CYBER SAVER - A MACHINE LEARNING APPROACH TO DETECTION OF CYBER BULLYING

1Mr. R. AZHAGUSUNDARAM, [2]CH.CHARAN SAI, [3]BV.PRASHANTH
[4]B. HIMAM, [5]SK.NADEEM

Dept of CSE
BHARATH UNIVERSITY
CHENNAI, TAMIL NADU

***Abstract*—Nowadays, many people use their social media platform to spread hate online and that is why the problem of cyber-bullying detection has been the focus of many researchers over the past decade. In this work, we tackle this problem with transfer learning. We use various compact BERT models and fine-tune them with hate-speech data. We incorporate Focal Loss function to handle class imbalance in the data. Using this approach, we were able to achieve state-of-the-art results of 0.91 precision, 0.92 recall and 0.91 F1-score on the hate-speech dataset. Additionally, using our transfer learning pipeline, we show that the more compact BERT models are significantly faster in detection and are suitable for real-time applications of cyberbullying detection.**

***Key words*- Cyber-bulling, Hate-Speech, Compact BERT, Transfer Learning, Focal Loss**

I. INTRODUCTION

During the past decade with the rapid growth of social media interactions, many real-world issues mirrored themselves in the online world. Society has dealt with bullying and hate for a long time. However, those bullies can now easily hide behind a computer or smart-phone, using their social media platform to write offensive, abusive or hateful texts about somebody else or a group of people. This phenomenon called cyber-bullying, has affected people and caused depression in many children and adolescents. [1] It would be very beneficial if instances of cyber-bullying were detected as rapidly as they appear online. That is why there has been increased focus on cyber-bullying detection on different social media platforms in the past few years.

While dealing with this problem, one encounters many challenges and difficulties. Using informal language and emojis, different languages, lack of a good benchman the need for speed real-time detection in the streaming data [2] are just a few In this work, we focus on increasing the speed of cyber-bullying detection and demonstrate that smaller networks can perform just as well as bigger ones using transfer learning techniques.

For speed real-time detection in the streaming data [2] are just a few important challenges. In this work, we focus on increasing the speed of cyber-bullying detection and demonstrate that smaller networks can perform just as well as bigger ones using transfer learning techniques.

We contribute to this research field in two ways. Firstly, we fine-tune various compact BERT models [3] to increase the cyber-bullying detection speed and achieve state-of-theart performance. Secondly, we use the Focal Loss function in fine-tuning of these models and show how effective it can be in achieving even better results from BERT models.

II. RELATED WORK

Many researchers have tried to solve the problem of cyberbullying detection over the past decade. In the early works like [4] and [5], more conventional natural language processing ideas such as N-grams and TF-IDF were used to extract the features from text and then those features were used to train a type classifier such as SVM or Naive Bayes. In fact there are many interesting articles written that are covered in surveys such as [2].

Later on, deep learning methods gained more momentum and neural networks, including recurrent and convolutional neural networks (RNN and CNN), have played a major role in language modeling. As a result, in many approaches such as [6][7][8], different variations of LSTM and CNN models were developed to tackle the problem of cyber-bullying detection. These type of methods also incorporate word embedding layers such as word2vec[9] or GloVe[10], which are usually pre-trained on large set of words. These layers map a word into a high dimensional vector in a space where words with similar meaning are closer to each other.

ISBN 978-1-7281-8899-7/21/$31.00 ©2021IEEE
DOI 10.1109/ICOSEC51865.2021.9591742

Some approaches like [8] also include user metadata such as number of followers and their network of friends in their detection method. The researchers train a combined classifier which has a text path and metadata path.

In the past couple of years, there have been various competitions and challenges around this topic. In fact, several published articles are from the teams that participated in challenges such as SemEval2019[11]. In many articles like [12] and [13], you can recognize a shift towards using Transformer based architectures like BERT[14] in this area. In fact, among top 10 teams that participated in SemEval2019 offensive language detection, 7were using BERT based architectures.[11] BERT has Transformer layers that allow for a significant parallelization.[15] parallelization leads to more speed and that is a bonus. Also, BERT pre-trained models are powerful language representation models that can be easily fine-tuned and produce state-of-the-art results.[14]

III. PROPOSED METHOD

A. Data Distribution

For the purpose of this project, we used the Hate-speech data-set gathered in [16]. This data-set consists of 85948 tweets which are labelled using a crowd-sourcing mechanism. There are 3 target classes which are labelled as Normal, Abusive and Hateful. As can be seen in Figure 1, the majority of the data is not abusive nor hateful. Moreover, the data-set is imbalanced and the hateful class is small.

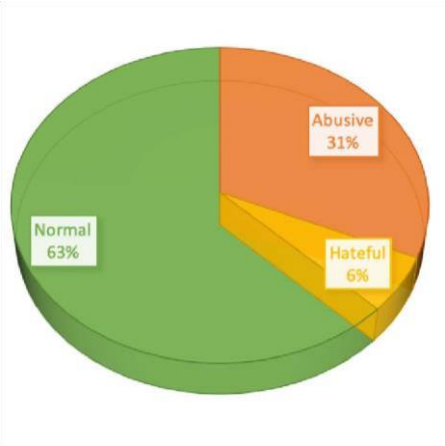


Fig. 1. Hate-Speech Data Distribution

B. Text Preprocessing

Twitters text usually includes emojis and hashtags as well as links to other pages. We first replaced the links with the word ”url” and replaced every @username with the word ”userid”. Following that, inspired by [12], we decided to use the valuable information hidden in hashtags and emojis.

As hashtags can be more than one word, or even a sentence, we used a open-source python library available on GitHub called Wordsegment [1] to split the hashtags into words. For example, a hashtag occurrence like ”#drawntodeath” will become ”drawn to death” after this segmentation step.

We used another open-source python library available on GitHub called Emoji[2] to convert each emoji instance to meaning behind that emoji. For example an emoji showing an angry face will be converted to ”:angry face:”. We also remove the ”:” from the two sides and add single space between chains of emojis.

Finally, we made sure to convert every uppercase letter into lowercase. This conversion is only necessary because the BERT models we fine-tuned are only trained using uncased text.

C. Compact BERT Models

Bidirectional Encoder Representations from Transformer or BERT [14] has proven to be a very powerful language model that can be used in many natural language problems, including sentiment analysis and text classification with fine-tuning. However, there is a downside to the original BERT and it has to do with its size. That is, BERT is a very large network and has so many transformer layers and hidden embeddings. Hence, fine-tuning with smaller data-sets would not lead to the best results.

Very recently, compact BERT models were introduced in [3] that address this issue. Researchers developed a total of 24 compact BERT models, varying in number of transformer layers and hidden embedding sizes. Each of these networks were trained with a teacher network that was essentially a very large BERT pre-trained model. They used unlabelled data and distillation method so that the student network could learn from the soft labels the teacher produces.

In this work, we selected 5 of these 24 compact BERT models to experiment with. As can be seen in Table I, the selected architectures are quite variant, with number of transformer layers ranging from 2 to 12 and hidden embedding sizes ranging from 128 to 768.

TABLE I
COMPACT BERT ARCHITECTURES THAT WERE INTRODUCED IN [3]

Model Name	Transformer Layers	Hidden Embedding Sizes
BERT-Base	12	768
BERT-Medium	8	512
BERT-Small	4	512
BERT-Mini	4	256
BERT-Tiny	2	128

D. Detection Pipeline

We use the compact BERT models in a pipeline shown in Figure 2 to classify the processed data into the 3 classes of normal, abusive and hateful. First, the whole prepossessed data is loaded as batches of text and true labels. Text instances are

1 https://github.com/grantjenks/python-wordsegment 2https://github.com/carpedm20/emoji 200

padded if necessary to match the sequence length. Then, the text is tokenized with a pre-trained BERT tokenizer.

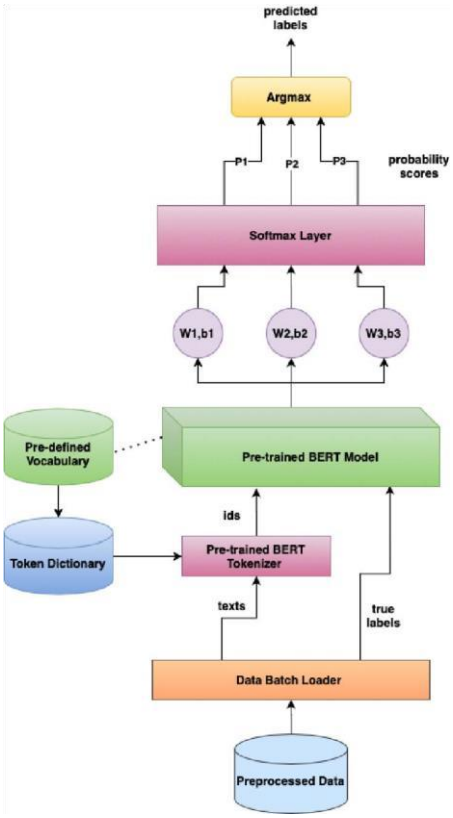
Each pre-trained BERT model comes with a corresponding pre-defined vocabulary set that then produces a token dictionary. The pre-trained BERT tokenizer uses this token dictionary to convert text as a sequence of words into sequence of identifiers which are numeric.

The final layer of the BERT model is removed and instead we include a dense layer with size 3, because we have 3 different classes. The dense layer is then followed by a softmax layer to produce probability scores for each class. The class with maximum probability score will result in the final predicted label.

E. Focal Loss

Inspired by the work of [17], we decided to use Focal Loss as our cost function. Focal Loss was first introduced in [18] as a variant of Cross-Entropy loss that also pays attention to how easy or hard it is to classify each sample. It has shown to be beneficial to applications with class imbalance problem [18]. The calculation of Focal Loss is shown in Equation 1, where *pt* = *p* if the sample is of positive class with true label

Fig. 2. Our Proposed Method Pipeline



$$FL(pt) = -\alpha t(1 - pt)^{\gamma} \log(pt) \quad (1)$$

It is very important to chose a right hyper-parameter γ for each application. In the next section, we will explain how this parameter was determined for our case.

IV. EXPERIMENTS AND RESULTS

A. Training Setup

To develop this project, we used Keras in the Google Colaboratory environment. We were able to use the TPU engines. For our optimization algorithm, we chose to use AdaBound [19] which was introduced very recently and can lead to more smooth training. All the hyper-parameters of our setup can be seen in Table II.

TABLE II TRAINING HYPER-PARAMETERS

Batch Size	128
Sequence Length	128
Number of Epochs	5
Learning Rate	0.0001
Focal Loss Parameter γ	0.1

B. FL Hyper-parameter Decision

To find the best value of γ for our usage of Focal Loss, we randomly split the data into 90% train and 10% validation. Then using Small-Bert and fixing every other hyper-parameter, we changed γ to different values to see which one would lead to better validation results. It is important to note that setting $\gamma = 0$ is equivalent to using the conventional Cross-Entropy loss.

TABLE III
IMPACT OF γ ON VALIDATION RESULT

γ	Accuracy	AUC	Precision	Recall	F1-score
0	0.9092	0.9702	0.9003	0.9092	0.9021
0.01	0.9138	0.9705	0.9062	0.9138	0.9077
0.1	0.9143	0.9709	0.9064	0.9143	0.9076
1	0.9125	0.9700	0.9029	0.9125	0.9033
2	0.9145	0.9683	0.9063	0.9145	0.9064
5	0.9113	0.9654	0.9026	0.9113	0.9026
10	0.9077	0.9643	0.8991	0.9076	0.8955

As can be seen in Table III, using values for γ which are too big resulted in worse evaluation metrics. This happens because by increasing γ we are reducing the weight of easy to classify samples more and more, which can damage the training process. It appears that the best result occurs for $\gamma = 0.1$, which is big enough to have a positive impact, but small enough to avoid ignoring easy to classify instances.

C. Evaluation Results

$y = 1$. Otherwise, $pt = 1 - p$. By this definition, a sample is easier

to classify if it has a smaller *pt*.

To evaluate the performance, we used 10-fold crossvalidation on the whole data-set, to be able to fairly compare our final results with previous research by Founta et al. [8]. We used 201Authorized licensed use limited to: University of Prince Edward Island. Downloaded on June 08,2021 at 13:46:19 UTC from IEEE Xplore. Restrictions apply.

various evaluation metrics to find the best model, with more emphasis on F1-score which is the harmonic mean Precision and Recall.

TABLE IV EVALUATION RESULTS

Model	Accuracy	AUC	Precision	Recall	F1-score
BERTBase	0.9156	0.9734	0.9090	0.9156	0.9103
BERT-Mediu	0.9140	0.9726	0.9071	0.9140	0.9084
BERTSmall	0.9147	0.9722	0.9080	0.9147	0.9093
BERTMini	0.9148	0.9717	0.9078	0.9148	0.9086
BERTTiny	0.9147	0.9699	0.9066	0.9147	0.9064
Founta et al. [8]	0.84	0.93	0.85	0.85	0.85

As can be seen in Table IV, our method is able to achieve better results than the previous work on the same data-set. The improvement is achieved in spite of the fact that our approach does not consider user-based and network-based metadata which is used by Founta et al.[8].

Moreover, it is interesting to see how close these compact BERT models are in evaluation metrics. BERT-Base has the highest F1-score and so if we are only considering the metrics, this is the winning model for our work.

However, we also considered the time it took to train and test each network. All models were trained and tested on Google Colaboratory TPU with 8 workers, and time was measured based on how long it took to process a batch of data, which was set to 128 for both train and test phases.

According to our time analysis in Table V, the models vary quite noticeably in their speed, with more variance in training times rather than test times. It is expected that adding more transformer layers and hidden embedding sizes would slow the networks down, but conventionally that meant also much better evaluation results. However, it is not the case here, as BERT-Tiny is the fastest with 6ms per step and yet only falls short by 0.04 percent in F1-score in comparison to BERTBase which takes 17ms per step. So, it is safe to say that in this case, the more compact networks have more to offer if they were to be employed in a system that needed real-time detection.

TABLE V
COMPACT BERT MODELS TIME ANALYSIS

Model	Training Time	Test Time
BERT-Base	136ms	17ms
BERT-Medium	65ms	10ms
BERT-Small	40ms	7ms
BERT-Mini	29ms	7ms
BERT-Tiny	19ms	6ms

V. CONCLUSION

In this work we presented a new method for cyber-bullying detection, which relied on the basis of transfer learning and fine-tuning compact BERT models. We achieved better results than previous work, without using any metadata. Moreover, We demonstrated that our method is both fast and reliable, which makes it very suitable for real-time detection of cyberbullying.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1813858. This research was also supported by a generous gift from the Herman P. & Sophia Taubman Foundation.

REFERENCES:

[1] M. P. Hamm, A. S. Newton, A. Chisholm, J. Shulhan, A. Milne, P. Sundar, H. Ennis, S. D. Scott, and L. Hartling, "Prevalence and effect of cyberbullying on children and young people: A scoping review of social media studies," *JAMA pediatrics*, vol. 169, no. 8, pp. 770–777, 2015.

[2] S. Salawu, Y. He, and J. Lumsden, "Approaches to automated detection of cyberbullying: A survey," *IEEE Transactions on Affective Computing*, 2017.

[3] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962v2*, 2019.

[4] M. Dadvar and F. De Jong, "Cyberbullying detection: a step toward a safer internet yard," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 121–126.

[5] A. Kontostathis, K. Reynolds, A. Garron, and L. Edwards, "Detecting cyberbullying: query terms and techniques," in

Proceedings of the 5th annual acm web science conference, 2013, pp. 195–204.
[6] P. Singh and S. Chand, "Pardeep at semeval-2019 task 6: Identifying and categorizing offensive language in social media using deep learning," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 727–734.
[7] V. Golem, M. Karan, and J. Snajder, "Combining shallow and deepˇ learning for aggressive text detection," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, 2018, pp. 188–198.
[8] A. M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis, "A unified deep learning architecture for abuse detection," in *Proceedings of the 10th ACM Conference on Web Science*, 2019, pp. 105– 114.
[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[10] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
[11] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)," *arXiv preprint arXiv:1903.08983*, 2019.
[12] P. Liu, W. Li, and L. Zou, "Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 87–91.
[13] P. Aggarwal, T. Horsmann, M. Wojatzki, and T. Zesch, "Ltl-ude at semeval-2019 task 6: Bert and two-vote classification for categorizing offensiveness," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 678–682.
[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[16] A.-M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, "Large scale crowdsourcing and characterization of twitter abusive behavior," in *11th International Conference on Web and Social Media, ICWSM 2018*. AAAI Press, 2018.
[17] S. Srivastava, P. Khurana, and V. Tewari, "Identifying aggression and toxicity in comments using capsule network," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, 2018, pp. 98–105.
[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss´ for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
[19] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," *arXiv preprint arXiv:1902.09843*, 2019.