Data Challenge

Professor: Giannis Nikolentzos

Domain Name Classification Challenge

Data Challengers

Athens, June 2024

Team members:

Our team (*data challengers*) comprised of the following members:

Dimitrios Stathopoulos

Dionysios Voulgarakis

Vasileios Ilias Drouzas

In the following, we demonstrate the basic methods and ways in general that we faced the challenge, both successful and unsuccessful ones. At first, we experimented with the text methods (TF-IDF, SVD) and when we noticed that these procedures had reached an optimal score, we decided to test the graph since we were confident we would achieve better scores this way.

TEXT

In the text part of the assignment, the code procedure looks like this:

1. Text pre-processing
   a. Converting to lowercase
   b. Removing accents and tones.
   c. Removing punctuation & numbers
   d. Tokenization
   e. Removing stop-words
   f. Lemmatization of tokens
   g. Hyper-links removal
   h. Join of tokens back into text

2. TF-IDF, FASTTEXT, Doc2Vec, Bert

3. SVD to reduce dimensionality to 5000 features of TF-IDF

4. Tried to handle class imbalance with under-sampling, over- sampling and SMOTE but did not see any improvements

5. Tried to split texts to subtexts (*#*), but didn’t improve the score it test set.

6. Classification (using Grid-Search and Random-Search for hyperparameter tuning):
    a. Logistic Regression
    b. Naïve Bayes
    c. SVM
    d. 5-fold Cross Validation
    e. Random Forest Classifier
    f. Adaboost
    g. Extreme Gradient Boosting
    h. MLP
    i. BERT with full model training

```
Epoch 10/10, Train Loss: 0.2373
Epoch 10/10, Validation Loss: 0.9626, Validation Accuracy: 0.6813
```

    j. Many attempts with Ensemble techniques (multiple different classifiers combined each time)

7. Use of predictions from GCN with over 0.99 probability, to augment the text dataset. Improved the score slightly, but not consistently.

<u>**GRAPH**</u>

A) Use of Graph Attributes like out-degree, in-degree, degree-centrality etc. as features,

- 1. Logistic Regression
- 2. SVM
- 3. Naive Bayes
- 4. XGBoost
- 5. Ada-Boost
- 6. MLPs
- 7. Ensembles

The above implementation did not even pass the baselines set by the professor.

<u>General procedure</u>

1. Added labels & masks in the graph

2. Generated node sequences by performing Random Walk sampling.

3. Trained a Word2Vec model to obtain node embeddings.

4. Trained SDNE to create node embeddings with results half as good as word2vec.

5. Generated node embeddings using the node sequences and the word2vec model.

6. Finally, extracted node features.

7. Classification:
   a. **GAE** (GraphAutoEncoder)
   Convolutional layer for encoder & decoder, classifier layer. For the encoding, used a RELU function and for the decoding used a sigmoid

function. Trained with the CrossEntropyLoss function and used the Adam optimizer.

```
Epoch 155, Train Loss: 0.9135, Validation Loss: 1.0083
Epoch 156, Train Loss: 0.9158, Validation Loss: 1.0093
Epoch 157, Train Loss: 0.9143, Validation Loss: 1.0071
Epoch 158, Train Loss: 0.9142, Validation Loss: 1.0118
```

Validation loss ranged around 1, so this model was not tested further.

b. **GCN** (2 convolutional layers, dropout layer, batch layer and classifier layer for 9 classes)

```
GCN(
  (conv1): GCNConv(128, 128)
  (conv2): GCNConv(128, 128)
  (dropout): Dropout(p=0.5, inplace=False)
  (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

 (skip_linear1): Linear(in_features=128, out_features=128, bias=Fal
se)
  (classifier): Linear(in_features=128, out_features=9, bias=True
))
```

Trained with the CrossEntropyLoss function and used the Adam optimizer.

```
Epoch: 36, Loss: 0.5638434886932373 | val_loss 0.6078035831451416
Epoch: 37, Loss: 0.5555347800254822 | val_loss 0.6079645156860352
Epoch: 38, Loss: 0.5505193471908569 | val_loss 0.6072205901145935
Epoch: 39, Loss: 0.5460321307182312 | val_loss 0.6079316139221191
```

c. **GraphsAGE**.

Included 2 SAGEConv layers, classifier linear layer and a dropout layer. Also did hyperparameter search to find the best parameters.

```
Testing hyperparameters: {'dropout': 0.3,
'hidden_channels': 64, 'lr': 0.001, 'weight_decay':
0.0001}
```

```
Epoch 196, Validation Loss Improved: 0.8591, Model Saved
Epoch 197, Validation Loss Improved: 0.8586, Model Saved
Epoch 198, Validation Loss Improved: 0.8583, Model Saved
Epoch 199, Validation Loss Improved: 0.8583, Model Saved
```

d. **GAT**
2 convolutional GAT layers, classifier linear layer and dropout layer. Did not achieve a loss under 1 in the first 70 epochs, so it was not further tested.

```
Epoch 49, Train Loss: 1.2778, Validation Loss Improved: 1.0864, Model
Saved
Epoch 50, Train Loss: 1.2407, Validation Loss Improved: 1.0835, Model
Saved
Epoch 51, Train Loss: 1.2413, Validation Loss: 1.0967
Epoch 52, Train Loss: 1.2321, Validation Loss: 1.1146
Epoch 53, Train Loss: 1.2204, Validation Loss: 1.1204
```

e. **GIN**
Included two GINConv layers, which where composed of: a linear layer, RELU, and a linear layer again. Also, one more linear layer was included, plus the classifier and dropout layers. This model proved to be the worst one, so it was not further tested.

```
Epoch 17, Train Loss: 2.2043, Validation Loss: 2.1799
Epoch 18, Train Loss: 2.1891, Validation Loss: 2.1821
Epoch 19, Train Loss: 2.1813, Validation Loss: 2.1801
Epoch 20, Train Loss: 2.1804, Validation Loss: 2.1776
Epoch 21, Train Loss: 2.1722, Validation Loss: 2.1747
Early stopping!
Training complete.
```

<u>Pre - processing</u> attempts that did not help:

1) Removing self loops, duplicate edges, isolated nodes

2) Identifying Communities and creating node features based on them

3) Even after creating a subgraph (representative of the initial graph), calculating metrics like betweenness centrality, closeness centrality, pagerank that would assist us to generate better features proved to be a very lengthy process.