NAIM Mouna
mouna.naim@ens-paris-saclay.fr

Lab session 2
ALTEGRAD 2024

11/24/24

Task 3: Node embedding with DeepWalk

Here is the the embedding I obtained using DeepWalk

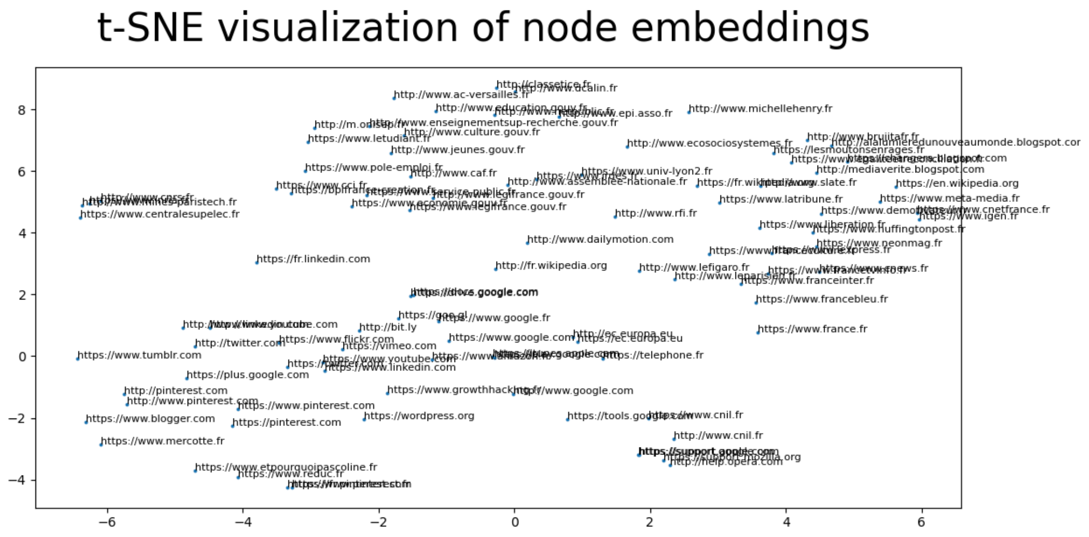

Figure 1: t-SNE visualization of node embeddingd on the French Web

We can observe that the embedding is not bad. For example:

```
http://pinterest.com
http://www.pinterest.com
https://www.pinterest.com
```

are close to each other.

1 Question 1

The DeepWalk architecture can be seamlessly extended to directed graphs without requiring any modifications. To generate a random walk, we simply follow a valid path from a node to its neighbors, ensuring that the direction of edges is respected in directed graphs. For weighted graphs, the process is similarly straightforward. Instead of selecting neighbors randomly, the choice is made proportionally to the edge weights, allowing the random walk to reflect the weighted structure of the graph.

2 Question 2

X_1 and X_2 are related by:

$$X_2 = X_1 \cdot R$$

where R is a reflection matrix given by:

$$R = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

This means X_2 is a reflection of X_1 along the first dimension. The two embedding matrices X_1 and X_2 represent equivalent embeddings of the graph's nodes, differing only by a geometric transformation (reflection). These transformations don'ot affect the performance of downstream tasks, such as classification, because the relative distances and similarities between node embeddings remain consistent.

3 Question 3

In a GCN, the adjacency matrix acts as a tool to aggregate information from a node's neighbors. By multiplying the adjacency matrix with the feature matrix, each node's features are updated to reflect both its own attributes and those of its neighbors.
With each additional layer in the GCN, the receptive field expands, incorporating information from nodes that are further away. Specifically, the k-th layer captures features from all nodes within k-hops of the original node. For the current GCN with 2 layers, the receptive field includes nodes up to 2-hops away. More generally, a GCN with n layers will have a receptive field that spans n-hops in the graph.

Node classification on Karate

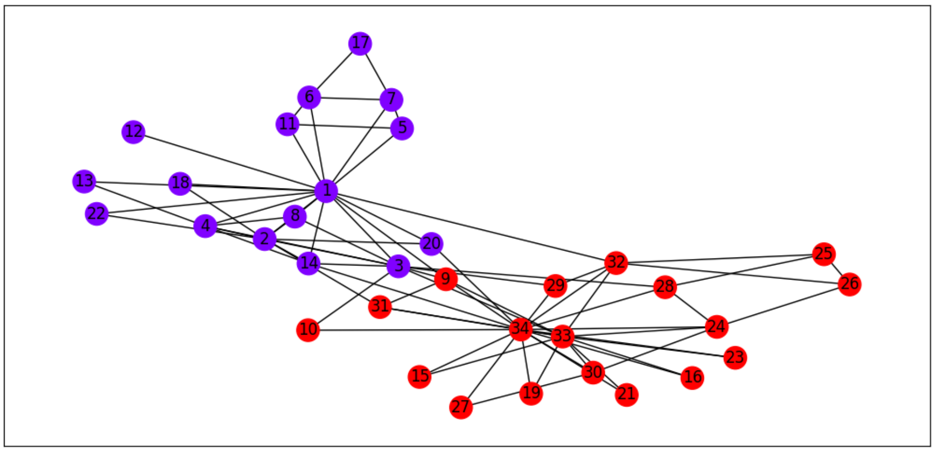

Figure 2: Classification of the 2 groups

Two embedding methods are considered: DeepWalk and Laplacian embedding, both of which do not involve any learning process. In addition, we have two embedding approaches for the Graph Neural Network (GNN): one where the features are distinct (similar to $X = I$) and another where all features are identical (similar to $X = 1$). The table below presents the results obtained using these methods:

Metric	DeepWalk	Laplacian	GNN (X = I)	GNN (X = 1)
Loss	Not calculated	Not calculated	0.0001	84.21%
Accuracy	100%	85.71%	100%	28.57%

Table 1: Comparison of metrics across different embedding and GNN configurations

4 Question 4

Computation of A, $\hat{A}$, $Z^{(0)}$, and $Z^{(1)}$ for K_4

1. Adjacency Matrix (A) The adjacency matrix for the complete graph K_4 is:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

2. Normalized Adjacency Matrix ($\hat{A}$)

To compute $\hat{A}$, we first add self-loops $(A+I)$ and then normalize using $\hat{A} = D^{-1/2}(A+I)D^{-1/2}$, where D is the degree matrix:

$$D = \text{diag}(4,4,4,4), \quad D^{-1/2} = \text{diag}(0.5,0.5,0.5,0.5)$$

$$A+I = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

3. First Layer ($Z^{(0)}$)

Let the feature matrix X and weight matrix $W^{(0)}$ be:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad W^{(0)} = \begin{bmatrix} -0.8 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

Compute $Z^{(0)} = f(\hat{A}XW^{(0)})$:

$$XW^{(0)} = \begin{bmatrix} -0.3 & 0 \\ -0.3 & 0 \\ -0.3 & 0 \\ -0.3 & 0 \end{bmatrix}, \quad \hat{A}(XW^{(0)}) = \begin{bmatrix} -0.3 & 0 \\ -0.3 & 0 \\ -0.3 & 0 \\ -0.3 & 0 \end{bmatrix}$$

$$Z^{(0)} = f(\hat{A}(XW^{(0)})) = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}$$

4. Second Layer ($Z^{(1)}$)

Let the weight matrix $W^{(1)}$ be:

$$W^{(1)} = \begin{bmatrix} 0.1 & 0.3 & -0.05 \\ -0.4 & 0.6 & 0.5 \end{bmatrix}$$

Compute $Z^{(1)} = f(\hat{A}Z^{(0)}W^{(1)})$:

$$Z^{(0)}W^{(1)} = \begin{bmatrix} -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \end{bmatrix}, \quad \hat{A}(Z^{(0)}W^{(1)}) = \begin{bmatrix} -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \end{bmatrix}$$

$$Z^{(1)} = f(\hat{A}(Z^{(0)}W^{(1)})) = \begin{bmatrix} 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \end{bmatrix}$$

Computation of A, $\hat{A}$, $Z^{(0)}$, and $Z^{(1)}$ for S_4

1. Adjacency Matrix (A)

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

2. Normalized Adjacency Matrix ($\hat{A}$)

$$\hat{A} = \begin{bmatrix} 0.25 & 0.35 & 0.35 & 0.35 \\ 0.35 & 0.5 & 0 & 0 \\ 0.35 & 0 & 0.5 & 0 \\ 0.35 & 0 & 0 & 0.5 \end{bmatrix}$$

3. First Layer Output ($Z^{(0)}$)

$$Z^{(0)} = \begin{bmatrix} 0 & 0.65 \\ 0 & 0.43 \\ 0 & 0.43 \\ 0 & 0.43 \end{bmatrix}$$

4. Second Layer Output ($Z^{(1)}$)

$$Z^{(1)} = \begin{bmatrix} 0 & 0.37 & 0.31 \\ 0 & 0.27 & 0.22 \\ 0 & 0.27 & 0.22 \\ 0 & 0.27 & 0.22 \end{bmatrix}$$

The matrix $\mathbf{Z^1}$ represents the embeddings of the nodes. We observe that the first coordinate of each embedding is always 0, due to the ReLU activation function. If the node features $\mathbf{X}$ had been randomly sampled, the features would all have been distinct, leading to distinct embeddings as well. In that case, no repeating pattern would appear in the embedding matrix $\mathbf{Z^1}$.

Node classification and representation on Cora

The GNN is applied to the Cora dataset, where the node features are included in the dataset. Test loss and accuracy are calculated:

- Test Loss: 60.65%
- Test Accuracy: 84.32%

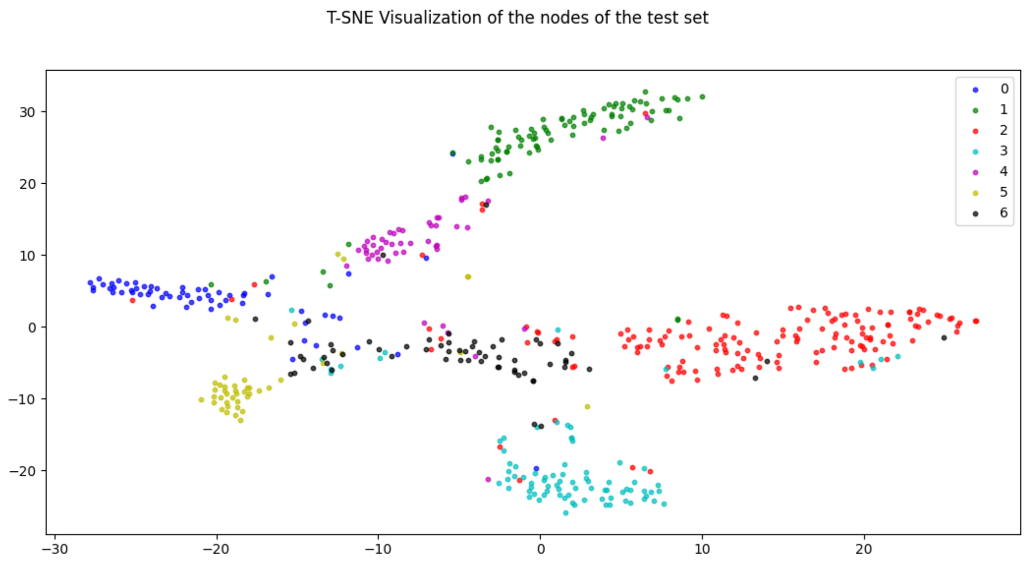

Figure 3: Classification on the Cora dataset using GNN & t-SNE

The classification results on the Cora dataset are very satisfying.