LEUPHANA
UNIVERSITÄT LÜNEBURG

Field of study: Master of Science Management and Data Science

Analysing Networks
Summer Semester 2024
Leuphana Universität Lüneburg

Semi-supervised Classification With Graph Convolutional Networks

Submitted By:

Bhavana Raju (4000149)

Neda Keshavarz Bahaghighat (4000077)

Shree Shangaavi Nagaraj (4000243)

Vignesh Mallya (4001498)

Examiner:
Prof. Dr. Peter Niemeyer
Leuphana Universität Lüneburg.

16.08.2024

Statuatory declaration

We hereby declare that we have not previously submitted the present work for other examinations. We wrote this work independently. All sources, including sources from the Internet, that we have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables, and images), have been acknowledged by us as such. we understand that violations of these principles will result in proceedings regarding deception or attempted deception.

Lüneburg, 16.08.2024

Bhavana Raju (4000149)

Neda Keshavarz Bahaghighat (4000077)

Shree Shangaavi Nagaraj (4000243)

Vignesh Mallya (4001498)

Contents

1 Introduction

The objective of this project is to explore and compare various approaches for identifying community structures in networks using both embedding-based and graph-based clustering techniques. The focus is on embedding graph data into a d-dimensional Euclidean space, followed by the application of well-known unsupervised clustering methods to detect communities. These communities are anticipated to correspond to significant substructures within the original graph. Through this exploration, deeper insights are expected to be gained into the effectiveness of embedding-based versus graph-based approaches for discovering meaningful communities in different types of networks. This project is intended to provide a deeper understanding of how different clustering methods function and how they can be applied to real-world networks.

1.1 Comparative Description

Two distinct approaches to clustering were utilized: embedding-based clustering and graph-based clustering.

1.1.1 Embedding-Based Clustering

Embedding-based clustering Graph Convolutions Networks (GCNs) allow to transform the nodes of the graph into points in a d-dimensional space. where standard clustering algorithms like k-means can be applied. This process allows for the identification of clusters based on the spatial arrangement of points in this new space.

1.1.2 Graph-Based Clustering

Graph-Based Clustering explore traditional graph-based clustering techniques, such as Label Propagation, Louvain, and Leiden algorithms. These methods operate directly on the graph's structure without requiring a transformation into Euclidean space. Each of these algorithms employs a unique approach to community detection, offering different strengths and potential outcomes depending on the graph's characteristics.

To assess the effectiveness of each method, the Newman modularity metric, a standard measure for evaluating the quality of detected communities, will be employed. This metric will be used to compare the results from both the embedding-based and graph-based approaches, providing a clear indication of which method performs better for different types of networks.

1.2 Datasets

For the analysis, three distinct datasets will be employed: Zachary's Karate Club Graph, a university email network, the Enron email network and the Cora citation network. Each of these datasets presents a unique challenge, and by applying the chosen methods to these graphs, the effectiveness of each technique will be evaluated. It is crucial to treat all these graphs as undirected; therefore, any directed graphs will be converted to undirected ones prior to analysis.

2 Methedology

In the following section, we will explore various clustering techniques, including modularity measures and methods such as GCNs, LPA, and the Louvain and Leiden algorithms. We will assess their strengths, limitations, and suitability for different types of network analysis tasks. This will provide guidance on choosing the most effective method for specific applications.

2.1 Clustering Methods

2.1.1 Embedding-Based Approach: Graph Convolutional Network (GCN)

Graph Convolutional Networks (GCNs) represent a revolutionary approach in the intersection of graph theory and deep learning. Unlike traditional neural networks that operate on regular grid data (like images or text), GCNs are designed to work directly with graph-structured data, making them particularly useful for a wide range of applications, including social network analysis, recommendation systems, and biological network analysis.

At the heart of GCNs is the idea of performing convolution operations on graphs. In simple terms, a convolutional layer in a GCN aggregates information from a node's neighbors to update its representation. This is akin to how CNNs aggregate pixel information from neighboring pixels in image data, but in GCNs, the neighbors are the connected nodes in a graph.

The basic operation of a GCN layer can be described by the following propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

Here, $H^{(l)}$ denotes the matrix of node features at layer l, and $H^{(l+1)}$ is the updated feature matrix at the next layer. $\tilde{A} = A + I$ is the adjacency matrix with added self-loops, ensuring that a node's own features are considered in the convolution. $\tilde{D}$ is the degree matrix corresponding to $\tilde{A}$, and $W^{(l)}$ is a trainable weight matrix specific to the layer. The function σ typically represents a non-linear activation function, such as ReLU (Rectified Linear Unit).

Through multiple layers of such convolutions, a GCN learns to encode the structural information of a graph into low-dimensional embeddings. These embeddings capture both the local structure (connections to immediate neighbors) and the global structure (connections across the graph) of each node, making GCNs powerful tools for node classification, link prediction, and clustering.

Once the nodes are embedded into a Euclidean space $\mathbb{R}^d$, semi-supervised clustering techniques can be applied to group nodes based on their learned representations. These embeddings serve as a compact and informative summary of the graph, where similar nodes (in terms of their roles or positions in the network) are positioned closer to each other in the embedding space.(Kipf and Welling, 2016)

2.1.2 Graph-Based Approaches

2.1.2.1 Label Propagation

The Label Propagation Algorithm (LPA) is a straightforward yet effective method for semi-supervised learning on graphs. The algorithm exploits the graph structure to propagate labels from a few labeled nodes to the rest of the graph. The key assumption in LPA is that connected nodes are more likely to share the same label. This assumption is often valid in social networks, citation networks, and many other real-world graphs where nodes tend to cluster into communities with shared characteristics.

The process begins with initializing each node with a label, where labeled nodes retain their initial labels, and unlabeled nodes may start with random or no labels. During each iteration of the algorithm, every node

updates its label to the most common label among its neighbors. This iterative process continues until the labels stabilize, meaning that no node changes its label anymore.

LPA is particularly valued for its simplicity and efficiency. It operates in linear time with respect to the number of edges, making it scalable to very large graphs. However, one limitation of LPA is that it can be sensitive to the initial label assignment and may converge to different solutions depending on the order of label updates.(Zhu et al., 2003)

2.1.2.2 Louvain Method

The Louvain method is one of the most popular algorithms for community detection in large networks due to its efficiency and ability to uncover hierarchical community structures. The method operates by iteratively maximizing modularity—a measure of the quality of the community structure—through two primary phases.

In the first phase, each node is assigned to its own community. The algorithm then considers moving each node to the community of each of its neighbors, selecting the move that results in the greatest increase in modularity. This process is repeated for all nodes until no further improvement in modularity is possible.

In the second phase, the algorithm creates a new network where nodes represent the communities identified in the first phase, and edges between these nodes are weighted by the sum of edges between nodes in the original communities. The two phases are repeated iteratively on this new network until modularity cannot be further optimized.

One of the strengths of the Louvain method is its ability to uncover hierarchical structures, where communities at one level may be further subdivided into smaller communities. However, the algorithm can sometimes result in disconnected communities or get trapped in local optima, where modularity cannot be improved even though the global maximum has not been reached.(Blondel et al., 2008)

2.1.2.3 Leiden Algorithm

The Leiden algorithm builds on the Louvain method, addressing some of its key limitations. While the Louvain method can produce disconnected communities and may get stuck in suboptimal partitions, the Leiden algorithm includes additional steps to refine the partitioning process and ensure that communities are well-connected.

In the Leiden algorithm, after an initial partitioning similar to the Louvain method, a refinement step is introduced to improve the quality of the partition. This step ensures that all nodes in a community are connected by a path of nodes that belong to the same community, which reduces the risk of disconnected communities. Additionally, the Leiden algorithm reassigns nodes to new communities more flexibly, which helps in finding better partitions that Louvain might miss.

The Leiden algorithm also performs better in terms of computational efficiency, especially on large-scale networks. Its enhanced ability to optimize modularity and produce more meaningful community structures makes it a preferable choice for applications where the quality of community detection is paramount.(Traag et al., 2019)

2.2 Comparison of Clustering Approaches

The clustering methods discussed—Graph Convolutional Networks (GCN), Label Propagation Algorithm (LPA), Louvain method, and Leiden algorithm—each offer unique strengths and are suited to different types of problems.

Graph Convolutional Networks (GCN): GCNs are highly versatile, capable of integrating node features with graph structure to produce rich embeddings that capture both local and global properties of the graph. They are particularly effective in scenarios where labeled data is scarce but graph structure is informative.

However, GCNs require significant computational resources and careful tuning of hyperparameters. They are also more complex to implement and may not scale as efficiently as purely graph-based methods for very large networks.

Label Propagation Algorithm (LPA): LPA is the simplest and most efficient method among the four. It is well-suited for scenarios where speed and scalability are critical, and the graph has a clear community structure. However, its simplicity comes with trade-offs in terms of robustness and stability. LPA may produce different results depending on the order of label updates and can struggle with complex community structures where nodes belong to multiple communities.

Louvain Method: The Louvain method is a workhorse for community detection in large networks. It is fast, scalable, and often finds high-quality community structures, making it a go-to method for many applications. However, it can sometimes yield suboptimal results, especially if the network has many small communities or if the initial partitioning is not ideal.

Leiden Algorithm: The Leiden algorithm is a refinement of the Louvain method, designed to overcome its limitations. It is more reliable in producing well-connected and meaningful communities and generally achieves better modularity scores. The Leiden algorithm is also faster in practice, particularly on large networks, making it an excellent choice when the quality of community detection is crucial.

In summary, the choice of method hinges on the specific demands of the application. For tasks that require deep node embeddings and semi-supervised learning, GCNs, as demonstrated by Kipf and Welling, provide remarkable flexibility and effectiveness. However, when the goal is fast and scalable community detection with rigorous quality assurances, the Leiden algorithm is often the preferred choice. For less complex tasks, algorithms like Label Propagation (LPA) or Louvain can offer sufficient accuracy with high efficiency.

2.3 Modularity

Modularity is a fundamental concept in network science that measures the quality of a division of a network into modules, also known as communities. The underlying idea is to capture how well a network is divided into groups of nodes with dense connections internally and sparse connections externally.

In practical terms, modularity can be thought of as a measure that compares the actual network structure with a randomly generated network that has the same number of nodes and connections but no modular structure. A high modularity value indicates that the network has a strong community structure, meaning that most of the edges fall within communities rather than between them.

Mathematically, modularity Q is calculated using the formula:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

Here, A_{ij} represents the adjacency matrix of the graph, where A_{ij} is 1 if there is an edge between nodes i and j and 0 otherwise. The terms k_i and k_j are the degrees (number of connections) of nodes i and j, respectively. m is the total number of edges in the network. The function $\delta(c_i, c_j)$ equals 1 if nodes i and j are in the same community and 0 otherwise.

The modularity measure essentially sums the difference between the actual edge distribution within communities and what would be expected in a random distribution. If the nodes are more interconnected within their communities than with the rest of the network, modularity will be positive, with values typically ranging between -1 and 1. Values close to 1 indicate strong community structure, whereas values near 0 or negative suggest weak or no community structure.

This metric is widely used not only because it is effective in quantifying the strength of community structures but also because it guides the optimization processes in many community detection algorithms, such as the Louvain and Leiden methods.

2.4 Hyperparameter Justification and Runtime Analysis

2.4.1 Zachary's Karate Club Dataset

- **Technical Description:** This dataset is a small social network of 34 nodes representing members of a karate club, with 78 edges denoting friendships. Given its modest size, the GCN model applied to this graph was kept relatively simple yet effective, ensuring quick training and reliable results. We employed a two-layer GCN model with 16 hidden units. The hidden layer size of 16 was chosen to provide a balance between model complexity and performance. The ReLU activation function was selected for its efficiency and widespread use in graph convolutional networks (GCNs), promoting non-linearity while avoiding vanishing gradients. Dropout, with a default rate of 0.5, was implemented to reduce overfitting.

- **Justification for Hyperparameters:** The learning rate of 0.01 was chosen as a moderate value to ensure stable convergence during training, especially considering the relatively small size of the dataset. The Adam optimizer, known for its robust performance in various settings, was selected to handle the stochastic nature of the gradient descent efficiently. A weight decay of 5e-4 was added to regularize the model, discouraging excessive complexity and promoting generalization. The choice of 200 epochs strikes a balance between allowing sufficient learning while avoiding overfitting or unnecessary computations on this small dataset.

- **Runtime Considerations:** Given the small size of the graph, the training process is extremely fast. The model converges quickly within the 200 epochs, making this setup computationally inexpensive and ideal for a proof-of-concept application or for educational purposes.

2.4.2 Enron Email Dataset

- **Technical Description:** The Enron Email dataset represents a larger and more complex graph, where nodes correspond to email addresses and edges represent email exchanges. The structure of the GCN model remains consistent with that used for the Karate Club dataset, leveraging the same hyperparameters. This decision was made to ensure consistency in model behavior across different datasets, simplifying comparisons. The use of synthetic binary labels allows for straightforward binary classification tasks, despite the complexity of the graph.

- **Justification for Hyperparameters:** Maintaining the same model architecture and hyperparameters across datasets allows us to directly compare performance, particularly how model complexity and data size interact. The same learning rate, dropout rate, and optimizer settings were applied, as they have proven effective in handling graph data. Given the larger size of the Enron dataset, the number of epochs (200) was again deemed appropriate, ensuring the model has enough time to learn the underlying patterns without overfitting.

- **Runtime Considerations:** The larger size of the Enron graph naturally leads to longer training times compared to the Karate Club dataset. However, the use of the Adam optimizer and a learning rate of 0.01 helps manage this, ensuring convergence within a reasonable time frame. The increase in runtime

is a direct result of the graph's complexity and is expected when working with real-world datasets of this size.

2.4.3 University Email Dataset

- **Technical Description:** The University Email dataset is structurally similar to the Enron dataset, involving a variable number of nodes and edges representing email communications. As with the previous datasets, the GCN model's architecture and hyperparameters were kept consistent to allow for direct comparison. This dataset also uses synthetic binary labels, maintaining a focus on binary classification tasks.
- **Justification for Hyperparameters:** Consistency in hyperparameter selection allows for a more straightforward analysis of how graph size and complexity affect model performance. The architecture—16 hidden units with ReLU activation, a dropout rate of 0.5, and a learning rate of 0.01—has been proven to be effective across various graph sizes. The selection of 200 epochs again ensures that the model has sufficient time to learn from the data without unnecessarily increasing computational costs.
- **Runtime Considerations:** Given the similarities in size and complexity between this dataset and the Enron dataset, the runtime considerations are analogous. The model will take longer to train compared to smaller datasets like the Karate Club dataset, but the training time remains manageable thanks to the careful selection of hyperparameters and the use of efficient optimization techniques.

2.4.4 Cora Dataset

- **Technical Description:** The Cora dataset is a benchmark graph dataset commonly used in the study of GCNs, featuring 2,708 nodes (representing scientific publications) and 5,429 edges (representing citations between these publications). Unlike the previous datasets, the Cora dataset requires a more complex GCN model, given its larger size and the multi-class classification task (with 7 distinct classes). Here, a two-layer GCN architecture is employed with 16 hidden units, similar to the other datasets, but with an increased output unit count matching the 7 classes in the dataset.
- **Justification for Hyperparameters:** The decision to retain 16 hidden units was made to manage model complexity while still providing sufficient representational power to capture the nuanced relationships in the data. The ReLU activation and Adam optimizer are standard choices that have consistently delivered robust performance in graph-based neural networks. The learning rate of 0.01 continues to be effective in this larger setting, facilitating stable convergence without requiring excessive fine-tuning. A key addition is the early stopping mechanism, with a patience of 10 epochs, introduced to prevent overfitting and to reduce unnecessary computation once the model stops improving.
- **Runtime Considerations:** Due to the larger size and complexity of the Cora dataset, training times are significantly longer compared to the previous datasets. The early stopping criterion helps mitigate this by halting training once further improvements are minimal, making the process more efficient. Memory usage is also higher, necessitating careful consideration of computational resources when working with this dataset.

The hyperparameters selected for the GCN models and clustering methods reflect a careful balance between complexity, runtime, and the need for generalizable insights across varied datasets. By maintaining consistency across datasets, we can better understand how these models scale and perform, providing a strong foundation for further exploration and refinement.

3 Result

3.1 Graph Descriptions

In this section, you would describe the different datasets used in your analysis. This includes details such as the number of nodes, edges, the density of the graph, and any particular features or attributes that are relevant to the datasets.

Dataset	Description	Nodes	Edges	Density	Average Clustering Coefficient	Is Directed	Connected Components
Zachary's Karate Club	A social network of friendships between 34 members of a karate club.	34	78	0.14	57.06	False	1
Enron Email Network	A communication network representing emails exchanged at Enron Corp.	143	623	0.061	43.39	False	1
University Email Network	An email communication network within a university.	1133	5451	0.0085	22.02	False	1
Cora Dataset	A citation network with nodes representing papers and edges as citations.	2708	5278	0.0014	31.53	False	1

Table 1: Summary of the datasets

3.2 Technical Approach

3.2.1 Label Propagation, Louvain, and Leiden:

The Label Propagation, Louvain, and Leiden algorithms were applied to detect communities within the networks. The approaches are outlined below:

- **Label Propagation:**
- Initialization: Each node in the graph is initially assigned a unique label.
- Propagation: Labels are propagated through the network iteratively, where each node adopts the most frequent label among its neighbors. The process continues until the labels stabilize.
- Stopping Criteria: The algorithm stops when no further changes occur in the labels of the nodes, indicating convergence.
- **Louvain Method:**
- Phase 1: Begin with each node as its own community and iteratively move nodes to neighboring communities to maximize modularity. This is done until no further improvement is possible.
- Phase 2: Communities are aggregated into supernodes, and the process is repeated on this smaller graph. The algorithm continues until no further modularity gain can be achieved.

- Modularity Optimization:] Louvain focuses on maximizing modularity, a measure of the quality of the community structure, where higher modularity indicates better-defined communities.
- **Leiden Algorithm:**
- Phase 1 Similar to Louvain but includes additional steps to refine the community structure.
- Phase 2: Nodes are re-assigned to communities to further optimize modularity, ensuring that all communities are internally well-connected.
- Efficiency: Leiden is an improvement over Louvain, as it addresses the issue of disconnected communities and provides better scalability.

3.2.2 GCN:

The Graph Convolutional Network (GCN) was implemented to learn node embeddings that incorporate both node features and the graph structure.

- **Data Preparation:**
- Graph Data: The adjacency matrix of the graph is prepared and normalized to ensure proper propagation of node features.
- Node Features: Initial node features are set, often as identity matrices if no other features are available. Model Definition:
- GCN Layers: The model consists of multiple GCN layers, each performing the operation typically ReLU.
- Output: The final output layer produces node embeddings which are used for downstream tasks like node classification or clustering.
- **Training:**
- Loss Function: A suitable loss function (e.g., cross-entropy) is used to train the model, with the goal of minimizing the difference between predicted and actual labels.
- Optimizer: Common optimizers like Adam are used to update the model weights during training.
- **Evaluation:**
- Embeddings: The learned embeddings are used for clustering, typically using KMeans to identify communities within the graph.
- Metrics: The quality of the clustering is evaluated using metrics such as modularity, coverage, and performance.

3.3 Results

3.3.1 Zachary's Karate Dataset:

- **For the Karate dataset, we report the following metrics:**

  Number of Communities Detected: Typically 2 to 4 communities are identified. Average Community Size: This can vary depending on the algorithm used. Modularity: A modularity score of around 0.4 is

common. Coverage and Performance: Metrics indicating how well the communities cover the graph and the performance of the algorithms.

	Algorithm	Number of Communi	Average Community	Modularity	Coverage	Performance	Iterations	Resolution
1	Label Propagation	3	11.333	0.309	0.769	0.668	1	-
2	Louvain	4	8.5	0.445	0.731	0.804	-	1.0
3	Leiden	4	8.5	0.445	0.731	0.804	-	None
4	GCN-based Clustering	4	8.5	0.437	0.718	0.818	-	-

Figure 1: Karate dataset clustering result

3.3.2 Enron Email Dataset:

- **For the Enron dataset:**

  Number of Communities: The algorithms detect between 7 to 10 communities. Modularity: Scores around 0.5 to 0.57, with Louvain and Leiden generally performing best. Comparison of Algorithms: GCN-based methods may struggle due to the lack of node features, while graph-based methods perform well.

	Algorithm	Number of Communi	Average Community	Modularity	Coverage	Performance	Iterations	Resolution
1	Label Propagation	10	14.3	0.505	0.73	0.84	1	-
2	Louvain	7	20.429	0.568	0.734	0.886	-	1.0
3	Leiden	7	20.429	0.568	0.734	0.887	-	None
4	GCN-based Clustering	7	20.429	0.516	0.742	0.832	-	-

Figure 2: Enron Email Dataset clustering result

3.3.3 University Email Dataset:

- **For the University Email Network:** Number of Communities: Between 9 to 13 communities are detected, with Label Propagation often detecting more (e.g., 44 communities). Modularity: Scores above 0.5 for Louvain, Leiden, and GCN with contrastive loss. Coverage and Performance: Variations in coverage and performance are noted across different methods.

	Algorithm	Number of Communi	Average Community	Modularity	Coverage	Performance	Iterations	Resolution
1	Label Propagation	44	25.75	0.463	0.744	0.76	1	-
2	Louvain	11	103.0	0.573	0.691	0.895	-	1.0
3	Leiden	11	103.0	0.58	0.7	0.896	-	None
4	GCN-based Clustering	13	87.154	0.315	0.522	0.861	-	-

Figure 3: Enron Email Dataset clustering result

3.4 Hyperparameter Selection for GCN

Hyperparameters were carefully selected to optimize the performance of the GCN model. The key parameters included:

1. **Zachary's Karate Dataset:**

   Hyperparameters: Hidden Channels: 24, Output Channels: 16, Attention Heads: 7 Learning Rate: 0.0001 Dropout: 0.74 Weight Decay: 0.0009

2. **Enron Email Dataset:** Similar to Karate but with slight variations in optimal parameters.

3. **University Email Dataset** Details specific to the University dataset with additional parameters fine-tuning.

3.5 Runtime Analysis

1. **Zachary's Karate Dataset:**

   Comparison of Runtime: Label Propagation is the fastest. GCN-based methods are significantly slower due to the complexity of the neural network operations.

2. **Enron Email Dataset:**

   Scalability: The Enron dataset being larger affects the runtime, especially for GCN-based methods, which may be orders of magnitude slower.

3. **University Email Dataset**

   Efficiency: Runtime increases with the size and complexity of the graph. The GCN's slower runtime is consistent across datasets.

3.6 Cora Data Set

3.6.1 Clustering Results

For the Cora dataset: (Grattarola and Alippi, 2020)

- Number of Communities Detected: Varies across methods.
- Modularity Scores: Evaluates how well the communities are separated.

	Algorithm	Number of Communi	Average Community	Modularity	Coverage	Performance	Iterations	Resolution
1	Label Propagation	7	387.429	0.341	0.615	0.753	1	-
2	Louvain	6	451.333	0.402	0.67	0.789	-	1.0
3	Leiden	6	451.333	0.405	0.675	0.792	-	None
4	GCN-based Clustering	8	338.5	0.359	0.629	0.769	-	-

Figure 4: Cora Dataset clustering result

3.6.2 Visualizations

- **Graph Visualizations:** Includes the visual representations of clusters for different methods, showing how well each method identifies community structures in the Cora dataset.

4 Discussion

4.1 Comparison of Clustering Approaches (Practical Results):

In this section, we compare the practical outcomes of the different clustering approaches applied to the datasets. The algorithms under consideration include Label Propagation, Louvain, Leiden, and the GCN-based clustering. The comparison will focus on key metrics such as the number of communities detected, average community size, modularity scores, coverage, and performance.

- **Label Propagation:**

- Strengths: Label Propagation is computationally efficient and can handle large datasets with ease. It quickly converges by iteratively propagating labels, resulting in good performance in terms of modularity and coverage.

- Weaknesses: However, it often struggles with smaller, more interconnected graphs like the Karate dataset, where its modularity scores were lower compared to other methods. Additionally, the lack of a global optimization objective can lead to suboptimal community structures.

- **Louvain:**

- Strengths: Louvain is well-known for its ability to maximize modularity, often outperforming other algorithms in terms of this metric. It is particularly effective for medium to large-sized graphs, as seen in the Enron and University Email Network datasets.

- Weaknesses: The algorithm can sometimes produce disconnected communities, especially in graphs with a complex structure. Its performance also tends to degrade slightly on very large graphs due to the resolution limit, where it may merge smaller communities into larger ones.

- **Leiden:**

- Strengths: Leiden improves upon Louvain by refining the community structure, ensuring that all communities are well connected. It consistently achieved high modularity scores across all datasets, making it the most robust choice for community detection.

- Weaknesses: Despite its advantages, Leiden requires more computational resources than Louvain and Label Propagation, particularly when applied to very large graphs. This can be a limiting factor in resource-constrained environments.

- **GCN-based Clustering:**

- Strengths: GCN-based clustering is unique in that it leverages node features and graph structure simultaneously to generate embeddings, which are then clustered. This approach allows for capturing more nuanced community structures, particularly in datasets like Cora, where node features play a significant role.

- Weaknesses: However, GCNs are computationally expensive and require careful hyperparameter tuning. The results showed that while GCN-based clustering can achieve competitive modularity scores, it often comes at the cost of increased runtime and complexity.

4.2 Graph-Specific Insights

Each graph dataset presented unique challenges and insights, which impacted the performance of the clustering algorithms:

- **Zachary's Karate Club:**
- Observation: The small size and high connectivity of the Karate Club graph favored algorithms that could effectively distinguish between tightly-knit communities. Louvain and Leiden performed better than Label Propagation, with Leiden providing the best balance between modularity and community size.
- Insight: For small social networks, methods like Louvain and Leiden that focus on modularity maximization are more effective, as they can better capture the subtle community structures.
- **Enron Email Network:**
- Observation: The Enron Email Network, being larger and less dense, highlighted the advantages of GCN-based clustering, which could leverage the additional node features. However, traditional methods like Louvain still performed strongly in terms of modularity.
- Insight: In graphs where node features are significant, GCN-based methods can outperform traditional clustering algorithms. However, for purely structural clustering, Louvain and Leiden are still highly effective.
- **University Email Network:**
- Observation: The large size of this network challenged all algorithms, but Leiden outperformed Louvain slightly in modularity, showing its ability to refine communities more effectively. Insight: For large-scale networks, Leiden's refinement capabilities give it an edge over Louvain, especially when the network has complex, overlapping communities.
- **Cora Dataset:**
- Observation: The Cora dataset, with its rich node features and citation network structure, was best captured by the GCN-based approach, which produced more meaningful embeddings that translated into high-quality clusters.
- Insight: GCN-based methods are particularly suited for citation networks like Cora, where node features are crucial. This highlights the importance of considering node features in addition to graph structure in certain types of networks.

4.3 Limitations:

While the clustering methods applied provided valuable insights, several limitations were noted:

- **Computational Complexity:**
- GCN-based Methods: These are computationally expensive, requiring significant resources for both training and inference. This limits their scalability to very large graphs.
- Leiden Algorithm: Although more effective than Louvain in terms of modularity, Leiden's increased computational demand can be a bottleneck, particularly in large, complex networks.

- **Scalability:**
- Label Propagation: While scalable, Label Propagation's lack of a global optimization objective can lead to suboptimal clusters in certain graphs, especially smaller, more interconnected ones.
- Louvain: The resolution limit of Louvain can cause it to miss smaller communities in larger graphs, impacting its effectiveness in networks with a diverse range of community sizes.
- **Generalization:**
- Dataset-Specific Performance: The effectiveness of each algorithm varies significantly with the dataset. For instance, GCN-based methods excel in feature-rich datasets like Cora but do not always outperform traditional methods in purely structural graphs.
- Hyperparameter Sensitivity: Particularly for GCNs, the results are highly sensitive to the choice of hyperparameters, making them less robust across different types of graphs.

5 Conclusion

In this report, we explored the application of Graph Convolutional Networks (GCN) and traditional graph clustering algorithms (Label Propagation, Louvain, and Leiden) on various datasets, including Zachary's Karate Club, the Enron Email Network, the University Email Network, and the Cora dataset.

- **Graph Descriptions** provided insights into the structure of each dataset, highlighting the characteristics such as node count, edge count, and average clustering coefficient. These descriptions were crucial in understanding the context in which the algorithms were applied.
- **Technical Approach** discussed the implementation details of each algorithm. The GCN was used to learn node embeddings that capture both node features and graph topology, while Label Propagation, Louvain, and Leiden were used to detect communities directly from the graph structure. The Leiden algorithm, an improved version of Louvain, demonstrated better scalability and more refined community detection, making it particularly effective in larger and more complex datasets like the Cora network.
- **Results** were presented in detailed tables, comparing the performance of each algorithm across the different datasets. Key metrics included the number of communities detected, average community size, modularity, coverage, and performance. The results showed that while GCN-based clustering often produced competitive modularity scores, traditional methods like Louvain and Leiden performed better in terms of modularity and efficiency, particularly on larger networks.
- **Hyperparameter Selection** was crucial for optimizing the GCN performance. The choice of hyperparameters such as hidden channels, learning rate, and dropout significantly impacted the quality of the embeddings and the subsequent clustering results.
- **Runtime Analysis** highlighted the computational efficiency of each algorithm, with GCN-based models requiring more time due to the complexity of the neural network training process. Traditional clustering methods, particularly Label Propagation, were much faster and still provided strong results, especially in simpler graphs like Zachary's Karate Club.

  In summary, while GCNs offer a powerful way to incorporate node features and graph structure into a unified model, traditional graph clustering algorithms like Louvain and Leiden remain highly

effective, especially in terms of modularity and runtime. The choice between these methods depends on the specific requirements of the task, such as the need for scalability or the ability to handle rich node features. For large-scale graphs or when node features are less critical, Louvain and Leiden are preferable. In contrast, for tasks requiring a deep understanding of node interactions and features, GCNs offer a more comprehensive approach.

List of Figures

List of Tables

List of References

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment, 2008*(10), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

Grattarola, D., & Alippi, C. (2020). Spektral: Graph neural networks with keras and tensorflow [Accessed: 2024-08-13]. https://github.com/danielegrattarola/spektral.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*. https://doi.org/10.48550/arXiv.1609.02907

Traag, V. A., Waltman, L., & Van Eck, N. J. (2019). From louvain to leiden: Guaranteeing well-connected communities. *Scientific reports*, *9*(1), 1–12.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions, 912–919.

6 Appendix

6.1 Description of LLM Usage

- How are Graph Convolutional Networks (GCNs) integrated into the architecture of Large Language Models (LLMs) for enhancing their understanding of relationships between tokens or concepts?
- In the context of semi-supervised learning, how does the performance of GCNs differ from that of the Label Propagation method when applied to networks with noisy or incomplete data?

6.2 Individual Contributions

- Bhavana Raju (4000149)

  Implemented code on the Enron Data Set

- Neda Keshavarz Bahaghighat (4000077)

  Implemented code on the University Mail Data Set

- Shree Shangaavi Nagaraj (4000243)

  Implemented code on the Cora Citation Network Data Set

- Vignesh Mallya (4001498)

  Implemented code on Zachary's Karate Club Data Set

Report documentation was done collaboratively by all members, with each contributing to and addressing the relevant topics.