MAGNET:
Multi-Label Text Classification using Attention-based Graph Neural Network

CS7.403.S22.Statistical Methods in AI

Team Number - 06
Persistent Perceptron
Annapoorani A(2021201017)
Debashish Roy(2021201034)
Malla Hemalata(2021201066)

MAGNET

- In this project, we try implementing the paper to obtain Multi-Label Classification using Attention Based Graph Neural Networks.
- The paper proposes a novel approach that doesn't rely on methods like Binary Relevance, Classifier Chains or Label Powers as they do not take into account the label correlation in a multi-label approach.
- It uses a GAN network with Graph Convolutional network as basis.
- The correlation is understood by using Adjacency matrix and attention weights to represent the correlation matrix.
- Embeddings are taken as input, fed to our model and the output is our multi labels which classify our text.
- Link to the Github Repository: https://github.com/debashish05/SMAI_S22_06

PROJECT SCOPE

- We will be implementing bi-lstm and used “glove embedding” for the articles. We will treat this as our baseline model and compare the result of MAGNET with this model.
- We will be implementing the MAGNET model with a single Graph Attention layer for multi label text classification and compare the results with the baseline.
- A graph attention based model is used to capture the correlation between the various labels in our problem.

PROJECT TIMELINE

April 1st - Project teams announcement.

April 4th, 5th - Paper reading.

April 6th - Paper and project discussion with our mentor.

April 12th - Embedding of the input text using Bert

April 15th - Bi-LSTM and preprocessing text

April 18th - Baseline model completed

April 25th- GAT network discussion

May 4th - MAGNET Model semi implementation

Task Completed

- We have used Reuters-21578. It is a collection of documents collected from Reuters News Wire in 1987. The Reuters-21578 test collection, together with its earlier variants, has been such a standard benchmark for the text categorization (TC) (Debole and Sebastiani, 2005). It contains 10,788 documents, which has 7,769 documents for training and 3,019 for testing with a total of 90 categories.
- Link of the dataset: https://drive.google.com/drive/folders/1H1zipkqap_0yMYcajzbM1jCTUkc3ZFVg?usp=sharing
- We have implemented bi-lstm and used glove embedding for the articles. We are treating this as our baseline model and compare the result of MAGNET with this model.
- We obtained an earlier accuracy of 57% on training data and 59% accuracy on the test data.

Task Completed

- For the baseline model we are getting an accuracy of **61.5% for train data and 62.9% for test data** on hyper-tuning parameters.
- We generated the label embeddings of the train and test sets by using the glove model with embedding size as 50.
- We have also built the correlation matrix from train dataset. It is constructed by counting pairwise co-occurrence of labels. It is normalised by the frequency vector.

  A = M / F where M is the co-occurrence matrix and F is frequency vector

GAT Model

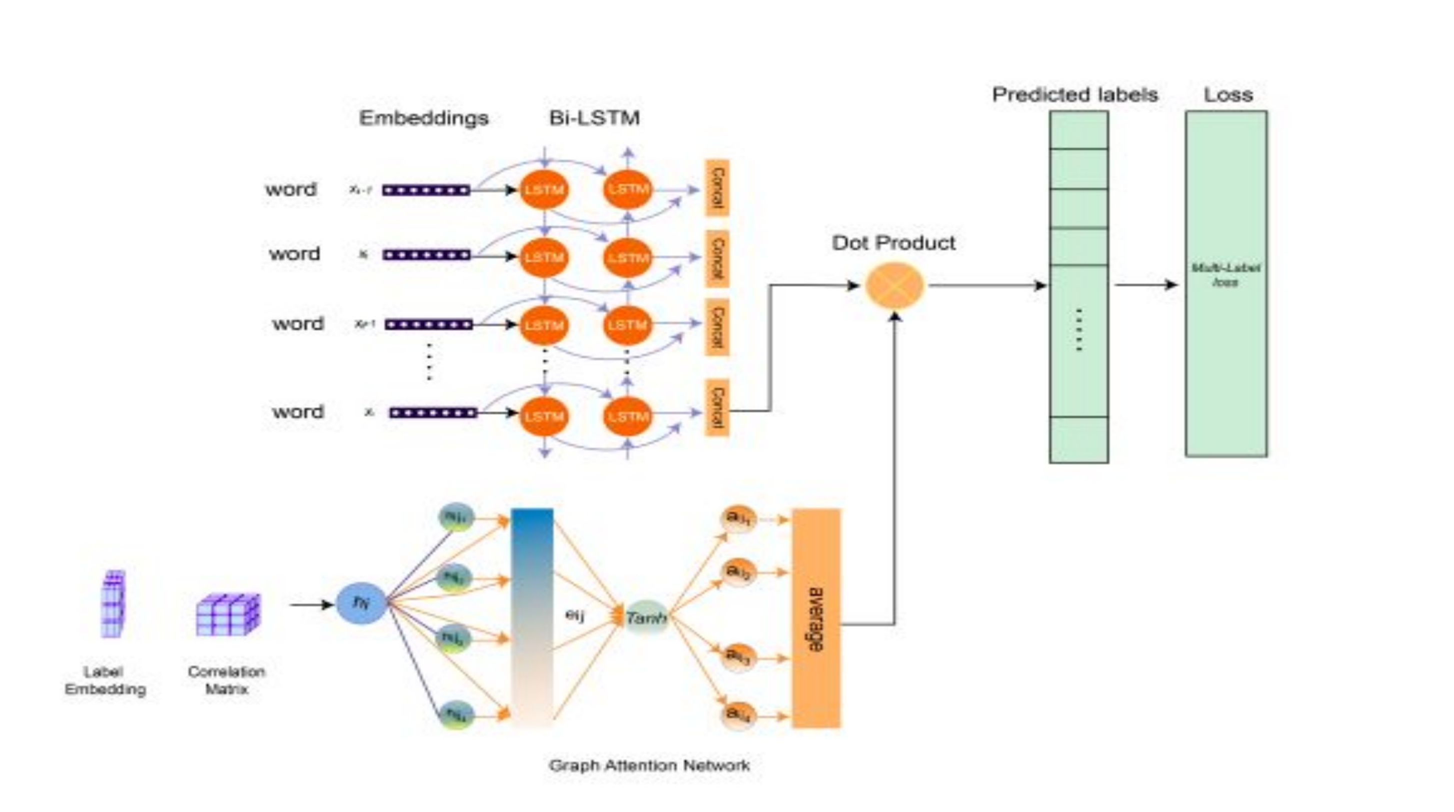
- We are using the Graph Attention Network(GAT) to implement our model.
- Label embedding and Correlation matrix are given as inputs for the GAT network.
- The GAT model uses attention coefficient which measures importance of a node, and a multi-head attention.
- The node states are updated using the convolution weight matrix, previous note states and attention coefficients for the neighbors of the node.

$$\mathbf{H}_2^{(\ell+1)} = \mathbf{ReLU}\Big(\alpha_{22}^{(\ell)}\mathbf{H}_2^{(\ell)}\mathbf{W}^{(\ell)} + \alpha_{21}^{(\ell)}\mathbf{H}_1^{(\ell)}\mathbf{W}^{(\ell)} + \alpha_{23}^{(\ell)}\mathbf{H}_3^{(\ell)}\mathbf{W}^{(\ell)} + \alpha_{24}^{(\ell)}\mathbf{H}_4^{(\ell)}\mathbf{W}^{(\ell)}\Big)$$

$$\alpha_{\mathbf{ij}} = \mathbf{ReLU}\left((\mathbf{H_i W}) \,\|\, (\mathbf{H_j W})^{\mathbf{T}}\right)$$

The model we tried to Implement

Image credit: MAGNET paper



GAT

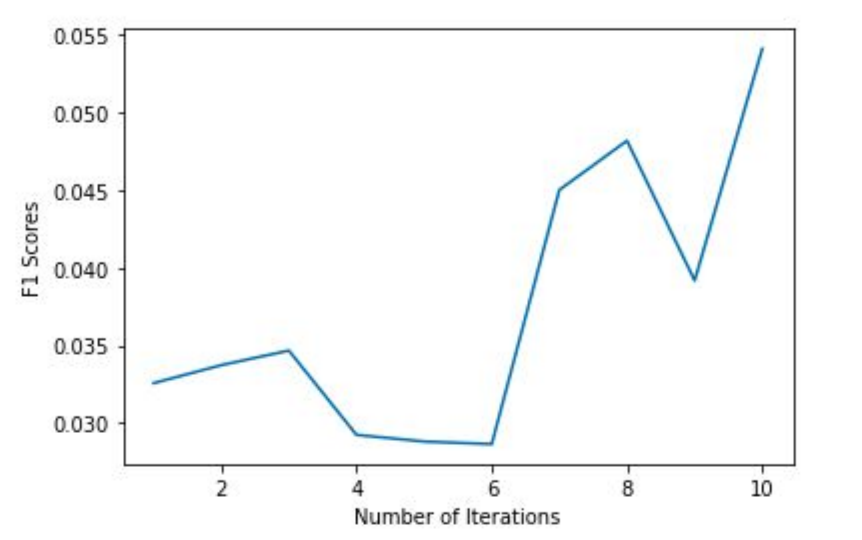
- We need to use attention weights with our GAT model, which would represent the correlation between the labels, and use GAT networks in order to learn these parameters.
- An activation function is used on our network represented in our hidden-layers, and finally the output obtained can be used to mark our multiple layers.
- A dot product has been performed with our initial Bi-LSTM model output to predict labels and determine the loss.

GAT Model Summary

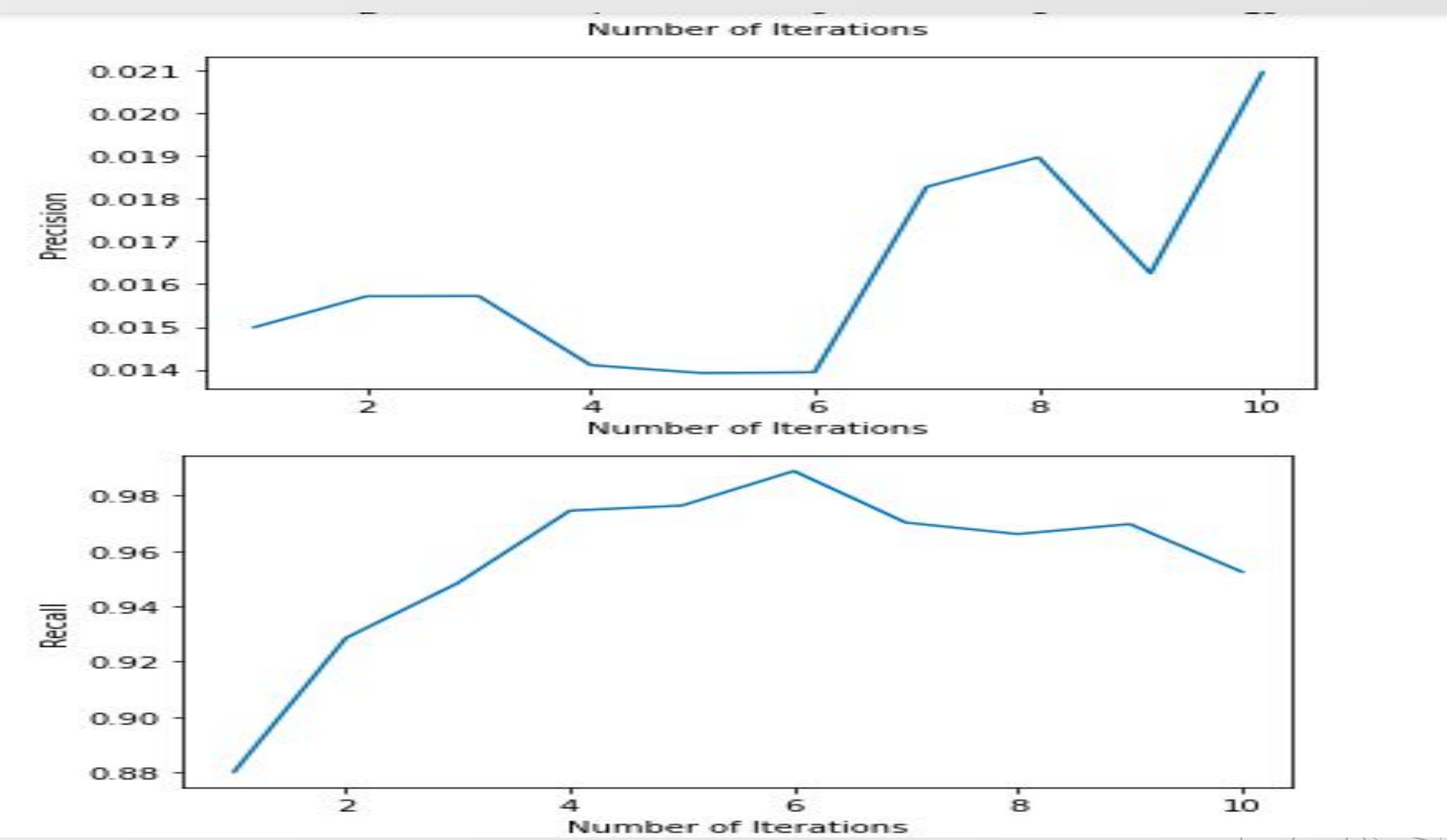
```
MAGNET(
  (embedding): Embedding(58675, 50)
  (rnn): LSTM(50, 250, batch_first=True, bidirectional=True)
  (gat): GAT(
    (gat1): GraphAttentionLayer(
      (W): Linear(in_features=50, out_features=500, bias=False)
      (a): Linear(in_features=1000, out_features=1, bias=False)
      (leakyrelu): LeakyReLU(negative_slope=0.01)
      (softmax): Softmax(dim=1)
    )
    (gat2): GraphAttentionLayer(
      (W): Linear(in_features=500, out_features=500, bias=False)
      (a): Linear(in_features=1000, out_features=1, bias=False)
      (leakyrelu): LeakyReLU(negative_slope=0.01)
      (softmax): Softmax(dim=1)
    )
  )
  (dropout): Dropout(p=0.5, inplace=False)
)
```

Plots

Number of iterations vs F1-score



Plots



Number of Iterations
Precision
0.021
0.020
0.019
0.018
0.017
0.016
0.015
0.014
2
4
6
8
10
Number of Iterations
Recall
0.98
0.96
0.94
0.92
0.90
0.88
2
4
6
8
10
Number of Iterations

References

https://www.youtube.com/watch?v=A-yKQamf2Fc

https://keras.io/examples/graph/gat_node_classification/

https://arxiv.org/abs/2003.11644

https://towardsdatascience.com/graph-attention-networks-under-the-hood-3bd70dc7a87

https://github.com/adrinta/MAGNET

https://towardsdatascience.com/graph-attention-networks-in-python-975736ac5c0c

THANK YOU