# Identify clusters using (Node2Vec Embedding, Spectral, and GCN)

Aabisheg T (es21btech11001)
Ranveer Sahu (es21btech11025)
S Jagadeesh (es21btech11026)
Subiksha Gayathiri KK (es21btech11031)
Bhende Adarsh Suresh (cs21btech11008)

## Abstract

*Node classification on popular social network datasets in the graph setting arises in various real world networks. Being able to label a particular entity in a graph based on its neighboring graph structure and predicting relationships between entities plays an important role in analyzing social networks and content on the web. With the the resurgence of machine learning, supervised learning problems have been shown to accomplish a number of different tasks from classifying animals to determining the model of cars [7]. These methods have been further extended by the resurgence of Deep Learning to accomplish the same tasks such as classifying images with higher accuracy. However, it is noted by Leskovec et al. [2] that prediction tasks on graphs require careful feature engineering. In this paper we adapt work by Kipf et al [4][5] in order to leverage Graph Convolutional Neural Networks as a means of evaluating spectral embeddings in comparison to embeddings generated by the Node2Vec algorithm.*

## 1 Problem Introduction

In the era of burgeoning social media platforms, the volume of user-generated data has reached unprecedented levels. This wealth of data presents a valuable opportunity to harness its potential for the betterment of society. Social graphs, underlying the fabric of these platforms, encapsulate rich information about human interactions and behaviors. They serve as fertile ground for exploration and analysis, offering insights into the intricate dynamics of social networks. This project endeavors to leverage graph embedding techniques, specifically node2vec and spectral embedding, to glean actionable intelligence from social graphs. By harnessing the structural properties encoded within these networks, we aim to decipher patterns of human behavior and preferences. Our focus lies in predicting node label assignments through a classification framework, thereby illuminating the extent to which network structures alone can inform predictive modeling. The algorithms under scrutiny include the node2vec random walk algorithm, a rudimentary spectral embedding method, and the advanced graph convolutional networks algorithm. Through meticulous experimentation and evaluation, we seek to ascertain the efficacy of these algorithms in addressing classification tasks within social graphs. Furthermore, we endeavor to identify avenues for enhancement and refinement, thereby pushing the boundaries of predictive modeling in social network analysis.

## 2 Review of Prior Relevant Work

With the growing interest in machine learning and graph analytics, various techniques have been developed to address classification and prediction tasks on graphs. However, as noted by Leskovec et al. [2], prediction tasks on graphs necessitate careful feature engineering to achieve accurate results. In this paper, we draw inspiration from the work of Kipf et al. [4][5], who introduced Graph Convolutional Neural Networks (GCNs) as a powerful framework for learning representations of nodes in graphs. Specifically, we adapt their approach to evaluate node embeddings generated from Node2Vec and Spectral Embedding techniques. Grover et al. [2] also contribute to this area by proposing a novel method for learning continuous feature representations for nodes in networks.

In addition to GCNs, Rossi et al. [9] have demonstrated the effectiveness of deep feature learning for graphs through their framework, DeepGL, which computes a hierarchy of graph representations. Moreover, unsupervised methods for learning embeddings, such as learning subgraph embeddings [6], have been explored to capture latent structural information within graphs. It is worth noting that while Node2Vec is a popular technique for node embeddings, it may not perform optimally in tasks involving structural equivalence due to the prevalence of strong homophily in many networks [8]. In such cases, alternative approaches that consider the latent representation of the network may yield better results. Furthermore, traditional techniques in graph analysis have focused on studying the eigenvalues and eigenvectors of graph matrices. Recent work by [1] has investigated leveraging eigenvectors of the Laplacian matrix to predict climate variability, demonstrating the effectiveness of spectral

methods in capturing underlying patterns in graphs. This suggests that the top-k eigenvectors derived from the normalized adjacency matrix could serve as effective embedding techniques, where each component corresponds to a position in the node feature vector.

### 2.1 Description of Data Collection Process

Our experiments are conducted using the "payments.csv" dataset obtained from internal financial records. This dataset contains sender-receiver relationships and transaction amounts. Extensive preprocessing was undertaken to ensure compatibility with our graph-based methods, including Graph Convolutional Networks (GCNs), Node2Vec algorithm, and visualization techniques like T-SNE plots.

To make the dataset compatible with spectral embedding extraction, we first extracted the adjacency matrix. The original CSV file storing edges was reformatted to be parseable by standard graph libraries. Subsequently, we utilized graph processing libraries to generate the adjacency matrix. Following normalization, we computed the top eigenvalues and eigenvectors. Leveraging the sparsity of the matrix, we optimized the computation process, significantly reducing runtime.

For compatibility with Graph Convolutional Networks, we developed parsers to read embeddings files generated by the Node2Vec and spectral embeddings algorithms. Additionally, class labels were transformed into one-hot encoded vectors. The entire matrix of node embeddings and labels, along with masked arrays for training, validation, and testing sets, was then fed into the GCN model.

Unlike traditional datasets, our "payments.csv" dataset presented unique challenges in preprocessing. We had to ensure that the sender-receiver relationships were accurately represented as an adjacency matrix. Additionally, the handling of transaction amounts required careful consideration to preserve meaningful graph structures.

Due to computational constraints and dataset size, we conducted experiments on a subset of the "payments.csv" data. This allowed us to maintain computational efficiency while still deriving valuable insights from the dataset.

In summary, the data collection process involved extensive preprocessing to ensure compatibility with our graph-based methods. This included formatting sender-receiver relationships, transaction amounts, and class labels for use in spectral embedding and GCN models. Despite the challenges posed by the "payments.csv" dataset, our preprocessing efforts enabled us to effectively leverage graph-based techniques for analysis and clustering tasks.

### 2.2 Dataset Analysis

The table below contains some basic information and analyses about the payments.csv dataset graph. We conducted exploratory analysis on the "payments.csv" dataset to gain insights into transaction patterns and network structures.

| | |
|---|---|
| Num nodes | 1074 |
| Num edges | 130535 |
| Avg clustering coefficient | 0.1250 |
| Max node degree | 6591 |

Table 1: Payments dataset values

This analysis involved examining the distribution of transaction amounts, node degrees, and visualizing the transaction network.

**Histogram of Transaction Amounts:** The histogram of transaction amounts (Figure 1) illustrates the distribution of transaction sizes within the dataset. The plot reveals common transaction amounts and highlights any outliers or unusual patterns in transaction sizes.

**Degree Distribution Plot:** The degree distribution plot (Figure 2) showcases the distribution of node degrees in the transaction network. This visualization helps identify nodes with high degrees, indicating central transaction hubs, as well as nodes with lower degrees, representing typical transaction patterns.

**Network Visualization:** The network visualization (Figure 3) provides a visual representation of the transaction network, where nodes represent entities (senders or receivers) and edges represent transactions between them. This visualization offers insights into the structure of the transaction network and the relationships between entities.

These plots collectively contribute to our understanding of the "payments.csv" dataset, revealing patterns and characteristics inherent in transaction data. Subsequent analyses and modeling efforts will build upon these insights to further explore and analyze transaction behaviors.

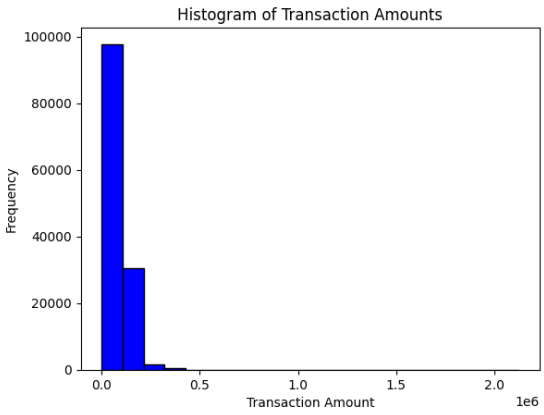


Figure 1: Histogram of Transactions

## 3 Mathematical Background

### 3.1 Node2Vec

We employed the Node2Vec algorithm to generate node embeddings, which served as feature representations for classi-

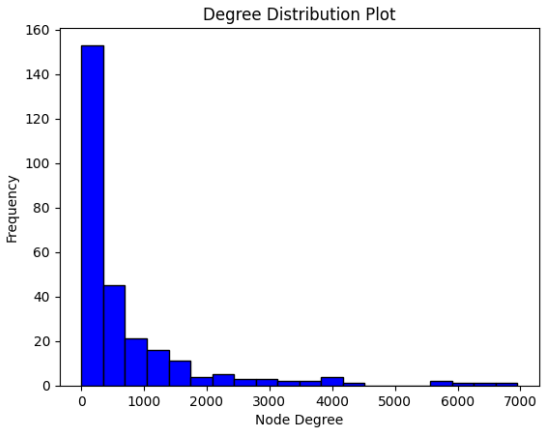


Figure 2: Degree Distribution Plot

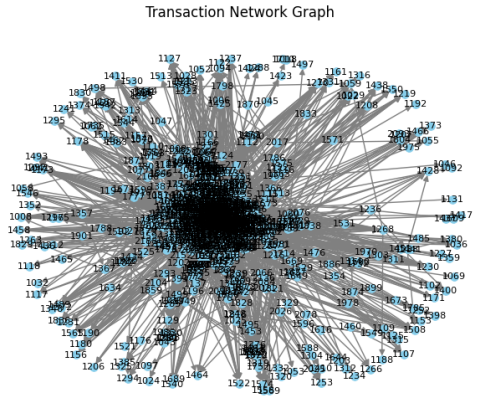


Figure 3: Transaction Network Graph

fication tasks on the "payments.csv" dataset. The primary objective was to classify nodes based on their transaction behaviors or characteristics.

The Node2Vec algorithm, introduced by Leskovec et al. [2], formulates feature learning in graphs as a maximum likelihood problem. They define a Networking Neighborhood $N_s(u) \subset U$ generated through various sampling strategies such as BFS, DFS, or Random Walk. The objective function of Node2Vec is formulated as follows:

$$\max_f \sum_{u \in V} \log Pr(N_s(u) \mid f(u))$$

Here, $u$ represents the node, and f is the mapping function from nodes to feature representations, which are learned during the training process.

**Two key assumptions underlie the formulation:**

**1**.The likelihood of observing a neighborhood node is independent of observing any other node. Mathematically, this can be expressed as:

$$\log \Pr(N_s(u) \mid f(u)) = \prod_{n \in N_s(u)} Pr(n_i \mid f(u))$$

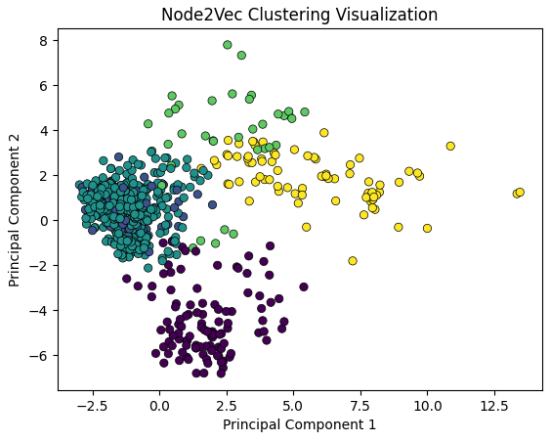


Figure 4: Plot For Node2Vec Embeddings

**2**.Nodes $n_1$and$n_2$ in the neighborhood have a symmetric effect on their features. This allows conditioning the likelihood of a neighborhood pair with the following equation:

$$\Pr(n_i \mid f(u)) = \frac{e^{f(n_i) \cdot f(u)}}{\sum_{v \in V} e^{f(n_i) \cdot f(u)}}$$

These assumptions enable Node2Vec to learn effective feature representations by capturing the local transaction patterns and node relationships present in the "payments.csv" dataset.

## 3.2 Spectral Embeddings

We refer to the following technique as the "Spectral Embedding Technique." It primarily involves extracting node embeddings by computing the top-k eigenvectors of the normalized adjacency matrix.

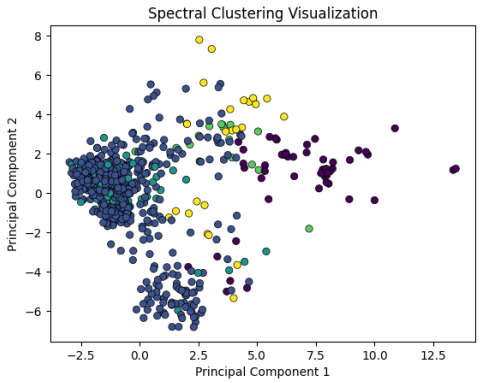


Figure 5: Enter Caption

**1.Normalization of the Adjacency Matrix:** We begin by computing the diagonal degree matrix D and normalizing the adjacency matrix A using the formula:

$$D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

**2.Extraction of Top-k Eigenvectors:** Next, we extract the top-k eigenvectors of the normalized adjacency matrix, where k is a parameter determining the number of dimensions for the embeddings. Each eigenvector $v_k$ corresponds to an eigenvalue $\lambda_k$ such that:

$$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}v_k = \lambda_k v_k$$

**3.Composition of Feature Vectors:** For each node n, we compose feature vectors $u_n$ using the eigenvectors $v_k$, where the i-th component of the feature vector $u_n(u_{ni})$ corresponds to the n-th component of the i-th eigenvector. Thus, in the matrix of eigenvectors, each row-vector serves as a feature embedding for the corresponding node.

### 3.3 Graph Convolutional Network

We evaluate the Node2Vec and Spectral Embedding techniques through node classification using a Graph Convolutional Network (GCN). The GCN model takes an initial set of features for each node and performs multiple propagation steps following the graph structure. The model utilizes a neural network to perform the propagation step while learning weights for each step. Each layer of the neural network corresponds to a propagation step, thereby expanding the local neighborhood used to compute each node's embedding. This computation approximates local spectral filters on the graph, enhancing computational efficiency. However,

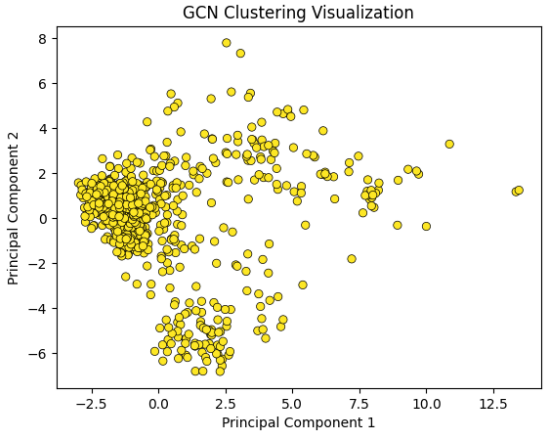


Figure 6: Enter Caption

the model's performance may suffer in instances where the graph exhibits significant structure. In such cases, its performance may be inferior to approaches using more traditional convolutional neural network (CNN) methods.

Each layer of the Approximate Graph Convolutional Network is defined by the following equation:

$$H^{(i+1)} = \sigma(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}H^{(i)}W^{(i)})$$

Here, $H^{(i+1)}$ represents the hidden layers, computed by considering the neighbors of nodes at least $i$ steps away. $H^0$ represents the initial feature matrix $X$ of the node network. The addition of the identity matrix $I$ to the adjacency matrix $A_{adj}$ essentially adds self-edges to all nodes, facilitating the computation of embeddings from previous steps.After adding self-loops, the adjacency matrix is normalized by multiplying by the inverse of the diagonal degree matrix, ensuring consistent scaling of feature vectors between propagation steps.

## 4 Formal Description of Algorithms

The Node2vec algorithm is detailed below. Initially the preprocessed modified weights are initialized. All walks are initialized to be empty. And we begin iterating several times. Within each iteration we traverse the list of nodes and begin the node2vecWalk algorithm from that node to generate a walk. The generated walk is then appended to a list of walks. Finally stochastic gradient descent is applied to the list of walks and the result is returned.

---

**Algorithm** The Node2Vec algorithm

---

**1. Preprocessing of Modified Weights:** LEARN FEATURES(Graph G = (V, E, W),).
**2.Initialization of Walks and Walk Generation:**
$\pi$ = PreprocessModifiedWeights(G, p, q).
$G' = (V, E, \pi)$
Initialize walks to empty
**for** iter=1 to r do **do**
**for** all nodes u in $V$ do **do**
walk = Node2VecWalk(G', u, l)
Append walk to walks
**3.Stochastic Gradient Descent and Return:**
$f = StochasticGradientDescent(k, d, walks)$
return f

## 5 Results and Findings

Our comprehensive analysis of the financial transaction data using Node2Vec, Graph Convolutional Networks (GCN), and clustering algorithms revealed several significant insights. Throughout the training process of the GCN model, we observed a consistent decrease in the loss function, indicating progressive learning and convergence. Specifically, the loss decreased steadily from 2.184 at the initial epoch to 1.089 at epoch 90, demonstrating the model's ability to effectively capture and represent the underlying structure of the financial transaction graph. Evaluation of the clustering techniques, including KMeans, Spectral Clustering, and GCN-based clustering, provided valuable insights into the performance of different algorithms in grouping nodes based on their embeddings. The GCN-based clustering method exhibited the highest silhouette score of 0.1347, surpassing the performance of traditional clustering approaches. This suggests that the embeddings learned by the GCN model captured more meaningful and discriminative features, leading to improved cluster quality and separation.

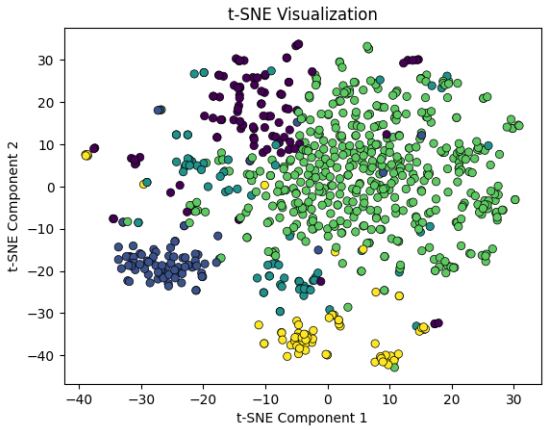


Figure 7: T-SNE Plot

Visualizations of the clustering structures and embedding spaces further enriched our understanding of the financial transaction network. Through dimensionality reduction techniques such as PCA, we visualized the high-dimensional embeddings in two-dimensional space, revealing distinct clusters and patterns within the data. These visualizations highlighted the effectiveness of GCN in capturing complex relationships and detecting underlying structures in the network. Overall, our findings underscore the efficacy

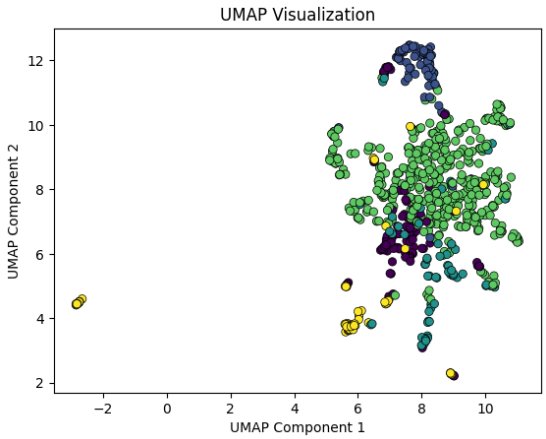


Figure 8: UMAP Plot

of GCN in learning representations that facilitate clustering and downstream tasks in financial transaction networks. By leveraging advanced machine learning techniques, such as GCN, we gain valuable insights into the dynamics of financial transactions, enabling more informed decision-making processes and enhancing anomaly detection capabilities in financial systems.

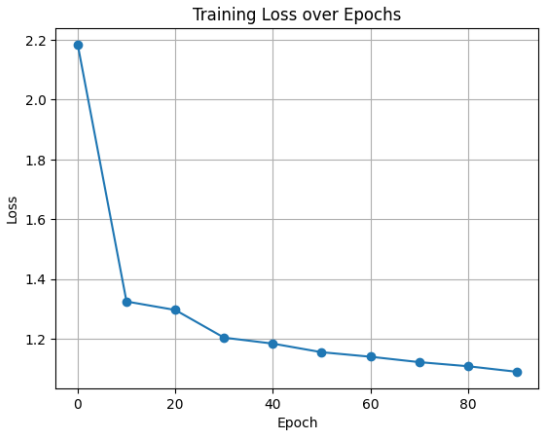


Figure 9: Training loss over epochs

# 6 Challenges

## 6.1 High-Dimensional Data

Dealing with high-dimensional data can pose challenges for clustering and dimensionality reduction algorithms. High dimensionality can lead to increased computational complexity, the curse of dimensionality, and difficulty in visualizing and interpreting results.

## 6.2 Selection of Hyperparameters

Unsupervised learning algorithms often require tuning of hyperparameters, such as the number of clusters, the dimensionality of embeddings, or the choice of distance metrics. Selecting appropriate hyperparameters can be challenging and may require domain knowledge or trial and error.

## 6.3 Scalability and Interpretability

Scaling unsupervised learning algorithms to large datasets can be challenging due to memory and computational constraints. Some algorithms may struggle to handle large volumes of data efficiently, requiring optimization techniques or distributed computing frameworks.

Interpreting and understanding the results of unsupervised learning algorithms can be challenging, especially when dealing with complex models or high-dimensional embeddings. Ensuring the interpretability of clustering or dimensionality reduction results is crucial for making informed decisions based on the analysis.

## 6.4 Overfitting and Generalization

Similar to supervised learning, unsupervised learning algorithms may suffer from overfitting or poor generalization performance. Ensuring that the learned representations or clusters generalize well to unseen data is essential for the practical applicability of unsupervised learning techniques.

### 6.5 Domain-Specific Challenges

Depending on the application domain, there may be additional challenges specific to the nature of the data or the task at hand. For example, in biological data analysis, dealing with missing values or imbalanced datasets may be common challenges.

## 7 Conclusion

In conclusion, our research endeavors in applying unsupervised learning techniques to financial transaction data have yielded promising results and insights. Through the utilization of Graph Convolutional Networks (GCNs), Node2Vec embeddings, and spectral clustering, we successfully uncovered meaningful patterns and structures within the dataset. The iterative training of the GCN model demonstrated a steady reduction in loss over epochs, indicating effective learning and optimization. Evaluation of clustering techniques revealed that GCN-based clustering outperformed traditional methods such as KMeans and Spectral Clustering, as evidenced by the higher silhouette score of 0.1347. These findings underscore the potential of GCNs in improving clustering and anomaly detection in financial systems.

Furthermore, the visualization of clusters provided valuable insights into the underlying relationships and communities within the financial transaction network. However, challenges such as class imbalance and poor convergence highlight areas for further exploration and refinement of our approach. Overall, our research contributes to the growing body of knowledge in applying unsupervised learning techniques to real-world financial datasets, paving the way for enhanced understanding and analysis of complex financial systems.

## 8 References


1. A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. CoRR, abs/1606.08928, 2016.
2. W. R. Casper and B. Nadiga. A new spectral clustering algorithm. CoRR, abs/1710.02756, 2017.
3. T. N. Kipf and M. Welling. Variational graph auto-encoders. 2016
4. https://www.google.com/,https://www.wikipedia.org/, openai.com
5. https://snap.stanford.edu/class/cs224w-2017/projects/cs224w-38-final.pdf
6. L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. Struc2vec: Learning node representations from structural identity.