its cluster and how close to other clusters. For a point i, it is given by:

$$s(i) = \frac{(b(i) - a(i))}{max(a(i), b(i))}$$

- $a(i)$: mean distance from i to all other points in the same cluster.
- $b(i)$: minimum mean distance from i to points in any other cluster (Rousseeuw, 1987).

The higher the silhouette score (the closer it is to 1), the better the clusters are formed and separated.

2. Davies-Bouldin Index (DBI) is a measure of cluster tightness and separation. It is given by:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \left(\frac{(\sigma_i + \sigma_j)}{d(c_i, c_j)}\right)$$

- σ_i and σ_j: intra-cluster average distances between clusters i and j.
- $d(c_i, c_j)$: Euclidean distance of centroids c_i and c_j.

Smaller DBI values indicate more compact and far-separated clusters(Davies & Bouldin, 1979).

3. Calinski-Harabasz Index (CHI) or the Variance Ratio Criterion, evaluates the ratio of between-cluster dispersion to within-cluster dispersion:

$$CHI = \frac{Tr(W_k)}{Tr\,(B_k)} \cdot \frac{k - 1}{N - k}$$

- $Tr\,(B_k)$: trace of between-cluster dispersion matrix.
- $Tr(W_k)$: trace of within-cluster dispersion matrix.
- N: total data points.
- k: number of clusters (Calinski & Harabasz, 1974).

Higher CHI indicates better-differentiated, highly varying between, and low-variance within clusters.

3.7 Visualization

To facilitate qualitative analysis and interpretation of the clustering structure, we employed t-distributed Stochastic Neighbor Embedding (t-SNE).

This evaluation framework ensures a comprehensive and objective comparison of clustering techniques, and supports the identification of meaningful patterns in user similarity data within OSNs (García-Alonso et al., 2014).

Three internal metrics of evaluation, used in common to measure the quality of clustering in an unsupervised context, are Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index.

3.8 Conclusion

This chapter proposes an approach to measuring user similarity in Online Social Networks by combining traditional measures of similarity with newer graph-based embedding methods. We used information from user profiles and user behaviors to create a feature vector that made it possible to use Jaccard, Cosine, and Euclidean measures. Deep Graph Infomax (DGI) was applied to obtain node embeddings from the OSN interaction and structural properties of the interaction network using GCN, GAT, and GraphSAGE.

We then ran three possibly ambiguous clustering algorithms (K-Means, Spectral, and Agglomerative) on the feature-based similarities and embedding-based similarities. We evaluated the clustering and computer eigengaps using the silhouette score, Davies-Bouldin index, and Calinski-Harabasz index, and included qualitative assessments using t-SNE.

The proposed method in this chapter was a comprehensive approach to allow meaningful comparisons between different methods of assessing user similarity in OSNs and meaningful discovery of user communities on OSN.

Chapter 4:

Experiments, Results, and Discussion

4.1 Introduction

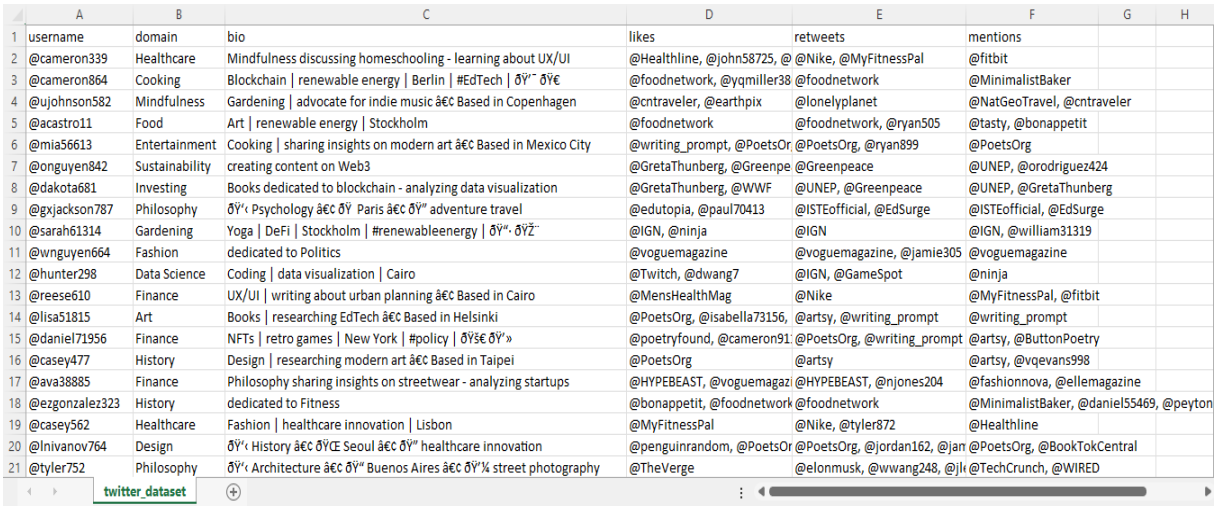
In this chapter, we will present the experimental scheme we implemented to examine user similarity measures on Online Social Networks (OSNs). The key contribution we wanted to make was to compare similarity measures and user similarity measures as traditional vs GNN.

The chapter will start with the dataset and preparations for the dataset; after we finish preparing the dataset, we explain the graph construction. Next we describe the results of the experiment we conducted using both traditional measures and embedding measures, and then we will end with a discussion of the experimental and analytical findings of the experiments, the limitations of the current study and the potential for future work.

4.2 Dataset Description

The dataset utilized in our research is a synthetic X dataset under the name "twitter_dataset.csv." It consists of data for 10,000 users and has been created to mimic realistic X-like user behavior and profiles. Each user profile consists of a bio, which conveys the user's interests, and a domain, which is a categorical feature identifying the user's interest (sports, politics ...). user interactions include likes, retweets, and mentions. Together, these components create the foundation for a graph that captures user interactions by representing users as nodes and interactions as edges.

The synthetic X dataset was selected for this research because of its capabilities to generate realistic user behaviors and interactions. The dataset provides a controllable and scalable environment with respect to specific parameters. It contains 10,000 users, along with user behaviors and profiles as described earlier. Given all of this, we consider it especially useful for assessing user similarity measures in Online Social Networks. **Figure 4.4** presents the dataset used.



	A	B	C	D	E	F	G	H
1	username	domain	bio	likes	retweets	mentions		
2	@cameron339	Healthcare	Mindfulness discussing homeschooling - learning about UX/UI	@Healthline, @john58725, @	@Nike, @MyFitnessPal	@fitbit		
3	@cameron864	Cooking	Blockchain | renewable energy | Berlin | #EdTech | ðŸ¯ ðŸ€	@foodnetwork, @yqmiller38	@foodnetwork	@MinimalistBaker		
4	@ujohnson582	Mindfulness	Gardening | advocate for indie music â€¢ Based in Copenhagen	@cntraveler, @earthpix	@lonelyplanet	@NatGeoTravel, @cntraveler		
5	@acastro11	Food	Art | renewable energy | Stockholm	@foodnetwork	@foodnetwork, @ryan505	@tasty, @bonappetit		
6	@mia56613	Entertainment	Cooking | sharing insights on modern art â€¢ Based in Mexico City	@writing_prompt, @PoetsOr	@PoetsOrg, @ryan899	@PoetsOrg		
7	@onguyen842	Sustainability	creating content on Web3	@GretaThunberg, @Greenpe	@Greenpeace	@UNEP, @orodriguez424		
8	@dakota681	Investing	Books dedicated to blockchain - analyzing data visualization	@GretaThunberg, @WWF	@UNEP, @Greenpeace	@UNEP, @GretaThunberg		
9	@gxjackson787	Philosophy	ðŸ‘‹ Psychology â€¢ ðŸ Paris â€¢ ðŸ‘ adventure travel	@edutopia, @paul70413	@ISTEofficial, @EdSurge	@ISTEofficial, @EdSurge		
10	@sarah61314	Gardening	Yoga | DeFi | Stockholm | #renewableenergy | ðŸ‘ ðŸŽ¨	@IGN, @ninja	@IGN	@IGN, @william31319		
11	@wnguyen664	Fashion	dedicated to Politics	@voguemagazine	@voguemagazine, @jamie305	@voguemagazine		
12	@hunter298	Data Science	Coding | data visualization | Cairo	@Twitch, @dwang7	@IGN, @GameSpot	@ninja		
13	@reese610	Finance	UX/UI | writing about urban planning â€¢ Based in Cairo	@MensHealthMag	@Nike	@MyFitnessPal, @fitbit		
14	@lisa51815	Art	Books | researching EdTech â€¢ Based in Helsinki	@PoetsOrg, @isabella73156,	@artsy, @writing_prompt	@writing_prompt		
15	@daniel71956	Finance	NFTs | retro games | New York | #policy | ðŸš€ ðŸ»	@poetryfound, @cameron91	@PoetsOrg, @writing_prompt	@artsy, @ButtonPoetry		
16	@casey477	History	Design | researching modern art â€¢ Based in Taipei	@PoetsOrg	@artsy	@artsy, @vqevans998		
17	@ava38885	Finance	Philosophy sharing insights on streetwear - analyzing startups	@HYPEBEAST, @voguemagaz	@HYPEBEAST, @njones204	@fashionnova, @ellemagazine		
18	@ezgonzalez323	History	dedicated to Fitness	@bonappetit, @foodnetwork	@foodnetwork	@MinimalistBaker, @daniel55469, @peyton		
19	@casey562	Healthcare	Fashion | healthcare innovation | Lisbon	@MyFitnessPal	@Nike, @tyler872	@Healthline		
20	@lnivanov764	Design	ðŸ‘‹ History â€¢ ðŸŒ Seoul â€¢ ðŸ‘ healthcare innovation	@penguinrandom, @PoetsOr	@PoetsOrg, @jordan162, @jan	@PoetsOrg, @BookTokCentral		
21	@tyler752	Philosophy	ðŸ‘‹ Architecture â€¢ ðŸ‘ Buenos Aires â€¢ ðŸ‘ ¼ street photography	@TheVerge	@elonmusk, @wwang248, @jl	@TechCrunch, @WIRED		

twitter_dataset

Figure 4.4 Dataset sight

4.3 Data Preprocessing and Graph Construction:

4.3.1 Data Preprocessing and Cleaning:

To ensure data consistency and prepare our dataset for accurate similarity analysis, a set of data normalization and cleaning steps was applied.

The start will be on text columns that were trimmed of white space at both the beginning and end. Some key column attributes included in this process were username, domain, bio, likes, retweets, and mentions. The last three attributes were combined in the user_interaction column.

Also, all text values for the username and user interaction fields were lower cased in order to standardize their format. To standardize identifiers in the dataset, non-alphanumeric characters were dropped from the username and all user interaction fields, as well as from the usernames involved in interactions.

All non-alphanumeric characters were dropped from bio, as only textual information was sought for each bio. The bio field was linked directly to the original usernames.

In creating the dataset for the core user base, duplicated entries were checked against the username attribute and removed, leaving only one username on the dataset.

To verify completeness of the dataset, a process was performed to double check the list of users for interactions against the list of core users in the dataset. If a user assigned to a user interaction was found in this verification process, but their username did not find an entry on

the core users dataset, the username was entered as a new user with their non-username fields left blank (potentially as missing).

By completing this process, all users representing interaction data were included in the final dataset, resulting in enough user references to create a clean user base to extract features and compute similarities. Like **Figure 4.5** showing

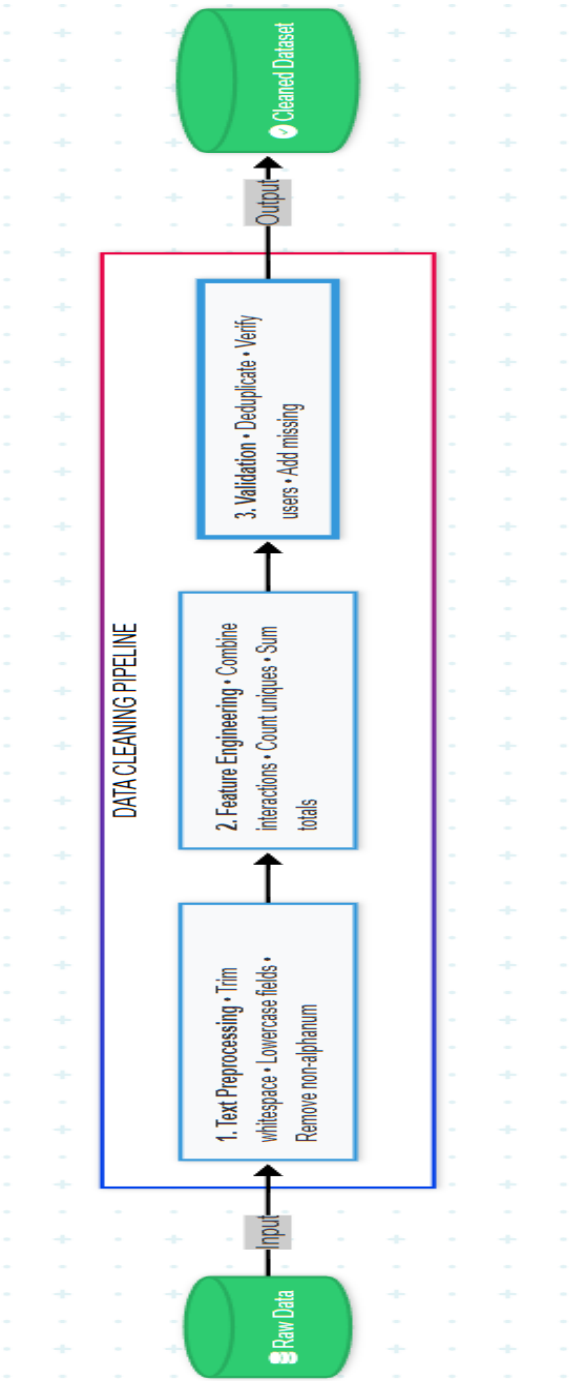
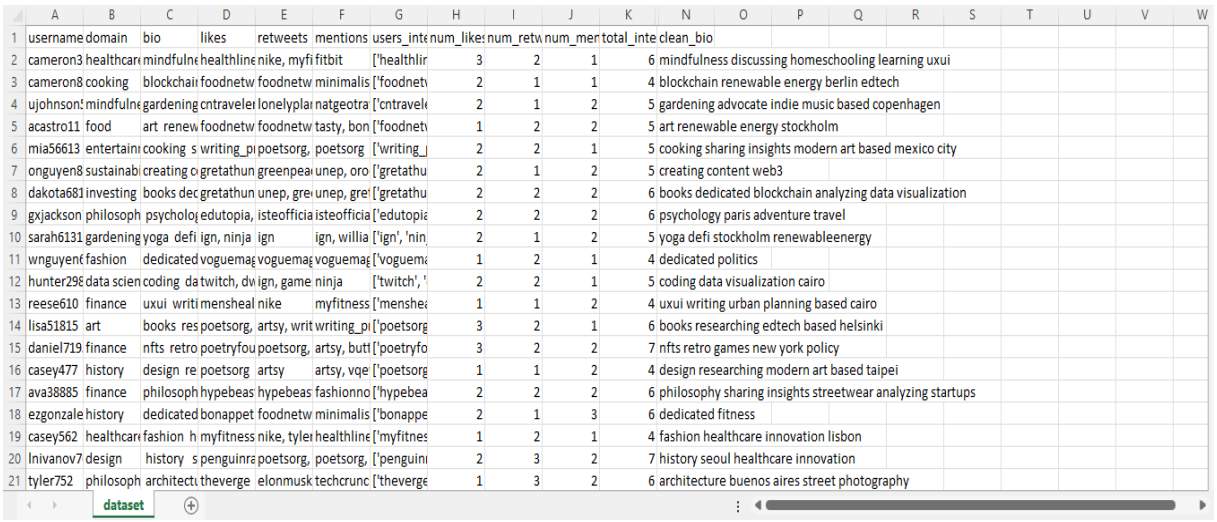


Figure 4.5 Architecture of data Preprocessing and cleaning

Part of dataset after data preprocessing and cleaning in **Figure 4.6** below :



	A	B	C	D	E	F	G	H	I	J	K	N
1	username	domain	bio	likes	retweets	mentions	users_inte	num_likes	num_retw	num_men	total_inte	clean_bio
2	cameron3	healthcar	mindfuln	healthline	nike, myfi	fitbit	['healthlir	3	2	1	6	mindfulness discussing homeschooling learning uxui
3	cameron8	cooking	blockchai	foodnetw	foodnetw	minimalis	['foodnet	2	1	1	4	blockchain renewable energy berlin edtech
4	ujohnson	mindfuln	gardening	cntravele	lonelypla	natgeotra	['cntravel	2	1	2	5	gardening advocate indie music based copenhagen
5	acastro11	food	art renew	foodnetw	foodnetw	tasty, bon	['foodnet	1	2	2	5	art renewable energy stockholm
6	mia56613	entertain	cooking s	writing_pi	poetsorg,	poetsorg	['writing_	2	2	1	5	cooking sharing insights modern art based mexico city
7	onguyen8	sustainab	creating c	gretathun	greenpea	unep, oro	['gretathu	2	1	2	5	creating content web3
8	dakota681	investing	books dec	gretathun	unep, gre	unep, gre	['gretathu	2	2	2	6	books dedicated blockchain analyzing data visualization
9	gxjackson	philosoph	psycholo	edutopia,	isteofficia	isteofficia	['edutopia	2	2	2	6	psychology paris adventure travel
10	sarah6131	gardening	yoga defi	ign, ninja	ign	ign, willia	['ign', 'nin	2	1	2	5	yoga defi stockholm renewableenergy
11	wnguyen	fashion	dedicated	voguemag	voguemag	voguemag	['voguema	1	2	1	4	dedicated politics
12	hunter298	data scien	coding da	twitch, dv	ign, game	ninja	['twitch', '	2	2	1	5	coding data visualization cairo
13	reese610	finance	uxui writi	mensheal	nike	myfitness	['menshea	1	1	2	4	uxui writing urban planning based cairo
14	lisa51815	art	books res	poetsorg,	artsy, writ	writing_pi	['poetsorg	3	2	1	6	books researching edtech based helsinki
15	daniel719	finance	nfts retro	poetryfou	poetsorg,	artsy, butt	['poetryfo	3	2	2	7	nfts retro games new york policy
16	casey477	history	design re	poetsorg	artsy	artsy, vqe	['poetsorg	1	1	2	4	design researching modern art based taipei
17	ava38885	finance	philosoph	hypebeas	hypebeas	fashionno	['hypebea	2	2	2	6	philosophy sharing insights streetwear analyzing startups
18	ezgonzale	history	dedicated	bonappet	foodnetw	minimalis	['bonappe	2	1	3	6	dedicated fitness
19	casey562	healthcar	fashion h	myfitness	nike, tyler	healthline	['myfitnes	1	2	1	4	fashion healthcare innovation lisbon
20	Inivanov7	design	history s	penguinra	poetsorg,	poetsorg,	['penguini	2	3	2	7	history seoul healthcare innovation
21	tyler752	philosoph	architect	theverge	elonmusk	techcrunc	['theverge	1	3	2	6	architecture buenos aires street photography

dataset

Figure 4.6 Part of dataset after Preprocessing

4.2.2 Visualization of different characteristics

The following images lay out a general overview of user activity, content attributes, and user interactions in different domains. This information came from different ways, summarizing user data, including total interactions per user, length of biographies, user types (sources vs. sinks), patterns by domain, and user biography word clouds.

- Bar chart in **Figure 4.7** visualizes the distribution of total interactions across different user groups, categorized by the number of total interactions per user
- Line plot visualizes in **Figure 4.8** the average count of different interaction types through various domains
- **Figure 4.9** Kernel Density Estimation (KDE) plot visualizes the distribution of bio lengths (measured in words) over a dataset
- Word cloud in **Figure 4.10** visualization provides an overview of the most frequently appearing words in Twitter biographies
- Pie chart in **Figure 4.11** represents distribution of user interaction roles in the dataset. The chart categorizes users in three distinct groups based on their interaction patterns

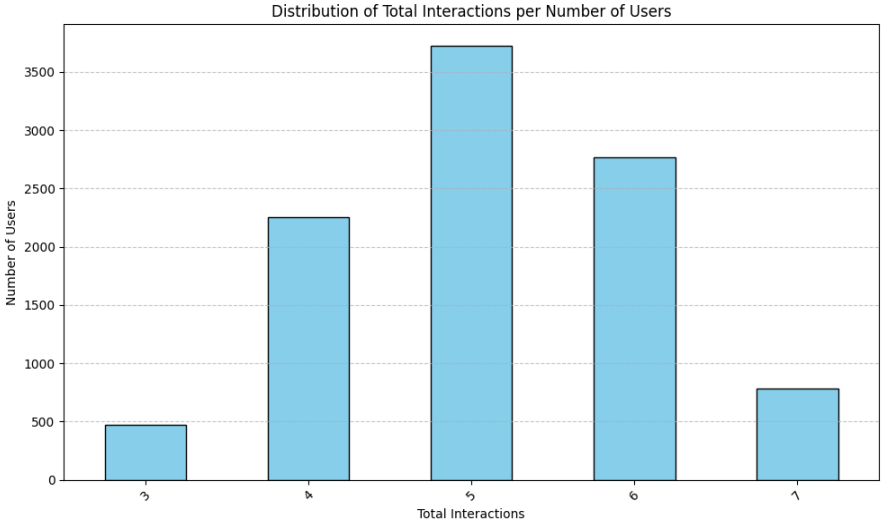


Figure 4.7 Distribution of Total Interactions per Number of Users

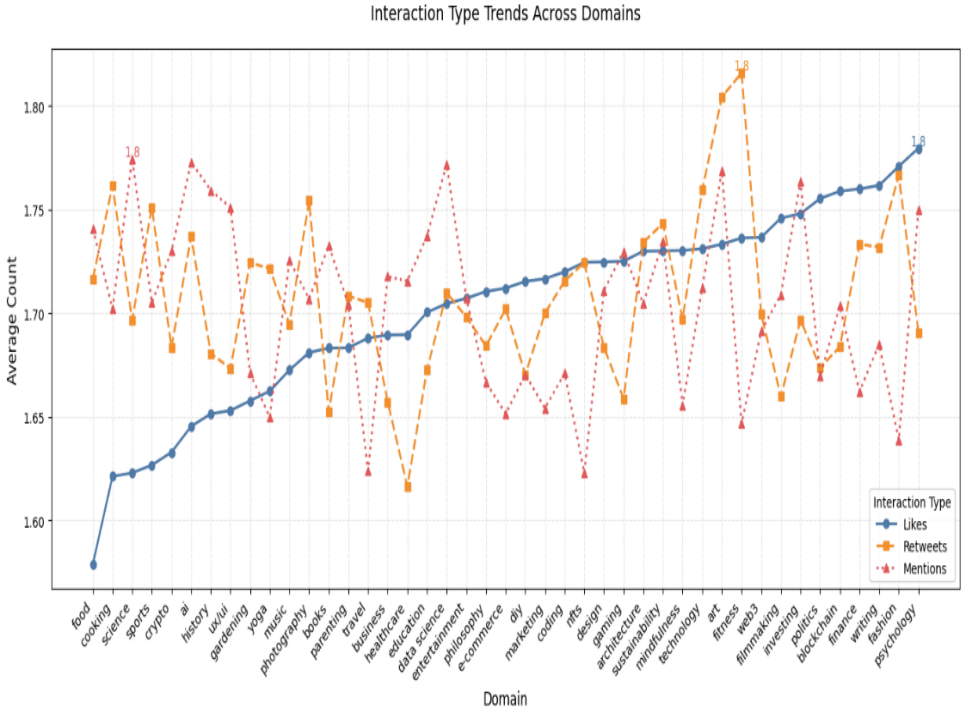


Figure 4.8 Interaction Type Trends Across Domain

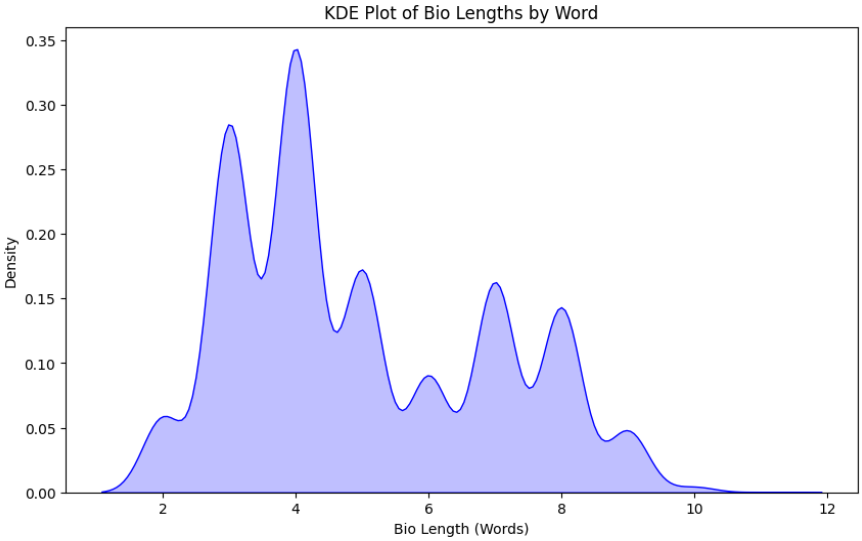


Figure 4.9 KDE Plot of Bio Lengths by Word

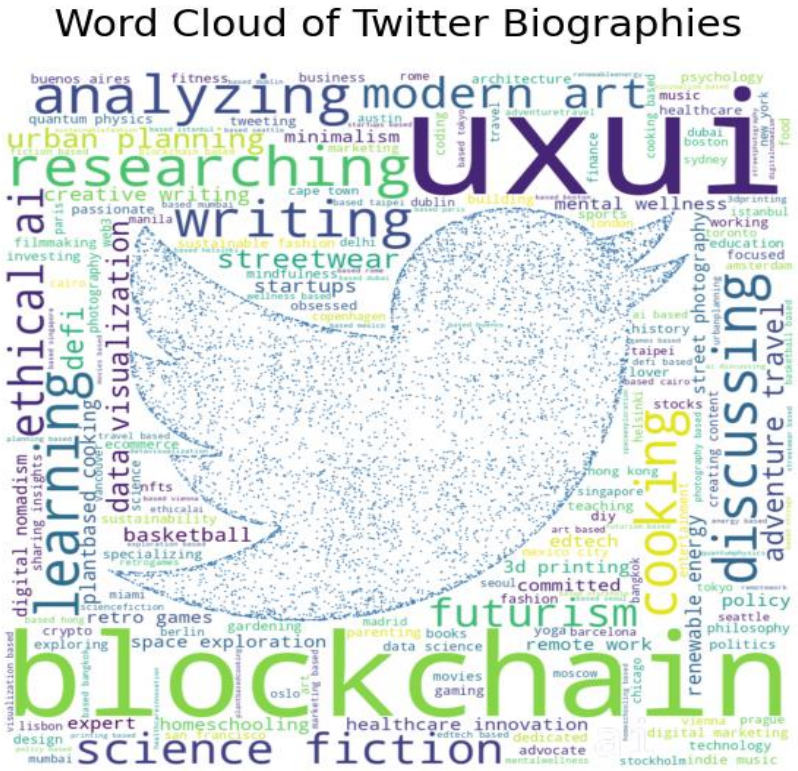


Figure 4.10 Word Cloud of Twitter biographies

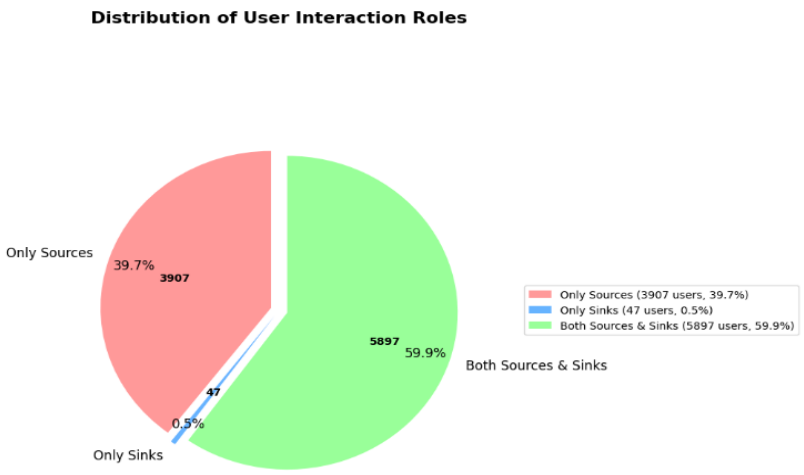


Figure 4.11 Distribution of User Interaction Roles

4.3 Experimental Results

4.3.1 Results for classic metrics

The table presents three clustering algorithms (KMeans, Hierarchical, Spectral) executed with similarity measures, Cosine, Jaccard, Euclidean, and then evaluated with clustering metrics; Silhouette Score, Davies-Bouldin Score, Calinski-Harabasz Score, Mean Cluster Size, Max/Min Cluster Size Ratio.

KMeans clustering, with Cosine similarity performed the best, with highest Silhouette Score .676 with the lowest Davies-Bouldin Score .072 indicating compact, and well-separated clusters.

The Hierarchical and Spectral clustering had highly imbalanced clusters (very high Max/Min ratios) particularly Euclidean, and Cosine similarities.

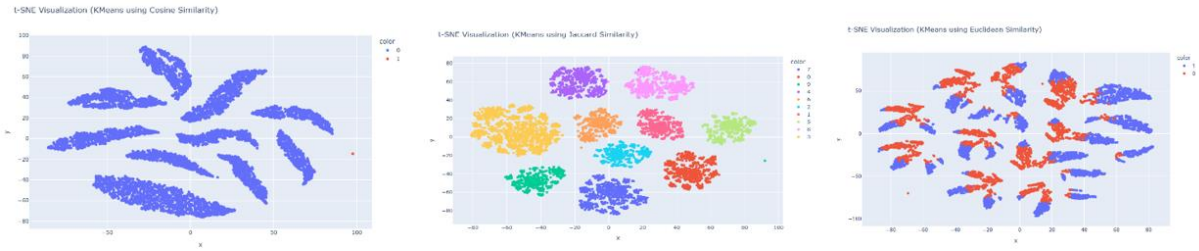
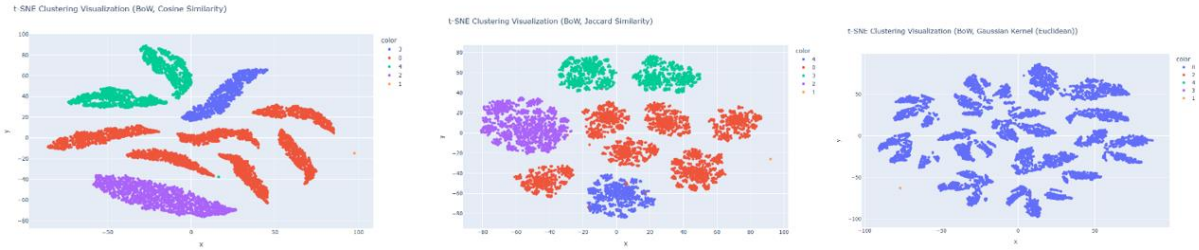
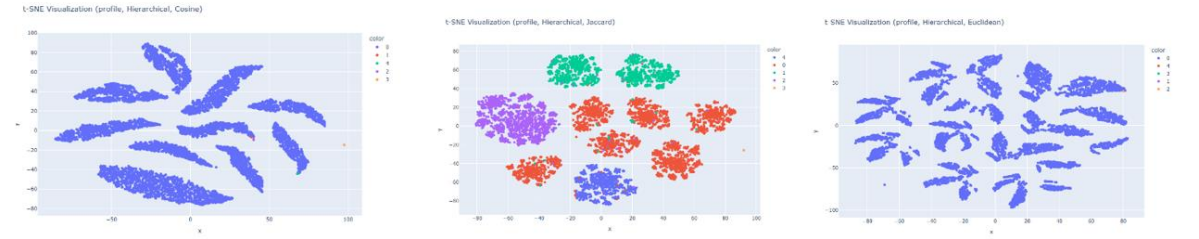
The Euclidean similarity always generated limits to the clustering structures indicating lower Silhouette and high Davies-Bouldin more difficult clustering without meaning.

Table 4.6 summarizes the clustering performance using different classical similarity measures and **Table 4.7** shows its Visualization.

Table 4.6 Clustering Performance of Algorithms Using Different classical Similarity Measures

Algorithm	Similarity	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score	Mean Cluster Size	Max/Min Cluster Size Ratio
KMeans	Cosine	0.676	0.072	8068.623	4925.5	208.60
	Jaccard	0.131	1.222	1684.975	985.1	2.79
	Euclidean	-0.036	1.637	3155.403	4925.5	1.39
Hierarchical	Cosine	0.271	0.583	2046.423	1970.2	9798.00
	Jaccard	0.071	1.647	1045.360	1970.2	100.00
	Euclidean	0.095	0.510	3.370	1970.2	9847.00
Spectral	Cosine	0.138	2.140	2996.619	1970.2	100.21
	Jaccard	0.073	1.629	1324.020	1970.2	100.49
	Euclidean	-0.269	2.288	58.203	1970.2	9829.00

Table 4.7 T-SNE Visualization of Clustering Performance of Algorithms Using Different classical Similarity Measures

KMeans (Cosine, Jaccard, Euclidean)

Spectral (Cosine, Jaccard, Euclidean)

Hierarchical (Cosine, Jaccard, Euclidean)


4.3.2 Results for embedding methods:

4.3.2.1 Results for GAT embedding:

The table shows GAT performance using clustering algorithms (KMeans, Hierarchical, and Spectral. And each method is evaluated with evaluation metrics: Silhouette Score, Davies-Bouldin Score, Calinski-Harabasz Score, Mean Cluster Size, and the ratio of Max/Min Cluster Size.

KMeans clustering Algorithm with Cosine similarity had the highest Silhouette Score (0.172). That indicates the clusters have moderate cohesion and separation.

Hierarchical clustering and Cosine similarity had the highest overall Silhouette Score 0.370, and lowest Davies-Bouldin Score of 0.377, demonstrating very strong cluster performance despite the extreme imbalance in size of the clusters.

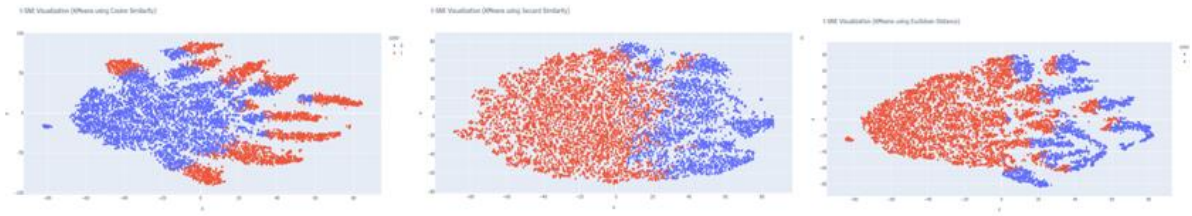
Spectral clustering generally performed the worst with very low Silhouette Scores and high Davies-Bouldin Scores, particularly with Jaccard and Euclidean similarities.

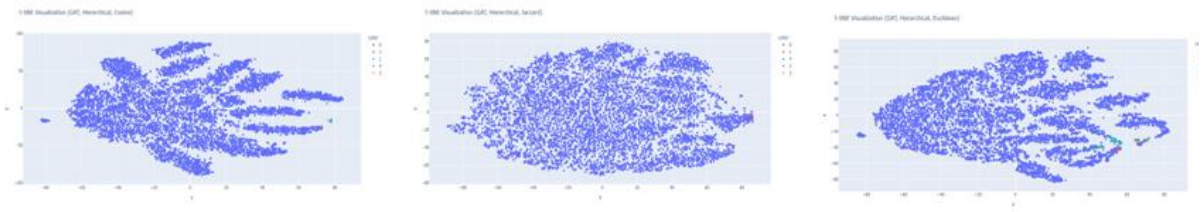
Table 4.8 presents the clustering results obtained using GAT embeddings across different algorithms and similarity measures and below **Table 4.9** T-SNVisualization.

Table 4.8 Clustering results using GAT embeddings

Algorithm	Similarity	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score	Mean Cluster Size	Max/Min Cluster Size Ratio
KMeans	Jaccard	0.130	1.280	5649.257	4925.5	1.67
	Euclidean	0.159	1.028	8053.226	4925.5	2.08
	Cosine	0.172	1.249	5890.134	4925.5	1.75
Hierarchical	Jaccard	0.298	1.586	34.096	1970.2	9827.00
	Euclidean	0.283	1.640	85.007	1970.2	4898.50
	Cosine	0.370	0.377	45.454	1970.2	9825.00
Spectral	Jaccard	0.073	3.267	628.648	1970.2	3.08
	Euclidean	0.083	2.279	1596.955	1970.2	6.54
	Cosine	0.106	2.192	1043.439	1970.2	2.69

Table 4.9 T-SNVisualization of clustering results using GAT embeddings

K-Means (Cosine, Jaccard, Euclideon)

Spectral (Cosine, Jaccard, Euclideon)

Herachical (Cosine, Jaccard, Euclideon)


4.3.2.2 Results for GCN embedding

The table displays performance of GCN using three clustering algorithms (Hierarchical, KMeans, Spectral) and evaluating using Silhouette Score, Davies-Bouldin Score, Calinski-Harabasz Score, Mean Cluster Size, and Max/Min Cluster Size Ratio with Jaccard, Euclidean, and Cosine similarities.

In general, KMeans with Euclidean was the best in terms of clustering quality, the largest Silhouette score (0.980) and the lowest Davies-Bouldin score (0.127), indicating that the quality of clustering was excellent .

Hierarchical clustering with Euclidean similarity distance, also performed strongly, with high Silhouette Score (0.931) and matching Davies-Bouldin Score (0.127), but showed large imbalance in cluster sizes (Max/Min=208.60).

Spectral clustering with Jaccard similarity produced balanced clusters (Max/Min = 1.07), with the lowest Silhouette Score (0.088), and bad clustering separation.

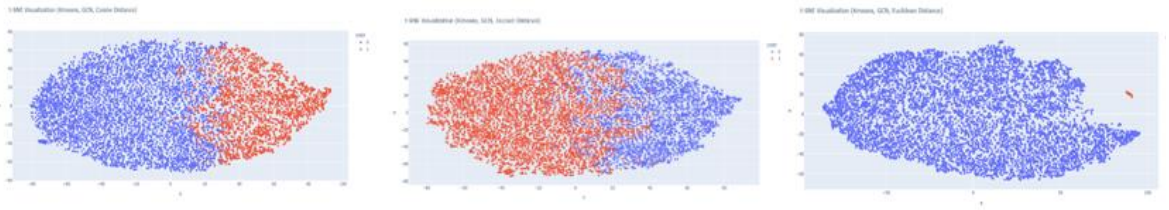
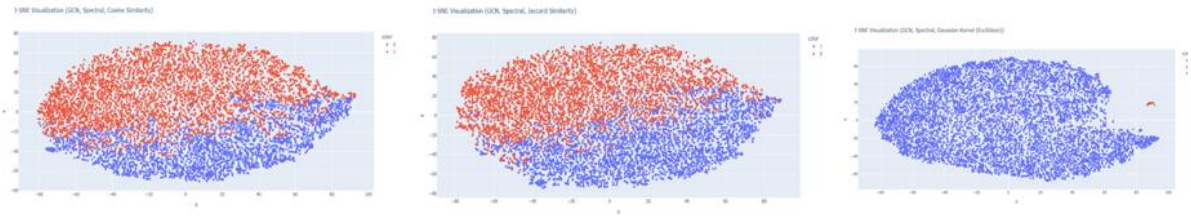
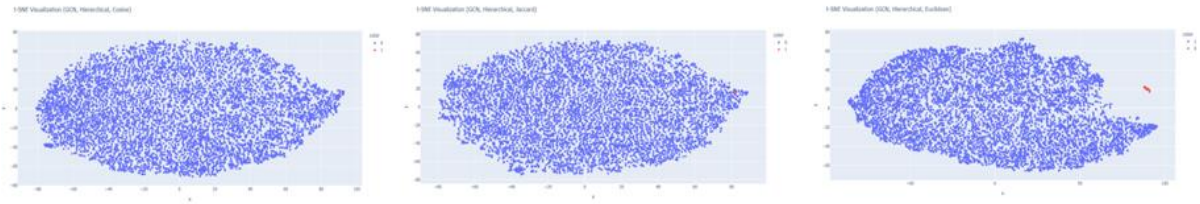
Hierarchical clustering with Cosine similarity had moderate quality (Silhouette = 0.478) but imbalance (Max/Min = 9850.00), making it less practical.

Table 4.10 presents the clustering results obtained using GCN embeddings with various clustering algorithms and similarity metrics and **Table 4.11** show the T-SNE Visualization

Table 4.10 Clustering results using GCN embeddings

Algorithm	Similarity	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score	Mean Cluster Size	Max/Min Cluster Ratio
Hierarchical	Jaccard	0.237	0.690	13.006	4925.5	3282.67
	Euclidean	0.931	0.127	136655.594	4925.5	208.60
	Cosine	0.478	0.290	10.506	4925.5	9850.00
KMeans	Jaccard	0.172	2.026	2302.623	4925.5	1.52
	Euclidean	0.980	0.127	136655.594	4925.5	208.60
	Cosine	0.329	1.335	4975.850	4925.5	2.30
Spectral	Jaccard	0.088	2.536	1470.256	4925.5	1.07
	Euclidean	0.766	0.301	2099.071	3283.7	1400.57
	Cosine	0.160	2.433	1411.850	4925.5	1.42

Table 4.11 T-SNE Visualization of Clustering results using GCN embeddings

KMeans (Cosine, Jaccard, Euclideon)

Spectral (Cosine, Jaccard, Euclideon)

Hierarchical (Cosine, Jaccard, Euclideon)


4.3.2.3 Results for GraphSAGE embedding

The table demonstrates the performance of Graghsage using Hierarchical, KMeans, and Spectral clustering for Jaccard, Euclidean, and Cosine similarities. The evaluation metrics used Silhouette Score, Davies-Bouldin Score, Calinski-Harabasz Score, Average cluster size, and Max/Min cluster size ratio.

Hierarchical clustering metric with Euclidean similarity has very high performance with the highest Silhouette Score (0.931) and the lowest Davies-Bouldin Score (0.127). Although, it had very unbalanced clusters (Max/Min = 208.60).

KMeans clustering metric with Cosine similarity was the best , with a high Silhouette Score (0.626) and low Davies-Bouldin Score (0.365), that reflect good clustering quality.

Spectral clustering metric with Jaccard similarity achieves the most balanced clusters (Max/Min = 1.07) but shows poor clustering quality (Silhouette Score = 0.088).

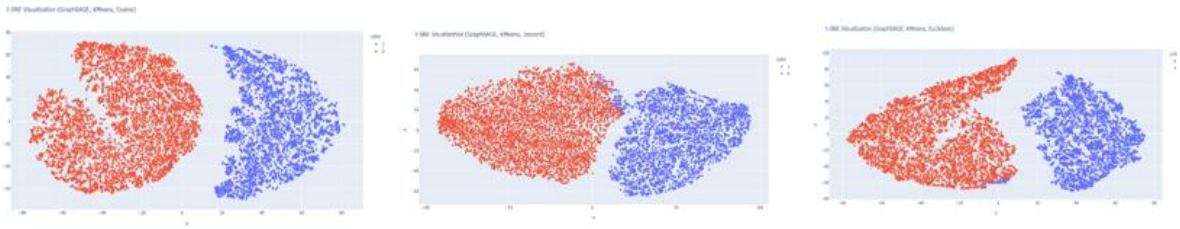
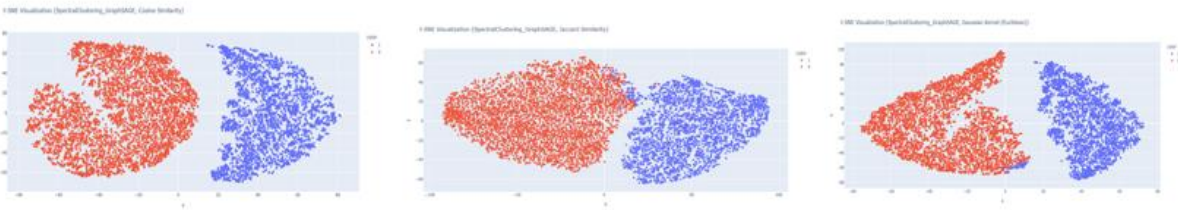
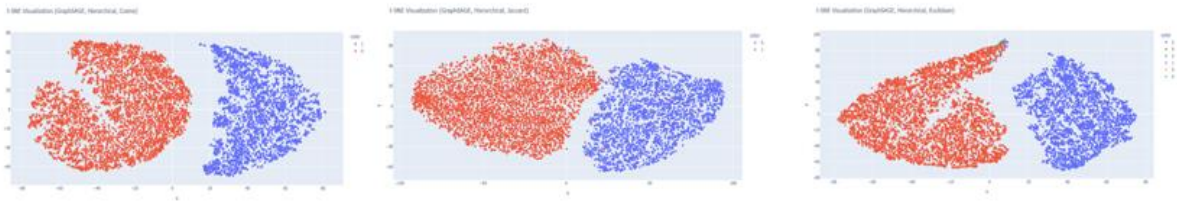
Cosine similarity with Hierarchical clustering was extreme imbalance (Max/Min = 9850.00), with decent Silhouette Score (0.478).

Table 4.12 summarizes the clustering performance using GraphSAGE embeddings across different algorithms and similarity measures and under it **Table 4.13** shows the Visualization.

Table 4.12 clustering performance using GraphSAGE embeddings

Algorithm	Similarity	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score	Mean Cluster Size	Max/Min Cluster Ratio
Hierarchical	Jaccard	0.237	0.690	13.006	4925.5	3282.67
	Euclidean	0.931	0.127	136655.594	4925.5	208.60
	Cosine	0.478	0.290	10.506	4925.5	9850.00
KMeans	Jaccard	0.330	0.716	17540.167	4925.5	1.37
	Euclidean	0.415	0.402	32518.490	4925.5	1.42
	Cosine	0.626	0.365	58749.902	4925.5	1.47
Spectral	Jaccard	0.088	2.536	1470.256	4925.5	1.07
	Euclidean	0.766	0.301	2099.071	3283.7	1400.57
	Cosine	0.160	2.433	1411.850	4925.5	1.42

Table 4.13 T-SNE Visualization clustering performance using GraphSAGE embeddings

KMeans (Cosine, Jaccard, Euclideon)

Spectral (Cosine, Jaccard, Euclideon)

Hierarchical (Cosine, Jaccard, Euclideon)


4.3.3 Overall Comparison

Table 4.14 Overall comparison of evaluation metrics

Metric	Best Value	Embedding / Feature Set	Algorithm	Similarity
Silhouette Score	0.980	GCN	KMeans	Euclidean
Davies-Bouldin Score	0.072	Classic	KMeans	Cosine
Calinski-Harabasz Score	136655.6	GCN / GraphSAGE	KMeans / Hierarchical	Euclidean
Cluster Ratio	1.47	GraphSAGE	KMeans	Cosine

In the case of silhouette score, the best option is GCN + KMeans + Euclidean , it has the highest silhouette score among the other combinations. Thus, it had the best cluster cohesion and cluster separation based on the distance formula.

In Davies-Bouldin score, the best option is Classic + KMeans + Cosine, which provided the best Davies-Bouldin score with low intra-cluster dispersion and a high inter-cluster distance which showed the cluster separation.

But in Calinski-Harabasz score, the best option is GCN/GraphSAGE + KMeans/Hierarchical + Euclidean , this provided the highest calinski-harabasz .

Finally in Cluster Ratio (Max/Min cluster size) the best option was GraphSAGE + KMeans + Cosine, this produced the best (lowest) Cluster Ratio suggesting that more balance was occurring with cluster sizes.

4.4 Discussion

4.4.1 Comparative Analysis of Results

The comparative assessment shows that embedding-based deep learning approaches on graphs outperformed traditional hand-crafted features using classical similarity metrics. Consider the following comparisons:

Traditionally, Cosine, Jaccard, and Euclidean similarity metrics using the raw features (e.g., bag of words, interaction degrees) performed reasonably with KMeans + Cosine having the best

Silhouette score of 0.676. These metrics did not account for higher-order relationships, which have a role in less cohesive clusters, especially in hierarchical or spectral methods. Embedding techniques using GCN, GAT, and GraphSAGE incorporated both structural attribute-level patterns into low-dimensional representations where KMeans formed compact and separate clusters (e.g., GCN + KMeans + Euclidean: Silhouette: 0.980). The embeddings were able to go beyond local neighbourhoods letting similarity computations check homophily, connectivity, and interaction patterns in the graph which we do not have in traditional metrics.

Among all evaluated combinations, the most discriminating method – the one that most effectively distinguishes between user groups – is the combination of GCN embeddings with KMeans clustering using Euclidean similarity. This setup achieved the highest Silhouette score (0.980) and the highest Calinski-Harabasz score (136655.594); on the other hand, the Davies-Bouldin score was very low (0.127). These are strong indicators of well-separated and compact clusters.

What distinguishes this method is its ability to learn local and global graph structures using GCN, which is an iterative feature aggregation method from neighbouring nodes in the graph. This means it can encode complex patterns of user similarity that were not easy to discern using classic feature sets or simple similarity functions. Although in this low-dimensional embedding space Euclidean distance is used, the embeddings learned by GCN are optimized to bring like-users closer in terms of Euclidean distance.

Although GraphSAGE and GAT also had superior performance over classic features, their results were slightly worse than GCN clustering quality. GAT, with its attention mechanism, didn't outperform the others, most likely due to the fact that the synthetic dataset was not heterogeneous enough for the attention mechanism to confer a particular advantage.

Therefore, GCN + KMeans + Euclidean is the most discriminating method among others. It achieves the best balance across all metrics and shows the strength of graph-based deep learning demonstrating nuanced user groupings in online social networks.

4.4.2 Most Effective GNN Model

Based on the analyses of the three GNN models the GCN (Graph Convolutional Network) provided the highest clustering performance consistently of the three GNN models reviewed:

GCN performed best in Silhouette and Calinski-Harabasz scores with KMeans and Euclidean. and its ability to smooth node features across a node's neighbors and tends to perform well when homophily exists in node features.

GraphSAGE performs well for KMeans and Cosine similarity: It achieved balanced clustering and solid CH scores. Its ability to aggregate around sampled neighborhoods lends itself to scalability, especially for large graphs like Twitter.

GAT (Graph Attention Network) scores lower than GCN: provides weights of districts with its attention, but GAT may over fit, suffer performance, or worst-case performance on sparse graphs or synthetic structures where the importance associated with individually sampled nodes is consistent or noisy. It is best suited in settings of heterogeneous graphs or where importance associated with nodes vary and attention is helpful.

The GCN consistently performed best , maintaining a balance between preservation of the graph (structure), preserving attributes (propagating), and separability.

4.4.3 Work Limitations

- Synthetic Dataset: The dataset is representative of Twitter interactions, but it does not allow for the same level of noise and complexity of real-world social networks, meaning we can not generalize to production environments without testing it on real data.
- No Ground-Truth Labels: Our evaluations faced challenges based on the lack of any actual user source group labels or community memberships, leading us to rely entirely on internal only clustering metrics (Silhouette, DB, CH), leaving us unable to measure clustering quality in terms that actually have meaning to real life (e.g., topic coherence, interest similarity).

4.4.4 Future Improvements

- Supervised Learning (If Labels Are Available): When community or user interest label(s) may be found in the data, we could take advantage of supervised node classification or contrastive learning to iteratively fine-tune the embeddings.
- Addition of More Modalities: Only structural and interaction-based features are considered in the analysis. Future work could include: Tweet text content, by applying BERT embeddings. Patterns of activity over time (i.e. frequency of tweeting over an amount of time). User profile metadata (e.g., the user's own location, hashtags related

to twitter account, domains related to twitter account). Adding these modalities would greatly add to the richness of embedding semantics and gain more meaningful clusters. Evaluation against Community Detection Benchmarking: Evaluating, especially for real-world datasets with existing known communities (e.g., PubMed datasets, Reddit, SNAP Twitter datasets), so that we could evaluate against standard benchmark tasks for graphs.

4.5 Implications

This research contributes to the concept of user similarity in Online Social Networks (OSNs) by demonstrating that graph-based deep learning models, specifically Graph Convolutional Networks (GCNs), outperform traditional methods in clustering users based on their profiles and interactions.The results have some important implications:

- ✓ Better User Recommendations and Communities: By being able to model complicated relationships between users, graph-based embeddings will improve the possibility to deliver accurate friend recommendations, community detection and personalize dcontents.
- ✓ Scalable User Similarity Approach: The proposed methodology represents a scalable and generalizable process for computing user similarity via unsupervised Graph Neural Networks, with applicability to real-world OSNs.
- ✓ Structured Evaluation Pipeline: The study provides a structured evaluation pipeline that applies internal evaluation metrics (Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Score) to compare traditional clustering methods and clustering based on deep learning, making a good reference for other studies that evaluate similar clustering or grouping of users.

4.6 Conclusion

In this chapter, we provided a thorough evaluation of user similarity measures in Online Social Networks based on a synthetic Twitter dataset including 10,000 users. The experiments compared standard similarity measures (e.g., Cosine, Jaccard, and Euclidean) with contemporary graph-based embedding measures derived from Graph Neural Networks (i.e., GCN, GraphSAGE, and GAT).

Conclusion

In this research, we presented a comprehensive comparison of traditional feature based similarity measures and modern graph embedding techniques to extract user similarity in Online Social Networks. Using a synthetic Twitter dataset, we showed the ability of clustering users using their profiles, behaviours and interaction.

The outcome illustrates that graph-based techniques, specifically GCN embeddings with KMeans clustering and Euclidean distance produced better quality measures across all metrics. These methods captured the user's local and global relationships in the network.

Despite limitations of synthetical datasets (e.g., no ground-truth labels), the most important result was that graph representation learning is an effective way to identify structures of regularity in social networks. Future research will hopefully compare real datasets, study the multi-modal fusion of data (e.g., sentiment scoring of tweet content, the influence of time, etc.) and develop supervised models to explore the quality and interpretability of embeddings.

This study gives exploration to the use of deep learning on graphs as a method for analysing user similarity and community discovery in the dynamic environment of online social networks that we hope to provide new insights and establish a foundation for future research.

References

Bhaskaran, R., Kannan, R., Barr, B., & Priebe, S. (2021). Science-Guided Machine Learning for Wall-Modeled Large Eddy Simulation. *2021 IEEE International Conference on Big Data (Big Data)*, 1809–1816. https://doi.org/10.1109/BigData52589.2021.9671436

Bisong, E. (2019). Google Colaboratory. In E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 59–64). Apress. https://doi.org/10.1007/978-1-4842-4470-8_7

Boyd, D. M., & Ellison, N. B. (2007). Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, *13*(1), 210–230. https://doi.org/10.1111/j.1083-6101.2007.00393.x

Calinski, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, *3*(1), 1–27. https://doi.org/10.1080/03610927408827101

Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224–227. https://doi.org/10.1109/TPAMI.1979.4766909

Ferres, L., Rossi, G., Almeida, V., & Herder, E. (2014, September). Proceedings of the 25th ACM Conference on Hypertext and Social Media. *Proceedings of the 25th ACM Conference on Hypertext and Social Media*. HT '14: 25th ACM Conference on Hypertext and Social Media, Santiago Chile.

Fey, M., & Lenssen, J. E. (2019). *Fast Graph Representation Learning with PyTorch Geometric* (Version 3). arXiv. https://doi.org/10.48550/ARXIV.1903.02428

García-Alonso, C. R., Pérez-Naranjo, L. M., & Fernández-Caballero, J. C. (2014). Multiobjective evolutionary algorithms to identify highly autocorrelated areas: The

case of spatial distribution in financially compromised farms. *Annals of Operations Research*, *219*(1), 187–202. https://doi.org/10.1007/s10479-011-0841-3

Grover, A., & Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864. https://doi.org/10.1145/2939672.2939754

Hamilton, W. L., Ying, R., & Leskovec, J. (2017a). *Inductive Representation Learning on Large Graphs* (Version 4). arXiv. https://doi.org/10.48550/ARXIV.1706.02216

Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). *Representation Learning on Graphs: Methods and Applications* (Version 3). arXiv. https://doi.org/10.48550/ARXIV.1709.05584

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Itkin, A., & Soleymani, F. (2021). Four-factor model of Quanto CDS with jumps-at-default and stochastic recovery. *Journal of Computational Science*, *54*, 101434. https://doi.org/10.1016/j.jocs.2021.101434

Jia, H., Ding, S., Xu, X., & Nie, R. (2014). The latest research progress on spectral clustering. *Neural Computing and Applications*, *24*(7–8), 1477–1486. https://doi.org/10.1007/s00521-013-1439-2

Kaplan, A. M., & Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of Social Media. *Business Horizons*, *53*(1), 59–68. https://doi.org/10.1016/j.bushor.2009.09.003

Kipf, T. N., & Welling, M. (2016). *Semi-Supervised Classification with Graph Convolutional Networks* (Version 4). arXiv. https://doi.org/10.48550/ARXIV.1609.02907

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: An overview. *WIREs Data Mining and Knowledge Discovery*, *2*(1), 86–97. https://doi.org/10.1002/widm.53

Najadat, H. M., Alshboul, A. A., & Alabed, A. F. (2019). Arabic Handwritten Characters Recognition using Convolutional Neural Network. *2019 10th International Conference on Information and Communication Systems (ICICS)*, 147–151. https://doi.org/10.1109/IACS.2019.8809122

Oti, E. U., Olusola, M. O., Eze, F. C., & Enogwe, S. U. (2021). Comprehensive Review of K-Means Clustering Algorithms. *International Journal of Advances in Scientific Research and Engineering*, *07*(08), 64–69. https://doi.org/10.31695/IJASRE.2021.34050

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (No. arXiv:1912.01703). arXiv. https://doi.org/10.48550/arXiv.1912.01703

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É.

(2018). *Scikit-learn: Machine Learning in Python* (No. arXiv:1201.0490). arXiv. https://doi.org/10.48550/arXiv.1201.0490

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710. https://doi.org/10.1145/2623330.2623732

Rodrigues, C. K. D. S., & Rocha, V. (2021). Enhancing BitTorrent for efficient interactive video-on-demand streaming over MANETs. *Journal of Network and Computer Applications*, *174*, 102906. https://doi.org/10.1016/j.jnca.2020.102906

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Singh, K., Seth, A., Sandhu, H. S., & Samdani, K. (2019). A Comprehensive Review of Convolutional Neural Network based Image Enhancement Techniques. *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 1–6. https://doi.org/10.1109/ICSCAN.2019.8878706

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2017). *Graph Attention Networks* (Version 3). arXiv. https://doi.org/10.48550/ARXIV.1710.10903

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2018). *Deep Graph Infomax* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1809.10341

W3Schools. (n.d.). *W3Schools*. https://www.w3schools.com/python/pandas/pandas_intro.asp

Zafarani, R., Abbasi, M. A., & Liu, H. (2014). *Social Media Mining: An Introduction* (1st ed.). Cambridge University Press. https://doi.org/10.1017/CBO9781139088510

Zhang, Z., Cui, P., & Zhu, W. (2022). Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 249–270. https://doi.org/10.1109/TKDE.2020.2981333