Political blog classificaiton with GCN

Farano Giuseppe

Introduction and problem description

Objective: to classify political blogs as 'liberal' or 'conservative'.

Dataset: Polblogs

Theoretical references: Graph Convolutional Networks

Tools used: Python and its libraries

Data preprocessing

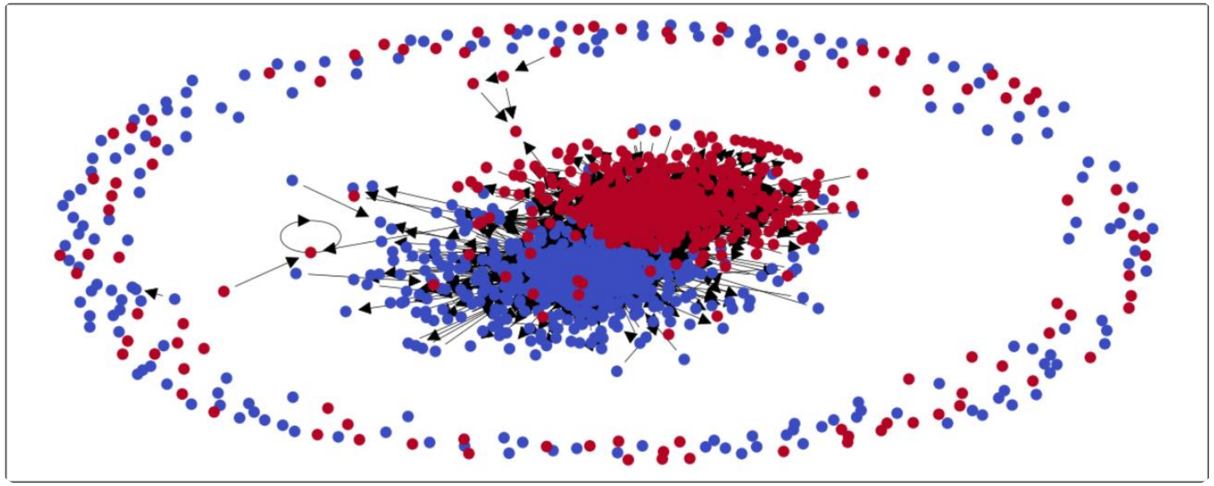
Dataset loading and analysis

- Collection of the political blogs that were most active in 2004 in the run-up to the US presidential elections.
- Zip folder containing a text document and a GML file
- Igraph library
- Grafo consisting of 1490 vertices e 19090 edges
- Node attributes: id, label, value, source
- Presence of isolated vertices

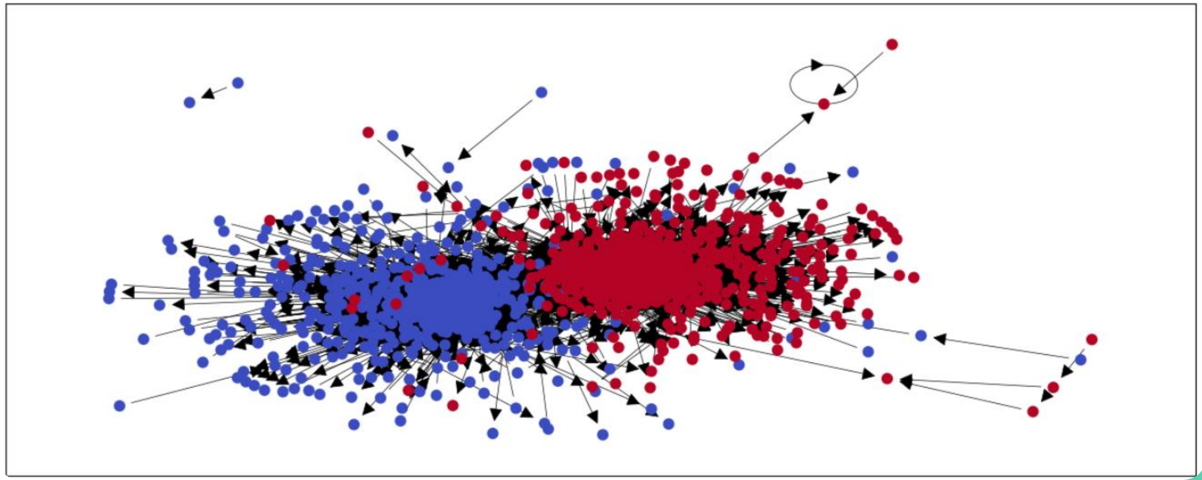
Dataset visualization

- networkx and matplotlib libraries
- Liberal blogs in blue, conservative blogs in red

Complete graph



Graph without isolated vertices



Text embeddings creation

- Web Scraping with BeautifulSoup
  - Classes: «post-body entry-content», «entry-body», «post»
  - 460 blog unreachable
- Text to embeddings with Bert
- PCA
  - 134 dimensions

Data preparation

- Object of «Data» type by torch_geometric
  - x, edge_index, y

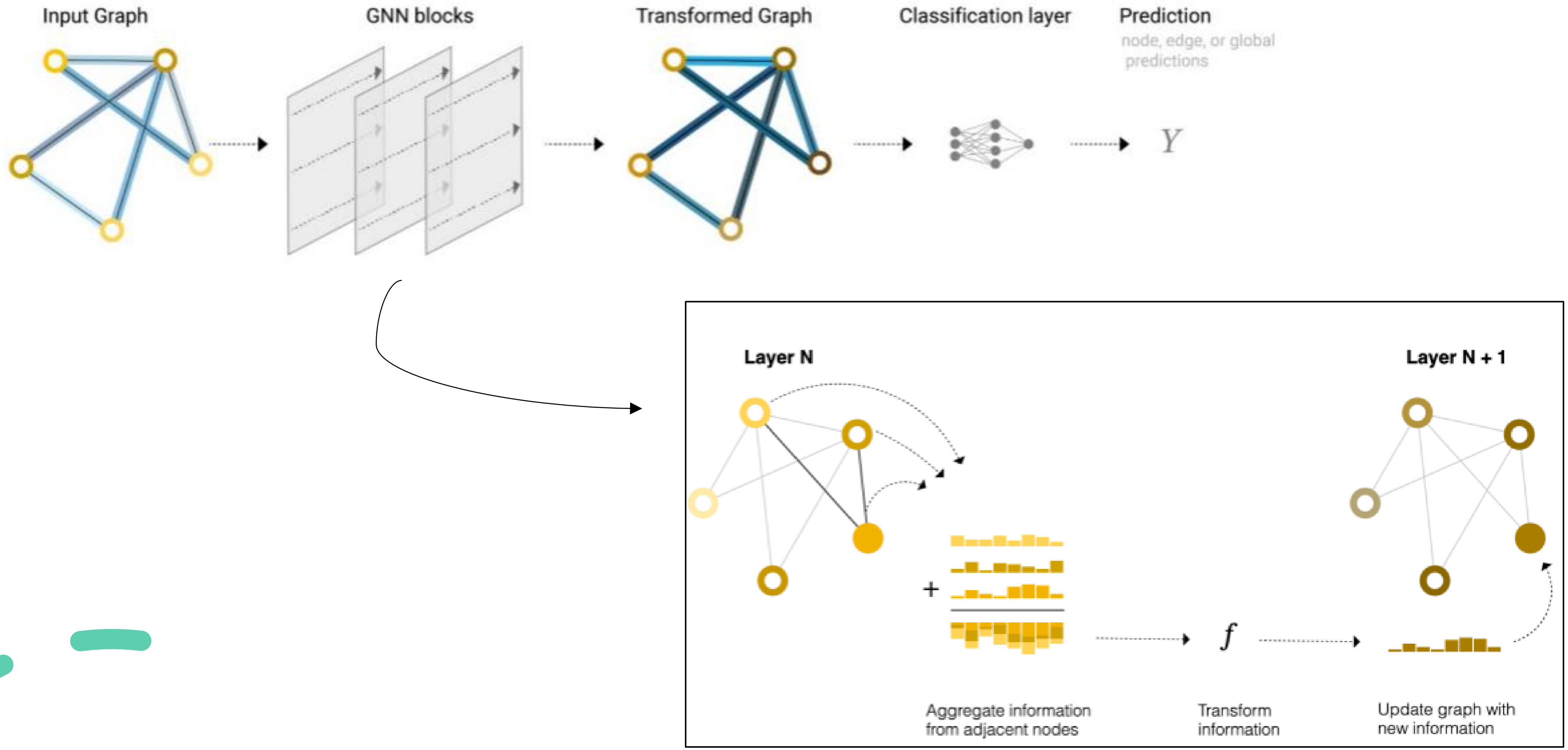
- Data shuffle to avoid bias

```
data.y
tensor([0., 0., 0.,  ..., 1., 1., 1.], dtype=torch.float64)
```

```
data_shuffled.y
tensor([0., 0., 1.,  ..., 1., 0., 0.])
```

- Split in training set, validation set, test set
  - train_test_split di sklearn
  - RandomNodeSplit not used because limited aboud seed and stratification

GNN introduction



Insight into GCNs

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \frac{\sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Mean of v's neighbour's embeddings at step $k-1$.

Node v's embedding at step $k-1$.

Color Codes:

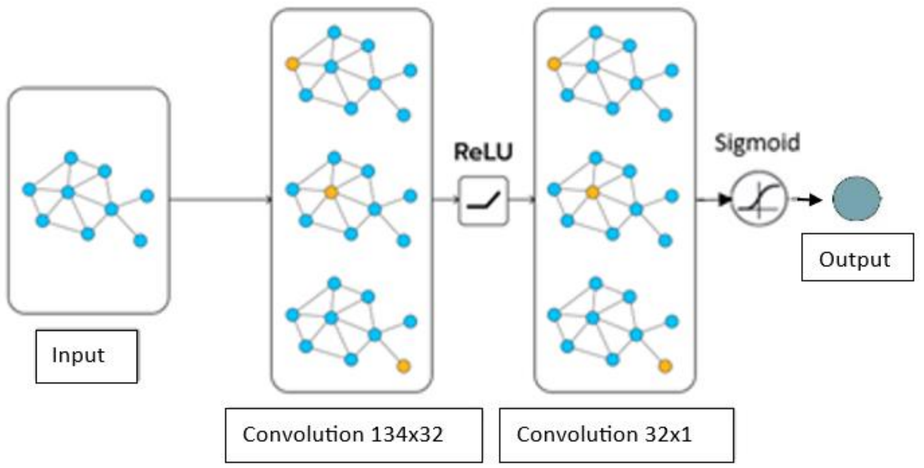
- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

- Node aggregation $m_i^{(l)} = \sum_{j\in N(i)} \frac{1}{\sqrt{d_i d_j}} h_j^{(l-1)}$
- Node update $h_i^{(l)} = \sigma\left(W^{(l)} m_i^{(l)}\right)$

First architecture

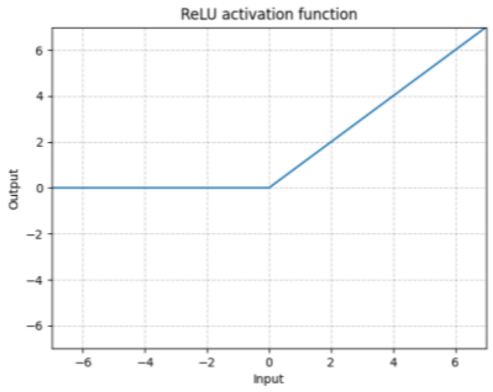
Model definition

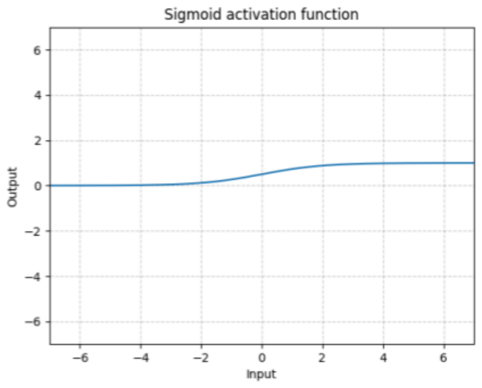
- Inspired to [Semi-Supervised Classification with Graph Convolutional Networks]



$$\text{ReLU}(x) = (x)^+ = \max(0,x)$$

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1+exp(-x)}$$





Training and performance evalutation

- Cost function: BCE

$$l_n = -w_n[y_n \log x_n + (1-y_n)\log(1-x_n)]$$

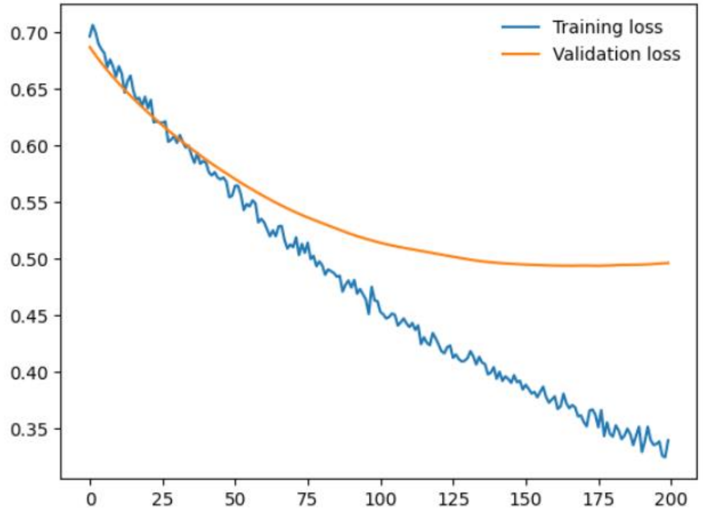
- Optimization algorithm: Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$$

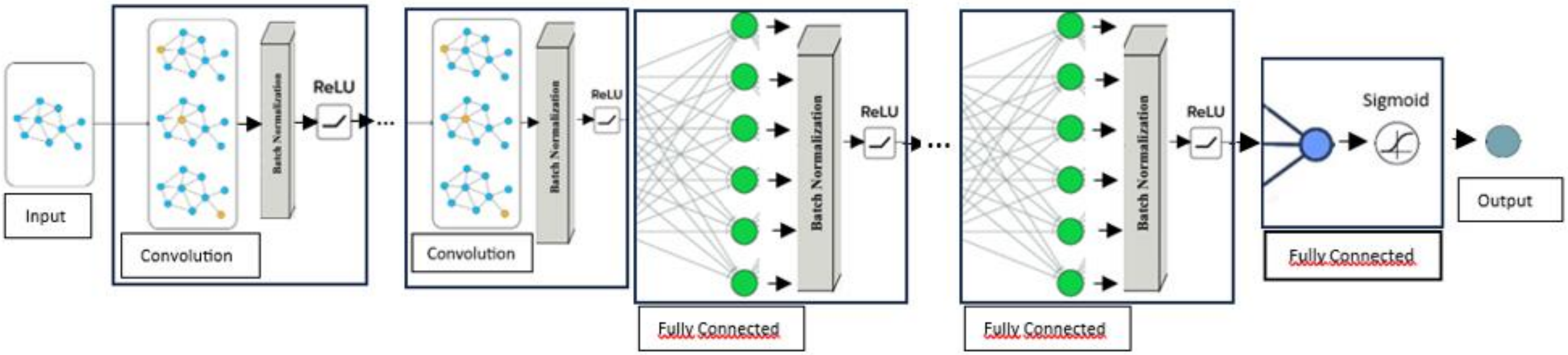
- Test set accuracy: 77.5%
- Learning curves



Architecture improvement

Architectural changes

- Added normalization layers
- Added linear layers



Training loop changes

- Weights initialization according to Xavier (Glorot)

$$w \propto \frac{np.random.randn()}{n_{in} + n_{out}} \Rightarrow \mathcal{N}(0, \sigma^2) \qquad \sigma = gain \times \sqrt{\frac{2}{n_{in} + n_{out}}}$$

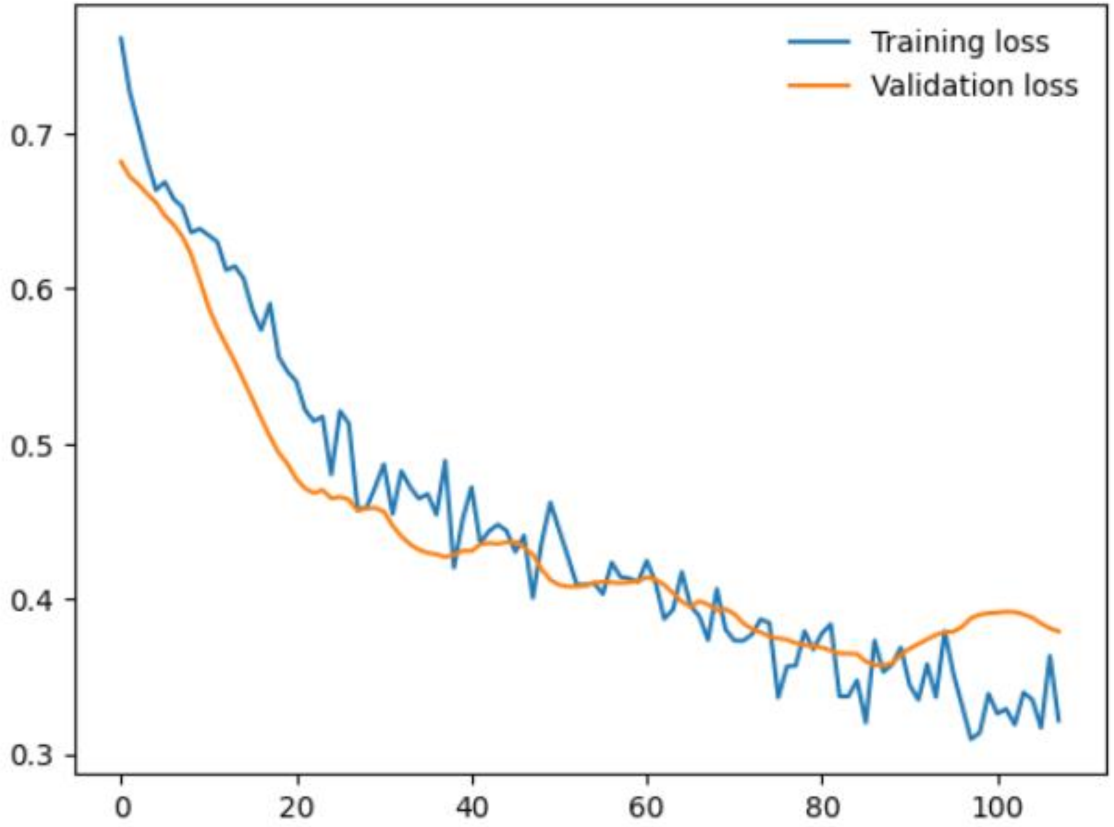
- Early stopping
- AdamW

New model training

- Grid search with following hyperparameters:
  - Number and dimension of Convolutional Layers
  - Number and dimension of Linear Layers
  - Dropout probability
  - Learning rate
  - Weight decay

Validation

- Best model showed 87% accuracy on validation set.
  - 4 convolutional layers with 16 neurons each one
  - 2 linear layers with 16 and 8 neurons
  - Dropout probability equal to 30%
  - Learning rate 0.01
  - Weight decay 1e-04

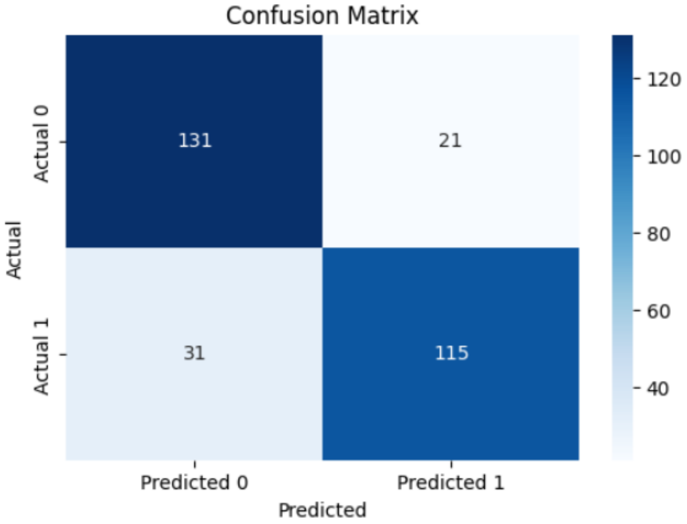


Test set results

- Predictions on random 20 samples from test set

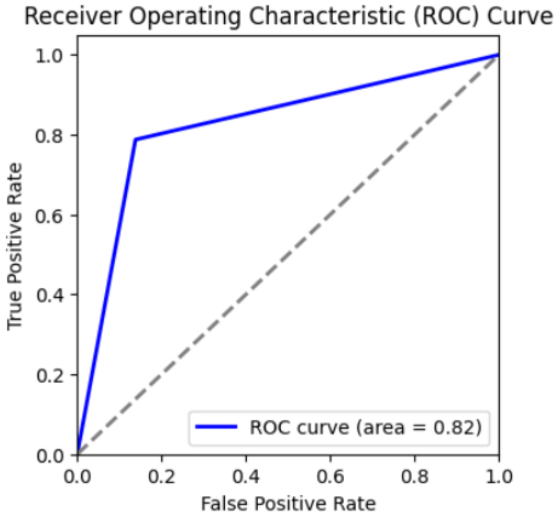
```
First 20 rows test_output:
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.93, Percentage for Conservative: 0.07      Prediction: 0, Target: 0
Percentage for Liberal: 0.85, Percentage for Conservative: 0.15      Prediction: 0, Target: 0
Percentage for Liberal: 0.58, Percentage for Conservative: 0.42      Prediction: 0, Target: 0
Percentage for Liberal: 0.91, Percentage for Conservative: 0.09      Prediction: 0, Target: 0
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.81, Percentage for Conservative: 0.19      Prediction: 0, Target: 0
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.88, Percentage for Conservative: 0.12      Prediction: 0, Target: 1
Percentage for Liberal: 0.90, Percentage for Conservative: 0.10      Prediction: 0, Target: 0
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.92, Percentage for Conservative: 0.08      Prediction: 0, Target: 0
Percentage for Liberal: 0.86, Percentage for Conservative: 0.14      Prediction: 0, Target: 1
Percentage for Liberal: 0.06, Percentage for Conservative: 0.94      Prediction: 1, Target: 1
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.10, Percentage for Conservative: 0.90      Prediction: 1, Target: 1
Percentage for Liberal: 0.05, Percentage for Conservative: 0.95      Prediction: 1, Target: 1
Percentage for Liberal: 0.92, Percentage for Conservative: 0.08      Prediction: 0, Target: 0
```

Confusion matrix, classification report and ROC curve



Test set Classification Report:

	precision	recall	f1-score	support
0	0.81	0.86	0.83	152
1	0.85	0.79	0.82	146
accuracy			0.83	298
macro avg	0.83	0.82	0.82	298
weighted avg	0.83	0.83	0.83	298

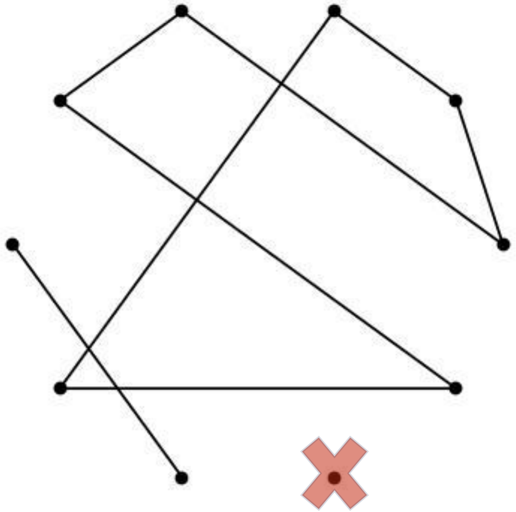


Critical Aspects

- Input to BERT for unreachable websites
- Choose of dimensionality reduction technique
- Isolated nodes management

Points of improvement

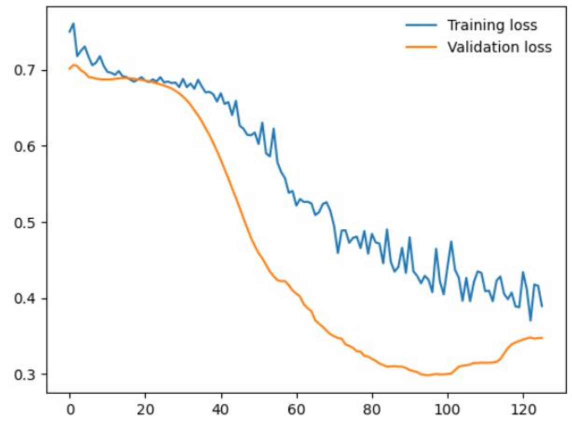
- Use a dataset without unreachable websites
- Explore other techniques to obtain text documents from the blogs
- Explore other techniques of text embeddings creation
- Use other layer types
- Train with cross validation

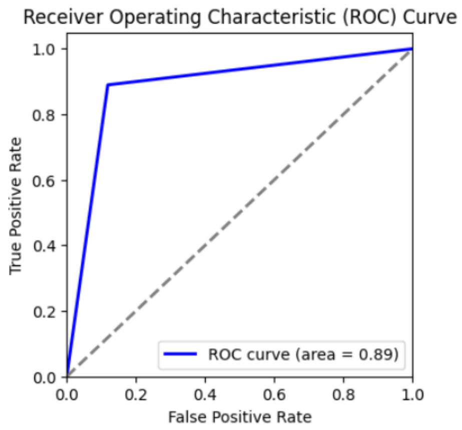


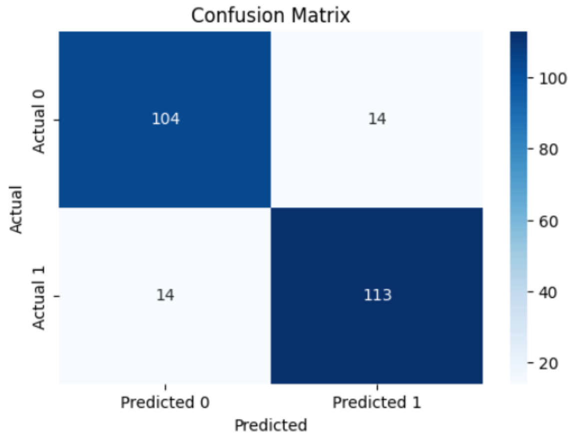
Isolated nodes exclusion

Model training on the new dataset

- The graph dataset has 266 isolated nodes
- Both two architectures are trained with the filtered input
- The first model showed a 88.16% accuracy
- In the validation phase of the second model, the same hyper-parameters were selected; the only exception was the dropout at 50% instead of 30%.
- The best model selected in the validation phase achieved an accuracy of 88.57%.







Test set Classification Report:

	precision	recall	f1-score	support
0	0.88	0.88	0.88	118
1	0.89	0.89	0.89	127
accuracy			0.89	245
macro avg	0.89	0.89	0.89	245
weighted avg	0.89	0.89	0.89	245

Thank you for your attention

Farano Giuseppe