Casting text classification to Graph Classification for Sentiment Analysis of Tweets 3-cfu Project Work

Dell'Olio Domenico, Delvecchio Giovanni Pio, and **Disabato Raffaele**
Master's Degree in Artificial Intelligence, University of Bologna
{ domenico.dellolio, giovanni.delvecchio2, raffaele.disabato }@studio.unibo.it

Abstract

Text Classification is a classical problem in NLP and has a large variety of applications[5]. The rise of Deep Learning and Large Language Models has greatly boosted performances on this task in recent years. In particular, Graph Neural Networks (GNNs)[10] and Graph Attention Networks (GATs)[8] have enabled leveraging external knowledge (e.g. knowledge graphs) to enhance accuracy and reduce latency. In this paper we investigate the performances of DNN models based on GCNs and GAT on Kaggle's "Coronavirus tweets NLP" sentiment analysis challenge, using dependency graphs as 'External' knowledge. We compare them with a GLoVe+LSTM baseline and a BERT-based model[3]. Our best GNN model improves on the baseline by ≈2%, reaching 68% of test accuracy, but is still behind the BERT (84% acc.), probably due to embedding over-smoothing and dependency graph extraction problems.

1 Introduction

In the field of NLP, Text Classification is one of the fundamental research problems that focuses on associating a class from a given set to a span of text. Social Networks like X (former Twitter), allow people to express their opinions on various topics, creating a relationship many to few with politicians, entrepreneurs, influencers and many other figures. Sentiment Analysis is a narrower field of application of Text Classification that allows the discovery of knowledge related to the reputation of a person/product/brand and gathering feedback, especially on social networks, in such a way that opinions are condensed according to a pre-defined scheme of emotions and can be used as insights to guide decisions. Since the proposal of BERT-like models[3], they've consistently excelled in text classification tasks, showcasing their adeptness in leveraging context for accurate categorization. However, new approaches involving Graph Neural Networks (GNNs) have been recently applied to solve and improve performances on these tasks. Modeling sentences as a graph better captures the complex relations between words and allows to inject syntactic knowledge into each sentence, enhancing model performance. H. Wang and F. Li propose an example in [9] where an intuitive architecture made of LSTMs [7] followed by two GAT modules [8], appears to surpass BERT [3] in classification tasks across different datasets. Similar techniques, like the one in [4], consider the graph of the adjacent words to model the context of sentences. This, coupled with a custom message-passing mechanism, aims to capture the global context of the sentence, thus leading to increased performances compared to BERT and similar performances to [9]. Motivated by the work of Wang and Li [9], this paper explores the capabilities of GNNs in the context of a Kaggle sentiment analysis challenge ("Coronavirus tweets NLP") both on performances, as well as a means to add External Knowledge, in the form of Syntactic Dependency Graphs, to the representation of tweets. We assumed that additional information could help in the classification, given that the brevity of these inputs can be an issue, particularly in sentiment analysis, where the meaning of a few keywords can outweigh those of others. We analyze the performances of models that can be summarized as a GloVe [6] or BERT embedding, followed by LSTMs and a GAT or GCN module on a validation split (16% of the dataset). Then, we compare the best model with a GloVe+LSTM baseline and a finetuned BERT on a test split (20% of the dataset). The GNN model outperforms the baseline by a ≈2%, reaching 68.74% of accuracy but is still behind BERT(84.29% acc.). This unexpected performance gap may be improved by avoiding embedding over-smoothing (caused by the shortness of the tweets), learning the sentence structure to serve as External Knowledge or using better word embeddings.

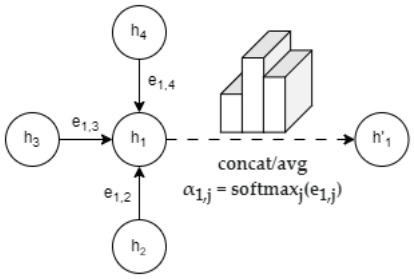

Figure 1: Graph Attention Mechanism

2 Background

With 'Graph Neural Networks' we denote an entire family of models, spanning from the earliest works like Recurrent Graph Neural Networks[12] to the most recent, convolution-based models, like Cluster-GCN[2]. For the purpose of this study, two are the important GNNs to be considered:

- Graph Convolutional Network [10], as it is employed in [11]. Is one of the models used for comparison in [4] and it involves simple intuitions.
  The GCN layer is defined[1] as:

$$x_i^{(k)} = \sum_{j \in N(i) \cup \{i\}} \frac{1}{\sqrt{deg(i)}\sqrt{deg(j)}} (W^T x_j^{(k-1)}) + b$$

  Where $x_i^{(k-1)} \in \mathbb{R}^F$ denotes the node features of node i at layer $k-1$, W is a weight matrix, N is the neighbourhood function and b is a bias vector.

- Graph Attention Network[8], as it represents the main innovation in [9] and could potentially be more expressive than standard attention mechanisms for this task.

  The GAT layer is defined as:

$$h_i' = \Big\Vert_{k=1}^{K} \sigma \left(\sum_{j \in N_i} \alpha_{ij}^k W^k h_j \right)$$

  Where $h_i' \in \mathbb{R}^{F'}$ is the node features of node i after being processed by the graph attention layer, $\Vert$ is the symbol denoting the concatenation operation over the output of K independent attention mechanisms, σ is a nonlinearity (which is optional), $W^k \in \mathbb{R}^{\mathbb{F}' \times \mathbb{F}}$ is a weight matrix, $h_j \in \mathbb{R}^F$ denotes the node features of the neighbors of node i, and α_{ij}^k are the attention coefficients of the neighborhood of i for the k-th attention mechanism. The latter is described as:

$$\alpha_{ij} = softmax_j(e_{ij})$$
$$e_{ij} = a(Wh_i, Wh_j)$$

  where $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \mapsto \mathbb{R}$ is the shared attentional mechanism needed to compute the un-normalized attention coefficients e_{ij}.
  Thus α_{ij} are the attention coefficients normalized across the first order neighborhood of node i ($j \in N_i$).

[1] https://pytorch-geometric.readthedocs.io/en/latest/tutorial/create_gnn.html

3 System description

In order to tackle the problem of sentiment analysis of tweets, we designed three pipeline consisting of dataset cleaning(discussed deeply in section 4) dataset preprocessing, which varies for each of the proposed architectures, model training and model evaluation. More specifically, we propose:

- The simplest baseline (Fig. 2). The dataset preprocessing for this model simply consists of the removal of special characters and symbols, word tokenization using `torch`'s 'basic-english' tokenizer, and vectorization using GloVe [6](glove.twitter.27B.100d). The model itself is composed by a single LSTM layer, followed by a linear projection;

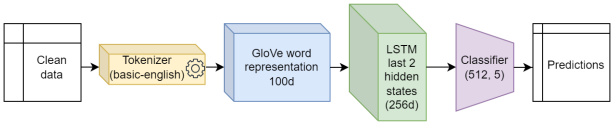

Figure 2: Simplest pipeline

- The graph convolutional pipeline (Fig. 3). Its dataset preprocessing is built on top of the one above, adding the creation of undirected graphs that will be used for classification in place of the sentences. These structures are built from the dependency graphs of each sentence, extracted and tokenized using `SpaCy`. Since each node represents a token, we make the embedding for each of them coincide with the respective word vector, obtained again

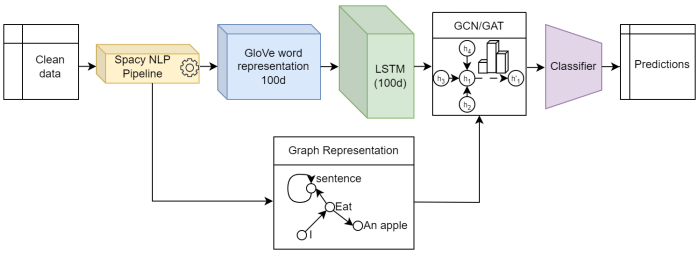

Figure 3: GCN/GAT pipeline

with GloVe. The model is based on the one proposed in [9], thus the word/node representations are processed using LSTMs and then fed to either GCN layers or GAT layers to smooth the values of the node embeddings across their neighbors. This produces the final feature sequence that is fed to a linear classifier.

- The BERT-based pipeline (Fig. 4), in which the preprocessing is the same as the simplest pipeline, but words are tokenized using BERT tokenizer. The model is simply a fine-tuned version of `'bert-base-uncased'` from HuggingFace's repository[2].

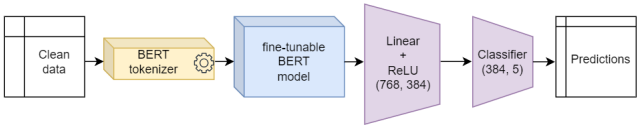

Figure 4: BERT-based pipeline

Part of the code for dataset analysis and preprocessing is taken from participants of the Kaggle challenge linked to the dataset we are using, while the remaining part was written from scratch using the libraries mentioned in section 5.

4 Data

The dataset consists of a collection of tweets about Corona Virus, extracted in 2020, and published as a Kaggle Challenge[3]. Each tweet can be labeled with one of five categories: 'Extremely Positive', 'Positive', 'Neutral Sentiment', 'Negative' and 'Extremely Negative' (Fig. 5.) The pre-processing steps adopted commonly across the models consist of the following:

- Removal of samples containing null values.
- Removal of tweets containing less than 10 characters.
- Removal of hashtags (#), mentions (@), urls (https?://), formatting symbols (tabulation, new line and such), extra spaces and numbers.

The total number of Samples after pre-processing is $39'898$. These have been split in proportions $64/16/20$ for the Training ($25'534$) Validation ($6'384$) and Test ($7'980$) sets. For what concerns the Graph-Based models, two additional pre-processing steps were performed: the extraction of the syntactic dependency tree using `SpaCy` and the construction of the corresponding graph (Fig. 6), compatible with `pytorch geometric`. This follows what has been done in [9].

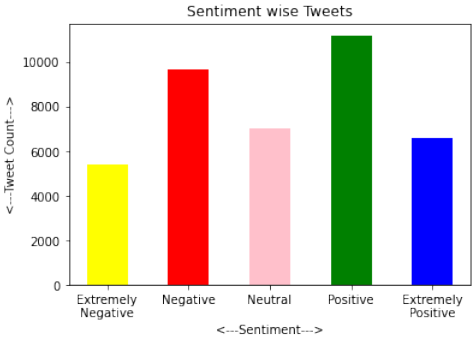

Figure 5: Count plot of the tweets grouped by sentiment

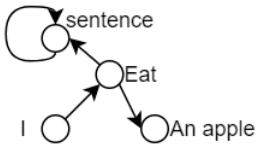

Figure 6: Visualization of a graph representing a sentence. The 'sentence' node will be initialized with an embedding representing the whole phrase, such as the centroid of the embeddings of each word

5 Experimental setup and results

Among the pipelines we introduced in section 3, the baseline and the BERT-based ones are used as lower and upper bound, respectively, for the values of accuracy in all our experiments. Because of this reason, we didn't thoroughly explore variants of

[2] https://huggingface.co/google-bert/bert-base-uncased

[3] https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification?resource=download

these models.
Regarding the graph convolutional pipeline, instead, we explored various possibilities, but we collected the most valuable ones in Table 1, together with their performances on the validation set.

	Validation Accuracy	Scheduler
GloVe+LSTM+GCN	58.29%	PolynomialLR
GloVe+LSTM+GAT	**68.95**%	CosineAnnealingLR (eta_min=1e-08)
BERT+LSTM+GAT	59.05%	CosineAnnealingLR (eta_min=1e-07)

Table 1: Graph Convolutional architectures results on validation set

The best scoring model on the validation set is then evaluated on the test set and compared with the other two pipelines in Table 2.

	Test Accuracy
GloVe+LSTM	66.36%
GloVe+LSTM+GAT	68.74%
BERT fine-tuning	**84.29**%

Table 2: Pipelines results on the test set

We carried out experiments with a maximum of `30 epochs`, employing early stopping monitoring the validation loss and a fixed batch size of `256`. Our chosen optimizer for all the experiments is `Adam` with `weight decay = 5e-5`, paired with the following learning rate schedulers:

- Polynomial Learning Rate with `power = 2`
- Step Learning Rate with `gamma = 0.5`
- Cosine Annealing Learning Rate with `eta min = [1e-6, 1e-7, 1e-8]`
- One Cycle Learning Rate with `steps per epoch = #batches` and `max learning rate = 1e-3`

All our experiments have been carried out on Google Colab, using `pandas` and `matplotlib` for data visualization, `pytorch`, `torchtext` and `pytorch_geometric` to build our models as well as `huggingface` to load the pretrained BERT model to fine-tune.

6 Discussion

BERT is pre-trained on a large corpus and it is established that it performs well on various NLP tasks, as it pushed the score on GLUE benchmark [1] to 80.5%. Fine-tuning BERT capitalizes on its robust pre-trained weights, yielding an accuracy of 84.29% on our classification tasks. However, the GloVe + LSTM model experiences consistent performance drops w.r.t. the other models, due to limited data for embeddings (GloveTweet is obtained from a corpus that compared with the one used for training BERT is smaller) and the absence of an attention mechanism to focus on specific tokens within the entire context. As a matter of fact, Although LSTMs are suitable for sequence modeling, they risk losing crucial contextual information. To address this limitation, we introduced GAT layers. These layers aim to enhance contextual understanding by smoothing each token's LSTM embedding based on its neighbors' embeddings, resulting in a slight performance increase from 66.36% to 68.74%. This technology is conceptually different to the standard attention mechanism employed in BERT, as the latter reweights the whole sequence bi-directionally. The poor performances of the GCNs are justified by the fact that we are averaging the embedding of a token with that of its neighbors without learning how to weigh the syntactic connections. This over-smoothing makes the embeddings too similar to one another and prevents the signal needed for classification from being propagated. Despite our attempts to combine BERT, LSTM, and GAT layers, the resulting model, BERT + LSTM + GAT, suffered from strong over-parametrization and failed to generalize due to overfitting in the initial epochs. The most confused classes are reported below, with an example for each case:

- Negative predicted as Extremely Negative
  Sample:
  `'just visited aldi at hawthorn adelaide and all meats, canned food among other things gone. panic buying due to coronavirus is insane what happens to those who are not panic buying and have a family to feed and keep healthy'.`

- Positive predicted as Negative
  Sample:
  `'thread interested to see what impacts covid has on a number of things as a result of working at home, self isolation. e.g. how will online behaviours change`

and stick online shopping, delivery services, media consumption'.

- Positive predicted as Neutral
  Sample:
  'petrol diesel prices static for 3rd successive day amid covid nationwide lockdown key things to know'.

- Negative predicted as Neutral
  Sample:
  'With new state guidelines on how grocery store employees can interact with the public. Stop yelling at workers because deli items are grab and go you rant over. Coronavirus bekind'.

- Negative predicted as Positive
  Sample:
  'food banks are really struggling, and they're the last line of defence for many of the most disadvantaged in our communities. please consider donating next time you're in a supermarket. covid_foodshortages'.

- Extremely Positive predicted as Positive
  Sample: 'the german chancellor nailed it with this line those who sit at supermarket cash registers or restock shelves are doing one of the hardest jobs there is right now. Be kind and show courtesy and respect covid on covid covidcanada covid'.

From these samples, it's evident that the model struggles to discern subtle differences necessary for distinguishing between close sentiments. Specifically, the network tends to avoid extreme classes and often misclassifies both positive and negative sentiments as neutral, despite the lower representation of neutral samples in the training set compared to positive and negative ones. This issue may stem from the use of LSTMs, as they require several consecutive tokens associated with a class to ensure robust signal propagation.

7 Conclusion

In this work, we proposed three solutions for a Kaggle challenge, with the aim of exploring the possibilities of what is stated in the work of Wang and Li [9]. These solutions are: a GloVe + LSTM baseline achieving 66.36% accuracy, a model based on [9] employing GloVe + LSTM + GAT achieving 68.74% and a fine-tuned BERT achieving 84.29%. The use of GAT slightly improved performances over our baseline, but it is significantly outperformed by BERT, as the latter model benefits from pre-training on a large corpus and on an attention mechanism that is better suited for the task. Indeed, Graph Attention is a sophisticated tool that allows for the modeling of complex structures and the injection of external knowledge. However, our experiments proved that for tasks that require the analysis of relatively short sentences, Graph Attention can cause over-smoothing of the embeddings, resulting in predictions that are less polarized. The use of an external tool to compute the dependency graph is limiting as well, as one powerful application of GNNs is to automatically learn the graph structure that better represent the sentence. Future work could be aimed toward the exploration of ways to automatically learn these graph structures for the purpose of classification.

8 Links to external resources

- The source code of all our experiments can be found here:
  https://github.com/GiovanniPioDelvecchio/GCNs_on_text

- The dataset and the Kaggle challenge can be found here:
  https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification?resource=download

References

[1] Afra Alishahi, Grzegorz Chrupała, and Tal Linzen. 2019. Analyzing and interpreting neural networks for nlp: A report on the first blackboxnlp workshop. *Natural Language Engineering*, 25(4):543–557.

[2] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[4] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. *arXiv preprint arXiv:1910.02356*.

[5] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[7] Ralf C Staudemeyer and Eric Rothstein Morris. 2019. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[9] Haitao Wang and Fangbing Li. 2022. A text classification method based on lstm and graph attention network. *Connection Science*, 34(1):2466–2480.

[10] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

[11] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

[12] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR.