Sentiment Analysis using Abstract Meaning Representation and Graph Neural Network

Shahad Althobaiti, Kaan Avci, Zena-Marie Gonzalez, Da Shi

Introduction

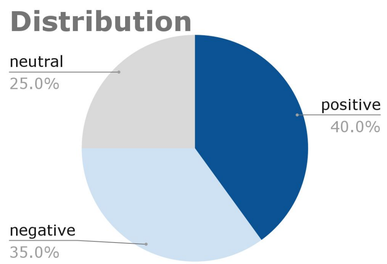
Graph Neural Networks are a promising approach in Natural Language Processing that have applications in dependency parsing and question answering systems [1][2]. This motivates us to build **a text classification model** that utilizes **Abstract Meaning Representation (AMR) and a Graph Neural Networks (GNN)**. We use the AMR to parse sentences into graph structure to be trained in GCN. For evaluation, we train our proposed model with a text classification task and compare it to baseline models such as Logistic Regression, as well as with state-of-the-art models such as BERT classifier.

Previous Work

To the best of our knowledge, there is no study that incorporates both AMR and GNN in the domain of text classification. A study, proposed using Graph Neural Network for text classification and their method achieved the state of the art performance[3]. They represented the text as a graph by linking the adjacent words together; the number of adjacent words connected to each word in the graph can be set as the prediction window size. The GNN based model was evaluated and compared to other models such as LSTM and CNNs with BoW and GloVe embeddings.

Dataset and Resources

For the dataset, we used randomly selected sample of 2000 tweets from Kaggle's coronavirus tweets for sentiment classification focusing on three classes - positive, negative, and neutral [4].



The pie chart shows the distribution of data per label. The ratio of the labels is maintained from the original data. We divided our data into 80% for training and 20% for testing.

Pipeline and Architecture

1 Preprocessing

To properly construct AMR we need to have the tweets in a proper language. Therefore, we used GPT-3's Davinci engine for tweets grammar correction[5].

Original Tweet: Found this at @Target You see someone walking out with this- you know who to befriend… #coronavirus #toiletpaper #ToiletPaperApocalypse #QuaratineLife https://t.co/cRqSGtp8D7
Processed Tweet: I found this at Target.

2 AMR Representation

Obtain the AMR graph for each tweets using AMRlib; a python module for AMR Representation[6].

```
(f / find-01
    :ARG0 (ii / i)
    :ARG1 (t / this)
    :location (c / company
        :name (n / name
            :op1 "Target")))
```

3 Constructing GCN Data

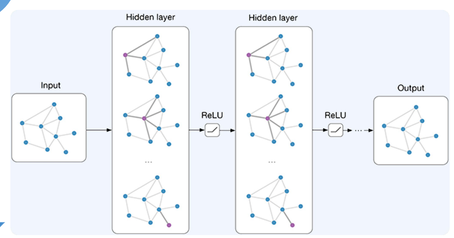
Edge List: directed edge tuples
Edges Attributes: one hot vector of AMR roles **[num_edges, num_edge_features = 109]**
Node Attributes: one hot vector of vocab **[num_nodes, num_node_features = 5138]**

Edges Attributes

Edge	ARG0	ARG1	Location	name	op1
0	1				
1		1			
2			1		
3				1	
4					1

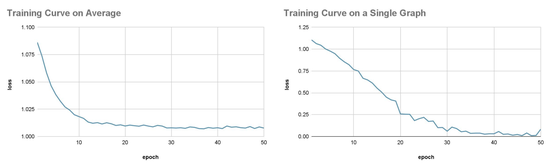
4 Building GCN

For building GCN, we used PyTorch Geometric [7]. The GNN model includes 2 GCN Conv layers and ReLU activation functions. The model was trained with 5 epochs and loss was calculated using negative log likelihood.



Experiments

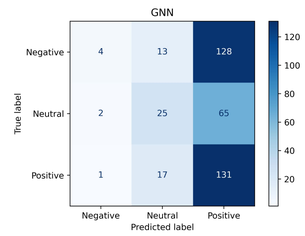
For evaluating our model's performance, we built various classification models. We used sklearn to represent the text with tf-idf and train models. The text was cleaned (removing stop words and punctuations) and lemmatized. We also built a BERT classifier using bert-sklearn, scikit-learn wrapper to finetune BERT[8]. We are using Accuracy/F1-score as the main metrics for comparison.

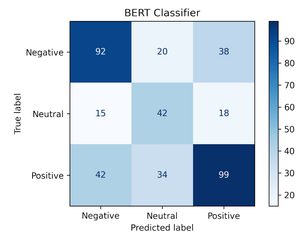


Additionally, we experimented with different layers in the architecture, eliminating the drop-out, using linear vs convolutional layers, and experimenting with the number of epochs and batch sizes.

Results

Model	GNN	Decision Trees	Random Forest	Logistic Regression	Gradient Boost	XGBoost	BERT
Accuracy	41.7%	38.75%	49.25%	51.75%	33.5%	44.0%	58.25%





Conclusion

Our experiment shows promising results on tweets sentiment analysis. A two-layer GNN yielded results outperforming Decision Trees and Gradient Boost and comparable to XGBoost. We believe that with more complex architectures (more layers), larger dataset and hyperparameter tuning, GNN can compete with BERT model.

Challenges:

- Word order not maintained (unlike BERT)
- Sparsity in distribution of words due to small dataset.
- Unlemmatized text (unlike those used for training the baseline models)

Future Work:

- Use a larger training set to experiment with larger models
- Address sparsity: replace low-frequency words with a single token

References

[1] Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based Dependency Parsing with Graph Neural Networks.
[2] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering.
[3] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text Level Graph Neural Network for Text Classification.
[4] Coronavirus tweets NLP - Text Classification
[5] GPT-3 Grammar Correction
[6] AMRlib python library
[7] pytorch-geometric
[8] bert-sklearn