1 Question 1

Let G be a graph that consists of M connected components. Each connected component is a complete graph K_2 (i.e., two nodes connected by an edge). Suppose you use the DeepWalk algorithm to embed the $2M$ nodes in some vector space. How do you expect the cosine similarities of the embeddings of the nodes within connected components and the embeddings of the nodes in different connected components to compare?

The DeepWalk algorithm allows to learn a latent representation of a graph by generating random walks of fixed length for each node. Using this walks, the algorithm learns the embedding using the principle behind SkipGram: given a node's embedding, the model tries to maximize the likelihood of observing its neighbours [3].
Formally, considering a specific node v_i and a random walk, this comes down to the following optimization problem:

$$\min_{\phi} -\log \prod_{\substack{j=i-w \\ j\neq i}}^{i+w} \mathbb{P}\left(v_j|\phi(v_i)\right)$$

This criterion is then minimized over each node v_i in each generated random walk.

The rationale behind (relatively short) random walks is to learn node representations by learning the local structure of the graph around each node. Therefore, looking back at the considered structure with M connected components, we intuitively expect the algorithm to learn a representation in which nodes from the same component will be close with respect to cosine similarity. Moreover, each connected component being composed by exactly two nodes (v_i, v_j), the optimization problem can be written as:

$$\min_{\phi} -w\left(\log \mathbb{P}\left(v_j|\phi(v_i)\right) + \log \mathbb{P}\left(v_i|\phi(v_i)\right)\right)$$

This problem is thus equivalent to maximizing the probability of each node in the connected component given the latent representation of one of these two. The roles of v_i and v_j being symmetric, this will lead to having similar latent representation for v_i and v_j as the respective conditional distributions are optimized over the same problem. Such a paradigm will consequently lead to cosine similarities close to 1 for elements inside the same component.

Finally, in the optimisation problems considered, the only goal is to maximize likelihood of neighbors' apparition given the latent representation of a node in the random walk ; this also means that we did not explicitly try to minimize the likelihood of non-neighbors' apparition. As a consequence, there are no particular reason why cosine similarity between nodes of disconnected components should be close to 1, but we cannot state either that this cosine similarity will be low. We normally expect the components to be distinct given the usual high-dimensional setting, but sometimes this is not the case, especially when the components are showing *structural equivalence* like here, as detailed in [1].

2 Question 2

Provide the time complexity of both the DeepWalk and spectral embedding techniques. For full marks you can either provide references to academic papers, in which the required complexities are stated, and copy down the complexities without derivation or provide us with your own derivation of the complexities.

2.1 DeepWalk

We note γ the number of walks per vertex, $|V|$ the number of vertices, w the window size, d the embedding size and t the walk length. The *DeepWalk* algorithm starts by iterating over two loops, one of size γ and one of size $|V|$. Inside those nested loops, we have two operations:

- *Random Walk*: requires t steps
- *SkipGram*: iterates over the t nodes in a sequence, and for each node tries to maximize the probability of its neighbors in the walk in a window of size $2 \times w$ (consider the w preceding and w succeeding neighbors). According to [2], when using a Hierachical Softmax, the time complexity of the SkipGram algorithm for a considered node is $2 \times w \times (d + d \log |V|)$.

Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

Figure 1: DeepWalk and SkipGram [3]

Therefore, we have the follwoing overall time complexity:

$$\mathcal{O}\left(\underbrace{\gamma \times |V| \times t}_{\textit{Random Walk}} + \underbrace{\gamma \times |V| \times t \times 2 \times w \times (d + d \times \log |V|)}_{\textit{SkipGram}}\right) = \mathcal{O}\left(\gamma \times |V| \times t \times (1 + 2 \times w \times (d + d \times \log |V|))\right).$$

2.2 Spectral Embedding

To perform Spectral Embedding, we need to proceed to the eigendecomposition of the Laplacian matrix of the considered graph.
In the case of a dense matrix (i.e., a graph well-connected), the time complexity of eigendecomposition is $\mathcal{O}(|V|^3)$.
However, graphs often show sparsity in their structure and some algorithms have shown a reduced complexity in those situations.

3 Question 3

Discuss what would be the effect on the architecture if no self-loops were added to the graph (and the graph contained no self-loops to start with), i.e., we would work with $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. In particular, focus on how the hidden states would evolve within a single layer of the GNN and in a two-layer GNN.

In graph neural networks, the propagation of information through the graph is modeled using a propagation rule that involves the adjacency matrix A and node feature matrices. In a GNN without self-loops, at each layer the aggregation of neighboring node information is performed, and this aggregated information replaces the original node features. However, adding self-loops allows each node to retain some of its original information. The self-loop effectively contributes to the aggregation process, ensuring that the node's own features are considered alongside the information from its neighbors.

For example, in a single layer GNN, the update after the message-passing layer would be computed by multiplying the feature matrix X by a matrix $\hat{A}$ with null diagonal entries: for a specific node, the features obtained as output of this layer will reflect information from the node's neighbors while losing the initial information carried by the node.

In the case of a two-layer GNN, a node could retrieve some of its original feature information as this was passed to its neigbors through the first message-passing layer. However, this information will have been tampered and mixed with all the information carried by all of the other neighbors of the node that received information during the first propagation.

One could then make the analogy of self-loops as residuals: for example residuals are added after Attention mechanisms in a Transformer, allowing the Attention to focus on relevant information without having to preserve the initial features. In our context, this could allow the GNN to learn relevant information about the local structure while preserving initial information about the nodes.
Finally, for really large graphs, isolated nodes might occur: adding self-loops ensures that the normalized adjacency matrix $\hat{A}$ is not singular.

4 Question 4

Consider the GNN architecture we have been working with so far. In particular, assume the following weight matrices:

$$\mathbf{W}^0 = \begin{bmatrix} 0.5 & -0.2 \end{bmatrix}, \quad \mathbf{W}^1 = \begin{bmatrix} 0.3 & -0.4 & 0.8 & 0.5 \\ -1.1 & 0.6 & -0.1 & 0.7 \end{bmatrix}$$

Assume that there are no biases and that f is the ReLU function. Please compute (showing your detailed calculations) the two matrices $\mathbf{Z}^0$ and $\mathbf{Z}^1$ for the star graph S_4, which contains a central vertex of degree 3 and three vertices of degree 1, and the cycle graph C_4, in which the vertices can be arranged in a cyclic sequence such that vertices are connected by edges if and only if they are consecutive in the sequence. Let $\mathbf{X}$ be the 4×1 all-ones matrix for both these graphs. What structure do you observe in the representations $\mathbf{Z}^1$ you calculated?

4.1 Star graph S_4

We have the following:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \tilde{A} = A + I = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{D} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \quad \tilde{D}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2\sqrt{2}} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

$$\hat{A}X = \begin{bmatrix} \frac{3\sqrt{2}+1}{4} \\ \frac{\sqrt{2}+2}{4} \\ \frac{\sqrt{2}+2}{4} \\ \frac{\sqrt{2}+2}{4} \end{bmatrix}$$

$$\hat{A}XW^0 = \begin{bmatrix} \frac{3\sqrt{2}+1}{8} & -\frac{3\sqrt{2}+1}{20} \\ \frac{\sqrt{2}+2}{8} & -\frac{\sqrt{2}+2}{20} \\ \frac{\sqrt{2}+2}{8} & -\frac{\sqrt{2}+2}{20} \\ \frac{\sqrt{2}+2}{8} & -\frac{\sqrt{2}+2}{20} \end{bmatrix}$$

$$Z^0 = ReLU(\hat{A}XW^0) = \begin{bmatrix} \frac{3\sqrt{2}+1}{8} & 0 \\ \frac{\sqrt{2}+2}{8} & 0 \\ \frac{\sqrt{2}+2}{8} & 0 \\ \frac{\sqrt{2}+2}{8} & 0 \end{bmatrix}$$

$$Z^0W^1 = \begin{bmatrix} \frac{9\sqrt{2}+3}{80} & -\frac{3\sqrt{2}+1}{20} & \frac{3\sqrt{2}+1}{10} & \frac{27\sqrt{2}+9}{80} \\ \frac{3\sqrt{2}+6}{80} & -\frac{2\sqrt{2}+2}{20} & \frac{\sqrt{2}+2}{10} & \frac{9\sqrt{2}+18}{80} \\ \frac{3\sqrt{2}+6}{80} & -\frac{2\sqrt{2}+2}{20} & \frac{\sqrt{2}+2}{10} & \frac{9\sqrt{2}+18}{80} \\ \frac{3\sqrt{2}+6}{80} & -\frac{2\sqrt{2}+2}{20} & \frac{\sqrt{2}+2}{10} & \frac{9\sqrt{2}+18}{80} \end{bmatrix}$$

$$\hat{A}Z^0W^1 = \begin{bmatrix} \frac{27\sqrt{2}+21}{320} & -\frac{9\sqrt{2}+7}{80} & \frac{9\sqrt{2}+7}{40} & \frac{81\sqrt{2}+63}{320} \\ \frac{9\sqrt{2}+30}{320} & -\frac{3\sqrt{2}+10}{80} & \frac{3\sqrt{2}+10}{40} & \frac{27\sqrt{2}+90}{320} \\ \frac{9\sqrt{2}+30}{320} & -\frac{3\sqrt{2}+10}{80} & \frac{3\sqrt{2}+10}{40} & \frac{27\sqrt{2}+90}{320} \\ \frac{9\sqrt{2}+30}{320} & -\frac{3\sqrt{2}+10}{80} & \frac{3\sqrt{2}+10}{40} & \frac{27\sqrt{2}+90}{320} \end{bmatrix}$$

$$Z^1 = ReLU(\hat{A}Z^0W^1) = \begin{bmatrix} \frac{27\sqrt{2}+21}{320} & 0 & \frac{9\sqrt{2}+7}{40} & \frac{81\sqrt{2}+63}{320} \\ \frac{9\sqrt{2}+30}{320} & 0 & \frac{3\sqrt{2}+10}{40} & \frac{27\sqrt{2}+90}{320} \\ \frac{9\sqrt{2}+30}{320} & 0 & \frac{3\sqrt{2}+10}{40} & \frac{27\sqrt{2}+90}{320} \\ \frac{9\sqrt{2}+30}{320} & 0 & \frac{3\sqrt{2}+10}{40} & \frac{27\sqrt{2}+90}{320} \end{bmatrix}$$

We witness here both in $\mathbf{Z}^0$ and in $\mathbf{Z}^1$ that the leaves are given the same representations (identical rows), while the root has a different representation. This makes sense as the considered graph is a star, and that each leave presents a similar structure.
We also notice the fact that one component (second column) is uniformly null due to the activation function.

4.2 Cycle graph C_4

We have the following:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \quad \tilde{A} = A + I = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \quad \tilde{D} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \quad \tilde{D}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{3}} \end{bmatrix}$$

$$\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} = \frac{1}{3}\tilde{A}$$

$$\hat{A}X = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$Z^0 = ReLU(\hat{A}XW^0) = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \end{bmatrix}$$

$$Z^0W^1 = \begin{bmatrix} \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \end{bmatrix}$$

$$\hat{A}Z^0W^1 = \begin{bmatrix} \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & -\frac{1}{5} & \frac{2}{5} & \frac{9}{20} \end{bmatrix}$$

$$Z^1 = ReLU(\hat{A}Z^0W^1) = \begin{bmatrix} \frac{3}{20} & 0 & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & 0 & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & 0 & \frac{2}{5} & \frac{9}{20} \\ \frac{3}{20} & 0 & \frac{2}{5} & \frac{9}{20} \end{bmatrix}$$

We witness here both in $\mathbf{Z}^0$ and in $\mathbf{Z}^1$ that the nodes are given the same representations (identical rows). This makes sense as the considered graph is a cycle, and that each node presents a similar structure.
We also notice the fact that one component (second column) is uniformly null due to the activation function.

5 Numerical results

5.1 Task 4

The following representation has been obtained by generating 10 walks of length 20 for each node in the graph and using SkipGram to learn a proper embedding in dimension $d = 128$.
We then represented the learned embedding over the 100 most frequent nodes in $\mathbb{R}^2$ using t-SNE.

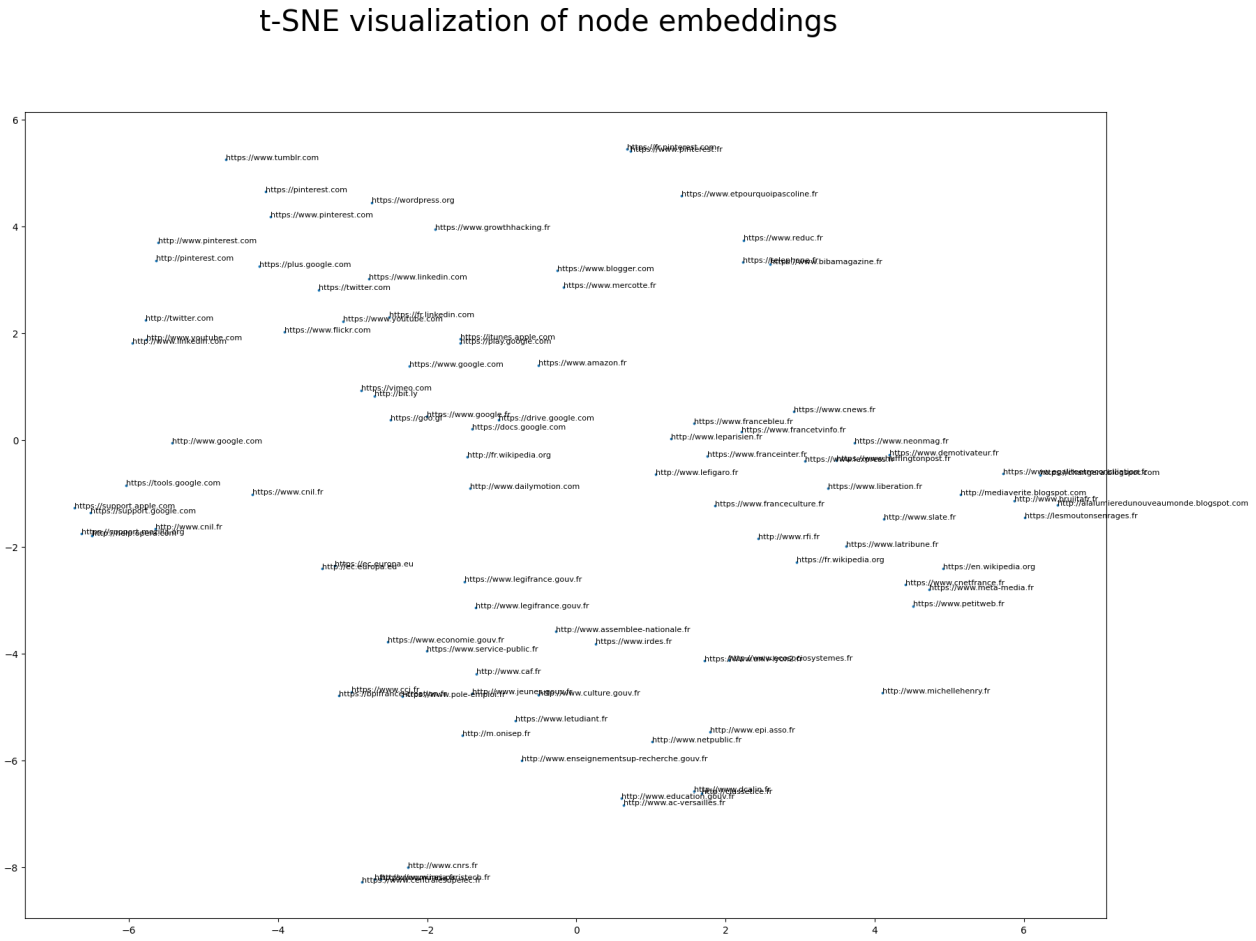

Figure 2: Representation of the embedding for the most frequent occurrences

5.2 Task 5

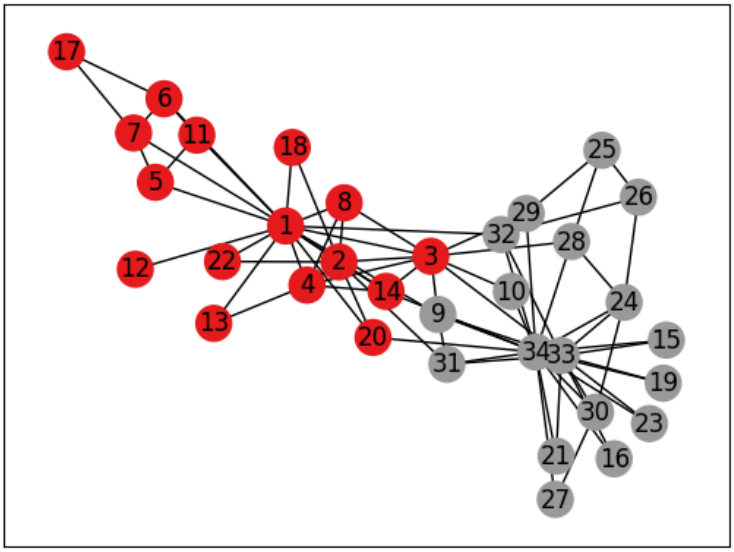

Figure 3: Representation of the Karate network

5.3 Task 8

Table 1: Comparison of the accuracies obtained on the Karate dataset

	Accuracy	
Model	DeepWalk	Spectral Embedding
LogisticRegression Classifier	100.0	85.71

5.4 Task 13

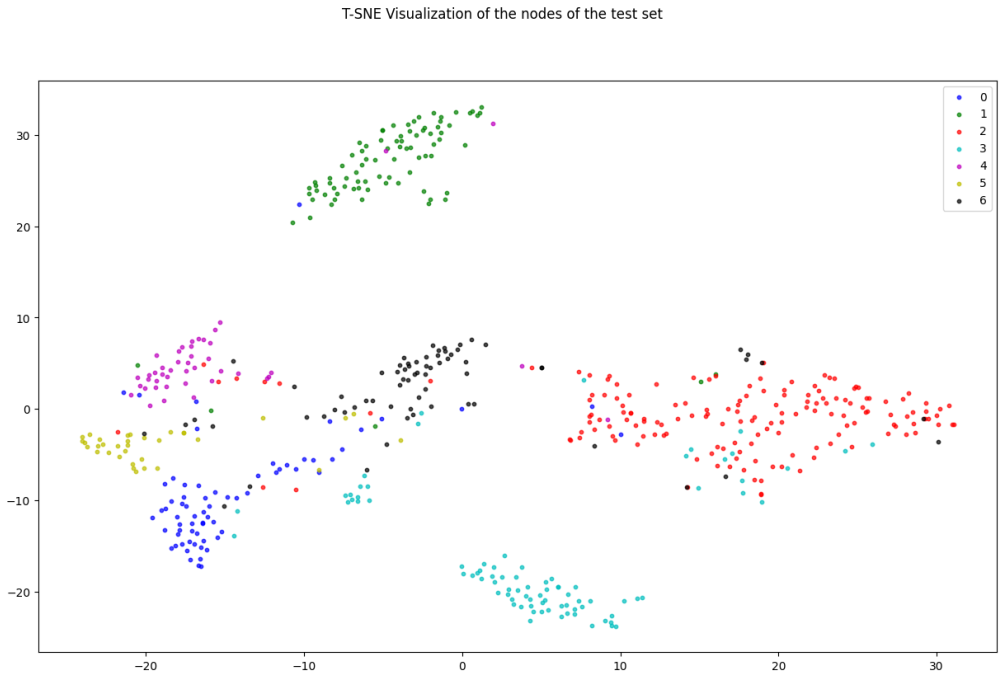

Figure 4: Projection in $\mathbb{R}^2$ of nodes' hidden representation

References

[1] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[3] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.