Pretrained Ensemble Learning for Fine-Grained Propaganda Detection

Ali Fadel, Ibraheem Tuffaha, and Mahmoud Al-Ayyoub

Jordan University of Science and Technology, Irbid, Jordan
{aliosm1997, bro.t.1996, malayyoub}@gmail.com

Abstract

In this paper, we describe our team's effort on the fine-grained propaganda detection on sentence level classification (SLC) task of NLP4IF 2019 workshop co-located with the EMNLP-IJCNLP 2019 conference. Our top performing system results come from applying ensemble average on three pretrained models to make their predictions. The first two models use the uncased and cased versions of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) while the third model uses Universal Sentence Encoder (USE) (Cer et al., 2018). Out of 26 participating teams, our system is ranked in the first place with 68.8312 F1-score on the development dataset and in the sixth place with 61.3870 F1-score on the testing dataset.

1 Introduction

Propaganda is an information, particularly of a misleading or biased nature, used to promote certain causes or views influencing specific audiences agenda using incorrect claims that might include emotional delusions.

Thus, propaganda detection problem is a real-life challenge that can affect how people understand news. Despite the uniqueness of the propaganda detection problem where the sentence can be affected by the context of the news articles and biased by external influences like the author writing style, the problem can still be considered as a binary sentiment analysis task (Medhat et al., 2014). Given a sequence of tokens representing a sentence from an article, tag it with one of two classes: 0 for non-propaganda or 1 for propaganda.

A new task has been proposed by the Propaganda Analysis Project[1] with a new manually annotated dataset at Natural Language Processing for Internet Freedom 2019 (NLP4IF 2019) workshop co-located with EMNLP-IJCNLP 2019 conference. For the full task and dataset descriptions, readers can refer to (Da San Martino et al., 2019b).

In this paper, we describe our team's effort to tackle this problem. Without any preprocessing steps, we build several models. The first two use Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) (uncased and cased versions) to extract word embeddings, then feed them to a Recurrent Neural Network (RNN) based on Bidirectional Long Short-Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) cells. The third one uses Universal Sentence Encoder (USE) (Cer et al., 2018) to extract sentence embeddings, then feeds them to a shallow Feed-Forward Neural Network (FFNN). After that, an average ensemble is used to merge the models predictions. Our system is ranked in the first place with 68.8312 F1-score on the development dataset and in the sixth place with 61.3870 F1-score on the testing dataset out of participating 26 teams. More insights about the teams results can be found in (Da San Martino et al., 2019a).

The rest of this paper is organized as follows. In Section 2, we describe our methodology, including the pretrained models used and our models structures, while, in Section 3, we present our experimental results and discuss some insights from our models in Section 4. Finally, the paper is concluded in Section 5.

2 Methodology

In this section, we present a detailed description of the extraction procedure for the word and sentence embeddings using both BERT and USE pretrained models. We then discuss the neural network models built on top of the extracted representations.

[1] https://propaganda.qcri.org/index.html

Proceedings of the 2nd Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda, pages 139–142

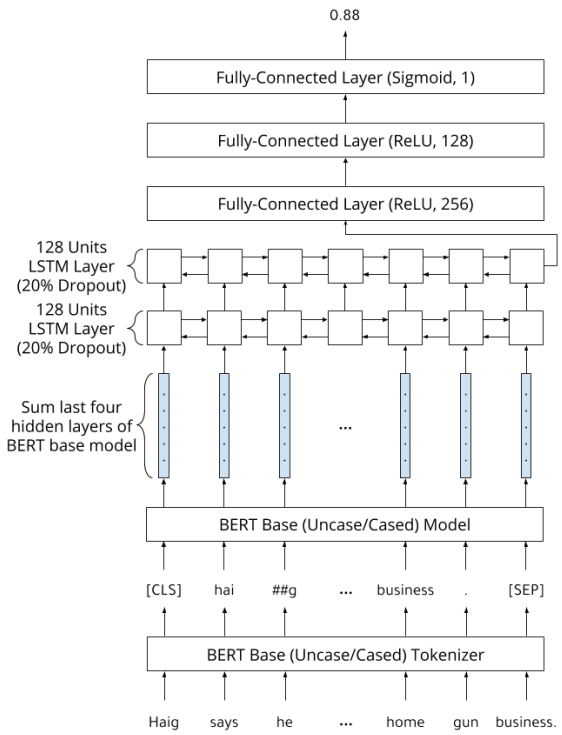

Figure 1: BERT-based models architecture

The implementation is available on a public repository.[2]

2.1 BERT-based Models

We use the small version of BERT (Base version) using each of the uncased and cased models provided by $pytorch - transformers$ Python package[3] to extract the word embeddings. The uncased and cased models are separately used to build two different models using the same RNN architecture. The usage of the cased version is to benefit from the cased words which mostly represent the named entities. As shown in Figure 1, the model can be divided into four layers/components:

1. Text Tokenization

   We use either the uncased or the cased BERT tokenizer (based on which model we want to train or inference) to tokenize text before feeding it to the BERT model. This step is important to run the BERT model and get the appropriate contextual words representations as the pretrained BERT model was trained on tokenized text (Devlin et al., 2018). The tokenizer applies several steps on the text to tokenize it. For example, it uses the WordPiece tokenizer (Wu et al., 2016) to segment the words into subwords.

2. Embeddings Extraction

   After the tokenization step, the tokenized text runs through the BERT model while saving the outputs of the hidden layers. The final embedding vector for each token is the summation of the last four hidden layers of the BERT model.

3. RNN Component

   The contextual embedding vectors extracted from the BERT model are fed to two consecutive BiLSTM layers (Hochreiter and Schmidhuber, 1997) with 128 hidden units, each with 20% dropout rate (Srivastava et al., 2014).

4. Shallow Feed-Forward Neural Network Component

   The thought vector (which is the final state outputted from the last step by the RNN cell) taken from the second BiLSTM layer is used as a representation vector for the input sentence. The vector is used as an input to two fully-connected layers that have 256 and 128 hidden units, respectively, with ReLU as their activation function. These layers are followed by an output layer with a Sigmoid activation function.

The uncased and cased models are trained for 4 and 5 epochs, respectively, on an Nvidia GeForce GTX 970M GPU in less than 20 minutes to train each model using Adam optimization algorithm (Kingma and Ba, 2014) with 0.001 learning rate, 128 batch size, and binary cross-entropy loss function. As for the inference time on the development dataset, which contains 2235 sentences, it is 3.5 minutes with an average around 10 sentences per second.

2.2 USE-based Model

Without any preprocessing steps, we use the Transformer (Vaswani et al., 2017) version of the Universal Sentence Encoder (Cer et al., 2018) model to encode the input sentences into fixed length vectors of size 512. These vectors are used as an input to two fully-connected layers with the same structure as the one used in the BERT shallow feed-forward neural network component.

[2] Link removed to maintain anonymity

[3] https://github.com/huggingface/pytorch-transformers

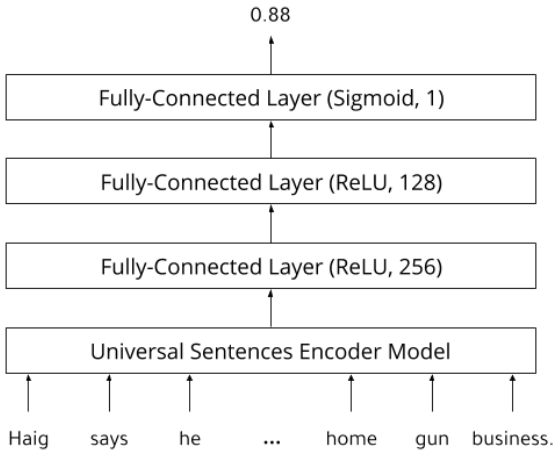

Figure 2: USE-based model architecture

This model is trained for 5 epochs on an Nvidia GeForce GTX 970M GPU in less than 5 minutes using Adam optimization algorithm (Kingma and Ba, 2014) with 0.001 learning rate, 32 batch size, and binary cross-entropy loss function. The inferencing time for this model is 30 seconds on the same development dataset with an average of 75 sentences per second.

3 Experimental Results

In this section, we present our experimental results by comparing our top performing system to several other attempts.

Our top performing system consists of three models. Two of these models are RNN models trained using contextual word embeddings extracted from BERT Base model using both uncased and cased versions. The uncased version achieves 66.1827 F1-score on the development dataset, while the cased version achieves 65.7849 F1-score on the development dataset. The ensemble average over these two models achieves 67.3279 F1-score on the same dataset. The third model, which is a shallow FFNN model that uses sentence embeddings extracted from Universal Sentences Encoder, achieves 63.7076 F1-score on the development dataset. Finally, the ensembling of the three models using average ensembing increases the results to 68.8312 F1-score on the development dataset, while the results decreased significantly on the testing dataset with 61.3870 F1-score. We adopted using 0.25 as our threshold for all experiments because using higher thresholds decreases the results significantly. For example, when using threshold 0.5 for the uncased BERT model, the results decreases to 58.1414 F1-score on the development dataset. Table 1 shows the models results on development and testing datasets.

Table 1: Models results on development and testing datasets

Model	Dataset	F1-score
Uncased BERT	Dev	66.1827
Cased BERT	Dev	65.7849
Uncased BERT + Cased BERT	Dev	67.3279
USE	Dev	63.7076
Uncased BERT + Cased BERT + USE	Dev	68.8312
Uncased BERT + Cased BERT + USE	Test	61.3870

Table 2: Uncased BERT model experiments results on development datasets

Model	F1-score
More Training	60.6282
3 Fully-Connected Layers	63.3349
3 BiLSTM Layers	65.5619
Duplicating Hidden Units	65.5355
Weighted Attention	65.9804

We reach the previously mentioned uncased model that achieves 66.1827 F1-score after conducting several experiments to explore the effect of applying different techniques on the network structure. The first experiment was to train the model for 10 epochs instead of 5, which yielded 60.6282 F1-score. Secondly, 3 fully-connected layers were used in training instead of 2. This reduced the result to 63.3349 F1-score. Similarly, an extra BiLSTM layer was added to the model, which decreased the result to 65.5619 F1-score. Then, we tried to duplicate the number of hidden units in each layers, yielding 65.5355 F1-score. Finally, we applied a sequence weighted attention (Felbo et al., 2017) on the outputs of the second BiLSTM layer. The output attention vector was used as a sentence representation instead of the thought vector, but the results did not improve giving 65.9804 F1-score. Table 2 shows the uncased BERT model experiments results on developments dataset.

4 Discussion

Although the USE model did not perform well compared to either the uncased or the cased BERT models (with 63.7076 F1-score compared to 66.1827 and 65.7849, respectively), adding the USE model to the ensemble average on top of both BERT models increases the results on the development dataset by around 1.5 F1-score. This indicates that the sentences representations from USE model could have captured unique information from the sentences which BERT models missed. Similarly, BERT cased performs worse than BERT uncased, but it increases its results by about 1.15 F1-score as it can differentiate the named entities which highly affects the semantic meanings.

It is worth noting that the results for all the teams significantly decreased in the testing dataset compared to their corresponding results on the development dataset. This is probably due to the fact that the testing dataset has a different distribution from the development dataset, which makes it harder to predict the outcome from such difference especially given a relatively small training dataset.

5 Conclusion

In this paper, we present our work on propaganda detection on sentence level classification, where we implemented three different models, two are based on BERT with uncased and cased versions and the last one uses USE. All these models build useful sentence representation which are used to make predictions. The ensemble average of these models achieved the first place with 68.8312 F1-score on the development dataset and in the sixth place with 61.3870 F1-score on the testing dataset out of 26 participating teams.

Acknowledgments

We gratefully acknowledge the support of the Deanship of Research at the *Jordan University of Science and Technology* for supporting this work via Grant #20180193.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Giovanni Da San Martino, Alberto Barron-Cedeno, and Preslav Nakov. 2019a. Findings of the nlp4if-2019 shared task on fine-grained propaganda detection. In *Proceedings of the 2nd Workshop on NLP for Internet Freedom (NLP4IF): Censorship, Disinformation, and Propaganda*, NLP4IFEMNLP '19, Hong Kong, China.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019b. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, EMNLP-IJCNLP 2019, Hong Kong, China.

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*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.