CS397 Final Paper

Northwestern University CS397 Project Group 8

Abstract

Extracting semantic information using graph networks and syntactical information from dependency parsing have both been shown to be effective in natural language processing. In this work, we combine the two and build an Edge-Conditioned Graph Convolutional Network that takes in dependency parse trees of sentences to perform sentiment analysis.

1 Introduction

The incorporation of syntax and semantic information in large pre-trained BERT-like models has been a driving force for modern advancements in natural language processing. More recently, works such as (Wu et al., 2021) have suggested the incorporation of graph networks to capture semantic information on top of what pre-trained language models have to offer. In addition, the applications of fundamental theories in linguistics, such as dependency parsing have been explored in works like (Xu and Yang, 2019). In this experiment, we aim to improve the integration of syntax and semantic information by combining pre-trained embeddings with dependency parse trees using edge-conditioned Graph Convolutional Networks.

2 Related Work

2.1 CNN

Past literature have explored and shown the effectiveness of utilizing CNNs for NLP tasks such as sentiment analysis (Kim, 2014). The earliest methods represented words using pre-trained word2vec and later with contextual embedding such as BERT. These experiments used both fixed and trainable embeddings, as well as different sizes of convolutional filters. (Safaya et al., 2020) selected the output of the last four hidden layers of the 12-layer BERT as the input for CNN. They then used convolutional filters of sizes $\{1,2,3,4,5\}$, 32 filters of each size, and max-pooling to extract the feature.

2.2 GCN

(Yao et al., 2018) modeled a whole corpus as a heterogeneous graph, whose edges are built by word co-occurrence, word frequency, or word's document frequency. The embeddings of words and documents are updated and learned during training. (Peng et al., 2018) used pre-trained word2vec as input features and built a graph using word co-occurrence. Convolution masks are applied to the sub-graphs. (Lin et al., 2021) built the graph in a similar way as (Yao et al., 2018), but used BERT embedding to initialize. The node of documentation is updated iteratively using GCN, and the final representations of the document nodes are sent to the softmax classifier for classification.

3 Dataset

Our main task is to build language models for binary sentiment classification of texts in movie reviews. We trained on the IMDB Movie Review Dataset (Maas et al., 2011) for sentence classification. The full dataset contains 50,000 reviews from IMDB. The IMDB dataset only contains polarized reviews that score either lower than 4 (negative) or higher than 7 (really positive) on a scale of 10. As the movie reviews in the IMDB dataset have variable lengths, we padded each review to the maximum length (set to 256 in our experiment) so that it can be used as input to our models.

4 Model

4.1 Embedding

Our models and baselines use a smaller version of BERT, which is explored in (Turc et al., 2019). We are not experimenting with the more popular

BERT base and BERT large models because we are restricted to the computational resources we have. Specifically, our smaller version of BERT has 10 hidden layers (Transformer blocks), 512 hidden size, and 8 self-attention heads. This version of BERT only has around 40 million parameters, compared to 110 million parameters in BERT base and 340 million parameters in BERT large. The model input is preprocessed using a vocabulary for English extracted from Wikipedia and BooksCorpus. All text inputs are transformed into lower case before tokenization.

4.2 BERT + CNN

In this model, we use BERT to get contextual embedding of input tokens and use CNN as a classifier. Due to the limitation of our computers, we use a small BERT with embedding size 512 and 10 hidden layers. The biggest difference between this model and other models is that it uses not only the final output of BERT but also the outputs of hidden layers. After trying various combinations, we take the last 4 hidden layers of BERT as the input channels of CNN and use convolution filters of size [3,5,7] with 64 of each size. Then, it goes through a max pooling and concatenation. Finally, it goes through a dropout and dense layer.

4.3 BERT + Sentence-level GCN

We build the GCN for sentence-level as the whole dataset is a graph, in which a sentence is represented as a node and the similarity of nodes represents edges between them. The prediction is also in the form of a graph for the sentiment of all the sentences. In the model, the BERT embedding of the sentence is used as the node feature, and the cosine similarity of node features is used as the similarity measure to compare embeddings for building the edges, calculated by the NSL (Neural Structured Learning) module in TensorFlow with a similarity threshold to discard dissimilar edges. The NSL graph then is converted into a graph in Spectral (Grattarola and Alippi, 2020), which is an open-source Python library for building graph neural networks. The graph is then fed into the GCN model described in (Kipf and Welling, 2016). It uses 2 GCNConv layers and 64 hidden channels in the first GCNConv layer, the Relu activation function for the first GCNConv layer, and SoftMax activation function for the second GCNConv layer, with learning rate of 0.5, dropout rate of 0.2 and L2 regularization strength of 2.5e-4.

4.4 Dependency Parsing

Dependency parsing of a sentence contains information about its grammatical structure. Specifically, dependency parsing establishes relationships between the “head” words and the ”dependent” words that modify heads. To obtain dependency parse trees, we use a transition-based parser described in (Chen and Manning, 2014). The dependency relationships generated are universal dependencies, which is a framework for consistent annotation of grammar across languages. All dependencies in this experiment were generated using the package stanza (Qi et al., 2020). See Figure 1 for an example of the generated dependency parsing.

4.5 GCN with Dependency Parsing

Our word-based graph model uses words as nodes and dependency relationships as edges to represent sentences as graphs. We perform graph convolutions, which is introduced in (Kipf and Welling, 2016) on the graphs. The Graph Convolutional Networks, or GCNs, rely on message passing, which means nodes receive the information from their neighbors and perform a convolution to generate a new node feature. Using a sequence of message passing layers, information can be passed beyond the immediate neighborhood. Both the nodes and the edges have their own features, represented by X and E. The presence of a relation between nodes is represented by the adjacency matrix A. Each review is first passed to the stanza dependency relation parser to get the list of words and relations. The words are then processed by an embedding to generate the node features.

While we planned to generate node features using BERT embeddings, the total computational cost was much higher than expected. In addition, the syntactic information that is encoded by BERT is possibly redundant here as we are representing syntax using dependency parsing. We scoped down our experiment and used 16-dimensional custom embeddings instead of the 512-dimensional small BERT. The custom embeddings are from a model trained on the same IMDB reviews from the training set. The 51 different types of edge relations are represented by a one-hot embedding E. Alternatively, we could also have used the type of edge to inform the edge weights in the adjacency matrix.

For our experiment, we used one edge-conditioned graph convolution layer (ECC) fol-

I have been known to fall asleep during films, but this is usually due to a combination of things including, really tired, being warm and comfortable on the sette and having just eaten a lot. However on this occasion I fell asleep because the film was rubbish. The plot development was constant. Constantly slow and boring. Things seemed to happen, but with no explanation of what was causing them or why. I admit, I may have missed part of the film, but I watched the majority of it and everything just seemed to happen of its own accord without any real concern for anything else.I can't recommend this film at all.

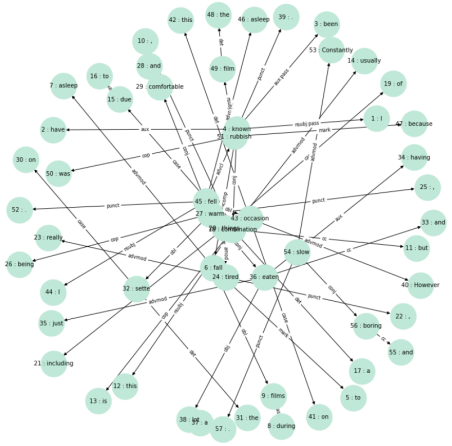

Figure 1: A generated dependency parsing graph using the movie review on the left.

lowed by an average pooling, a dense, a dropout, and the output layer (Simonovsky and Komodakis, 2017). More graph convolution layers may be able to improve performance however training costs become increasingly expensive. Note that we utilized Spektral's implementation of ECC, which does not contain the batch normalization layer. Our model did not include the graph coarsening layers in (Simonovsky and Komodakis, 2017).

4.6 Baseline Models

A simple baseline we have built takes the pooled output of our smaller BERT and feed it into a feed forward neural network. The feed forward model is consisted of a 16 unit dense layer, a dropout layer with 0.3 dropout rate, and the output layer. For this model, we use the RMSprop optimizer described in (Hinton, 2012) and a binary cross entropy loss.

Another baseline we implemented takes the sequence output of the smaller BERT and uses it as input into a bidirectional LSTM layer (Hochreiter and Schmidhuber, 1997). The LSTM layer contains 64 units. We feed the output from the LSTM layer into a max pooling layer, a dropout layer with rate of 0.5, a feedforward layer with 32 units, another dropout layer with rate of 0.5, and the final output layer. We use the RMSprop optimizer and a binary cross entropy loss.

5 Results

5.1 Experiment

For this experiment, we trained and evaluate our models on the full IMDB movie review dataset. We used 20,000 reviews for training, 5,000 for validation, and 25,000 for testing. We trained baseline models, CNN, and GCN sentence-level models with the smaller BERT we mentioned earlier. We freeze the weights of the smaller BERT to reduce computation cost. For the GCN model with dependency parsing, we used 16-dimensional custom embeddings. The primary metric for our experiment is accuracy. The effectiveness of the accuracy metric will be dependent on the balance of labels in the dataset.

5.2 IMDB Summary

We found that there were equal amounts of positive and negative reviews in the dataset, so no additional imbalanced learning or evaluation techniques were utilized for the experiment. The most challenging reviews for the models to classify were long reviews containing both positive and negative comments on the movie.

5.3 Model Performance

Table 1 shows the accuracy of each model for binary sentiment classification on the IMDB test set. The word-level GCN with dependency parsing achieved the highest performance. In addition, the edge-conditioned GCN outperformed the standard GCN which does not take edge labels into account. Surprisingly, the sentence-level GCN did not converge and yielded very low accuracy values. We also found that our accuracy levels were still significantly lower than SOTA, this is likely because our pre-trained embeddings were much smaller than the BERT-base/BERT-large and the

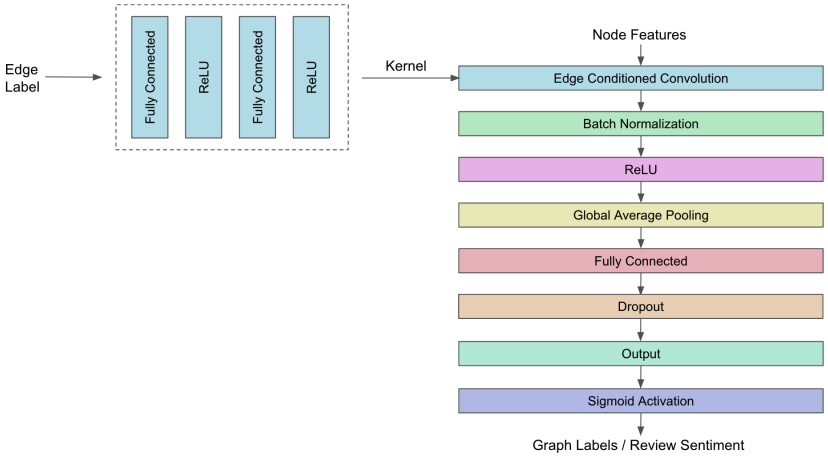

Figure 2: Model architecture of the edge-conditioned GCN with dependency parsing model.

Model	Train p.	Total p.	Acc.
Feed Forward	8k	47M	76.30%
BiLSTM	300k	47M	84.73%
CNN	7.9M	55M	84.50%
GCN sent.	33k	47M	65.68%
GCN dep.	561	500k	85.32%
GCN dep.(ECC)	20k	480k	87.36%

Table 1: Results for all models, Train p. stands for trainable parameters

600D word2vec used in previous works on IMDB sentiment analysis.

5.4 Edge Labels

We hypothesized that using one-hot encoded dependency relation as edge labels would improve the performance of the model but the results did not completely support this. While the ECC-GCN did outperform the standard GCN, we found that the masking of edge labels did not result in a hypothesized decrease in testing performance for edge-conditioned GCN. After masking out all edge labels, the test accuracy for ECC-GCN was 87.49%, slightly higher than the performance of the same model with edge labels 87.36%.

6 Discussion

Looking at the sentence-level GCN, the performance is the weakest among all models. This is likely due to the graph being constructed using cosine similarly and the over-smoothing of convolutional layers. Compared to the word-level models where the convolution is within each sentence, in sentence-level GCN the entire data set is feed into the model as a single graph with each sentence as a node. In order to capture the global state within this large volume, the convolution model needs to increase its depth. There are no edge types for a graph constructed using cosine similarity, creating neighborhoods with simple relation that makes it easy for the graph to be over-smoothed by when the neighborhood features are aggregating as a weighted sum in the GCN.

To tackle this issue, we could switch to Graph Attention Networks (GAT), as attention mechanism which could specify different types of labels and weights to edges between different neighbor nodes, thus could capture the relation between neighbors relatively without being smoothed by the aggregation. The Graph Star Net paper proposed using multi-head attention and a star transformer mechanism to capture local and long-range relationships without increasing the model depth, achieving 96.0% accuracy on the IMDB task (Haonan et al., 2019).

While the ECC-GCN model performed better than baseline, our analysis showed that edge features represented as one-hot encoded vectors did not contribute to this. Our proposed method of representing different dependency relation types was not sufficient for informing the model of review sentiment. There could be a variety of reasons for why this approach did not work for edge types. One possibility is that complexity of the kernel network and the amount of data was not sufficient for learning the representations of 51 different types of dependency relations. It could also be that the

task of sentiment analysis alone is not enough to train a model to learn the meaning of dependency relation labels. For future experiments, the edge types should be converted to vectors using either pre-trained embedding models or graph models which can co-embed both edges and nodes (Jiang et al., 2020).

One thing that could allow better understand how the graph models makes decisions is to use the GNNExplainer model, detailed in (Ying et al., 2019), to explain graph level predictions. The GNNExplainer model can produce a sub-graph and a subset of node features that are critical to the prediction. We could have used this model to understand what specific parts of a sentence is crucial to sentiment analysis. However, we did not get this to work as the GNNExplainer model does not work on edge-conditioned graphs, as it does not consider any edge feature information. Future work could try to modify the GNNExplainer model to work on edge-conditioned graphs.

7 Conclusion

In this work, we introduced a Graph Convolution Network that performs sentiment analysis on dependency parse trees of movie reviews. We also explored both a standard graph convolution layer and an edge-conditioned convolution layer that incorporates dependency relations as edge labels. The source code and notebooks can be found at https://github.com/xiaojoey/CS397Project.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Daniele Grattarola and Cesare Alippi. 2020. Graph neural networks in tensorflow and keras with spektral.

Lu Haonan, Seth H. Huang, Tian Ye, and Guo Xiuyan. 2019. Graph star net for generalized multi-task learning.

Hinton. 2012. Overview of mini-batch gradient descent. University Lecture.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Xiaodong Jiang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. 2020. Co-embedding of nodes and edges with graph neural networks.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1063–1072, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages.

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media.

Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021. Infusing finetuning with semantic dependencies. *Transactions of the Association for Computational Linguistics*, 9:226–242.

Yinchuan Xu and Junlin Yang. 2019. Look again at the syntax: Relational graph convolutional network for gendered ambiguous pronoun resolution.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks.