Link Prediction using GNNs
Deep Learning | MSc Artificial Intelligence

Andreas Sideras

NCSR Demokritos & University of Piraeus

June 2023

Outline

1. Graph Analysis
2. GNNs
3. Link Prediction
4. Results

Table of Contents

1. Graph Analysis
2. GNNs
3. Link Prediction
4. Results

Graph

- We used a subset of "Statistics and Social Network of YouTube Videos" that comprises of 9064 nodes and 43963 edges.
- Each node represents a YouTube video and it is described with 7 attributes.
- Each video is correlated with some other videos. This relation is represented with directed edges in our graph.

Connected Subgraph

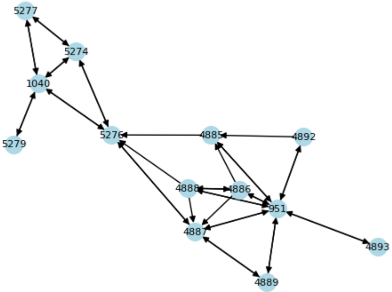


Figure: A random strongly connected component.

Graph Analysis

- A network can be an exceedingly complex structure, as the connections among the nodes can exhibit complicated patterns.
- By plotting the distributions of certain metrics, we can gain meaningful insights about our graph.

Scale Free Property

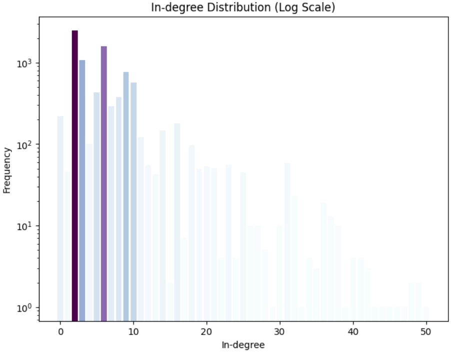


Figure: In-coming edges distribution

Asymmetric Graph

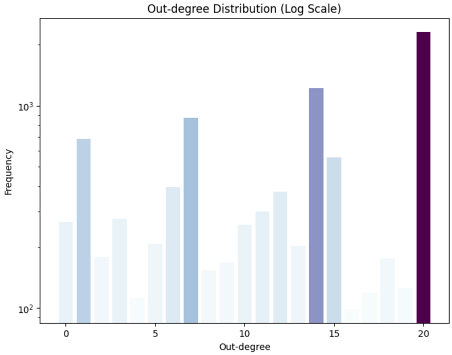


Figure: Out-coming edges distribution

Small World Property

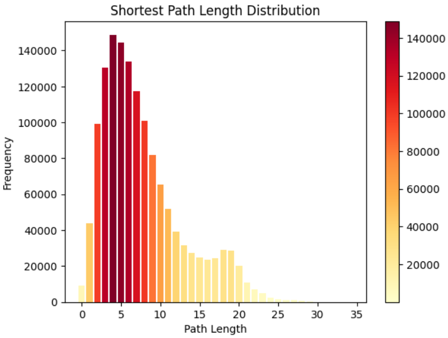


Figure: Shortest path distribution

Table of Contents

1. Graph Analysis
2. GNNs
3. Link Prediction
4. Results

GNNs

- GNNs are a class of neural networks specifically designed to operate on graph-structured data.
- They capture both the structural information of the network and the node features, allowing them to learn complex patterns and dependencies among nodes.
- GNNs operate by iteratively aggregating information from neighboring nodes and updating node representations.
- This process allows nodes to gather information from their local neighborhoods and propagate it through the graph (*message passing*).
- Learn new features/embeddings for each node.

Graph Convolutional Networks

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \frac{1}{|\mathcal{N}(v)|} W^{(l)} h_u^{(l)}\right)$$

- $h_u^{(l)}$ represents the feature vector of a neighboring node u of v at layer l.
- $\mathcal{N}(v)$ denotes the set of neighboring nodes of v.
- $|\mathcal{N}(v)|$ represents the degree of node v, which indicates the number of neighbors of node v.
- $W^{(l)}$ is the weight matrix associated with the l-th layer.
- σ is a non-linear activation function applied element-wise to introduce non-linearity (e.g., ReLU or sigmoid).

GraphSAGE

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \mathsf{CONCAT}\left(h_v^{(l-1)}, \mathsf{AGGREGATE}\left(\{h_u^{(l)} \mid u \in N(v)\}\right)\right)\right)$$

- $h_v^{(l)}$ represents the representation of node v at layer l.
- σ is the activation function (e.g., ReLU, sigmoid) applied element-wise.
- $W^{(l)}$ is a learnable weight matrix for layer l.
- AGGREGATE is an aggregation function that combines the representations of neighboring nodes. It can be a simple mean, max, or sum aggregation, or more complex functions like attention mechanisms.

Graph Attention Networks

$$h_v^{(l+1)} = \sigma \left(\sum_{u \in N(v)} \alpha_{vu}^{(l)} \cdot W^{(l)} \cdot h_u^{(l)} \right)$$

where:

- $h_v^{(l)}$ represents the representation of node v at layer l.
- σ is the activation function (e.g., ReLU, sigmoid) applied element-wise.
- $\alpha_{vu}^{(l)}$ represents the attention coefficient between nodes v and u at layer l. It is computed as:

  $$\alpha_{vu}^{(l)} = \text{softmax} \left(\vec{a}^{(l)} \cdot \left[W^{(l)} \cdot h_v^{(l)}, W^{(l)} \cdot h_u^{(l)} \right] \right)$$

  where $\vec{a}^{(l)}$ is a learnable weight vector specific to layer l and $[\cdot, \cdot]$ denotes concatenation.
- $W^{(l)}$ is a learnable weight matrix for layer l.

Graph AutoEncoder

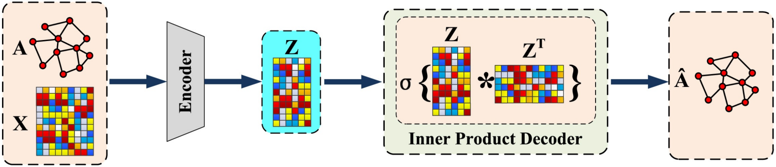


Figure: GAE architecture

- Learns embeddings in latent space using an Encoder network.
- Decodes the representations and reconstructs the initial graph.

Table of Contents

1 Graph Analysis

2 GNNs

3 Link Prediction

4 Results

Link Prediction

- We will use the GAE architecture in order to predict the existence of an edge between two nodes.
- We augmented the feature node vectors with some features extracted from the graph, like eigenvector and closeness centrality, pagerank score etc.
- The feature vectors for all the nodes are passed through the following encoder network resulting in a new 64-dimensional representation for each node.

```
GAE(
  (linear1): Linear(in_features=15, out_features=64, bias=True)
  (tanh): Tanh()
  (linear2): Linear(in_features=64, out_features=256, bias=True)
  (tanh): Tanh()
  (linear3): Linear(in_features=256, out_features=1024, bias=True)
  (tanh): Tanh()
  (conv): GNN(1024, 256, num_layers=2)
  (linear4): Linear(in_features=256, out_features=64, bias=True)
)
```

Link Prediction

- The decoder is an inner-product computation, followed by a sigmoid function that predicts the likelihood for the connection of two nodes.
- In each iteration, we use the the positive edges of our graph (their corresponding nodes' representations) and a sample of negative edges.
- The decoder computes $\sigma(\text{InnerProduct}(z_{start}, z_{end}))$ which is the probability for the edge from node z_{start} to node z_{end} to exist (z_i is the latent representation of each node).
- The result goes into a Binary Cross Entropy Loss followed by backprop and an Adam's optimizer step.

Table of Contents

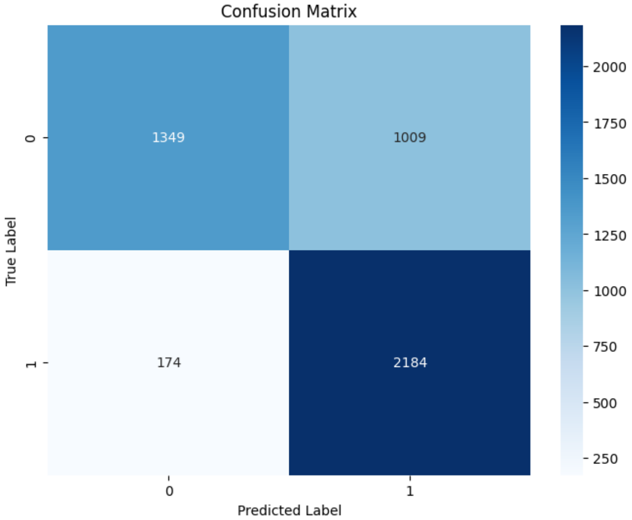
1. Graph Analysis
2. GNNs
3. Link Prediction
4. Results

Results

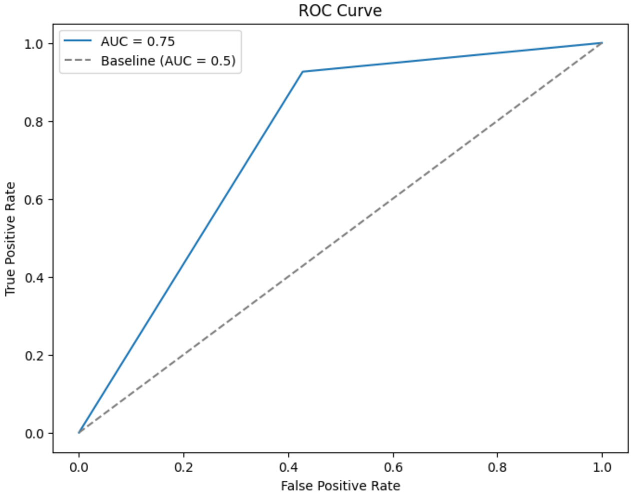
Table: Test Performance Metrics

GNN	Precision	Recall	F1 Score	ROC AUC Score
GCN	0.684	0.926	**0.787**	0.749
GAT	0.694	0.904	0.785	0.753
SAGE	0.674	0.907	0.773	0.734

Confusion Matrix for GCN



ROC Curve


ROC Curve
AUC = 0.75
Baseline (AUC = 0.5)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Thank you!