# Chrestotes@HASOC 2020: Bert Fine-tuning for the Identification of Hate Speech and Offensive Language in Indo-European Languages

Tochukwu Ezike[a], Manikandan Sivanesan[b]

[a] *Xend Tech, Nigeria*
[b] *RedHat, Canada*

**Abstract**
This article describes our team Chrestotes' approach to the solution submitted to HASOC 2020: Hate Speech and Offensive Content Identification in Indo-European Languages. We demonstrate an end to end solution to the fine-grained detection of hate speech in tweets. Our solution is focused on the English Task which has been split into two subtasks. Our model achieved macro-average f1-scores of 0.4969 and 0.2652 on the subtasks A and B respectively. This solution places us in the middle of the leaderboard for subtask A and first place for subtask B.

**Keywords 1**
Hate speech, Offensive language, Text Classification, Bert, Transformers

## 1. Introduction

The wide adoption of internet use today has seen a massive increase in the use of internet services like social media for communication. This mass use has led to an increasing number of Hate speech on the various social media platforms. A lot of research has gone into the automatic detection and flagging of these hateful and derogatory comments by most social media companies [1]. The timely and accurate detection of hate speech using well-defined algorithms will go a long way to clean up platforms used for continuous discourse and also prevent the mental and physical effects these derogatory comments have on its victims.

HASOC 2020 is aimed at building efficient and scalable artificially intelligent models for detecting Hate speech and offensive Language in multilingual settings without human assistance. The challenge is divided into 2 subtasks across 3 languages which are English, German, and Hindi. The dataset for the 3 languages was obtained entirely from Twitter [2]

We participated in subtasks A and B for the English language. We approached the task using the transformer-based model BERT [3] which has proven to be very effective in Natural Language Processing tasks across multiple languages. The provided English dataset is fine-tuned using a pre-trained BERT transformer model from the HuggingFace library [4]. Our approach placed us in the middle of the leaderboard in subtask A and on top of the leaderboard in subtask B.

*FIRE '20, Forum for Information Retrieval Evaluation, December 16–20, 2020, Hyderabad, India.*
EMAIL: eziket18@gmail.com (T. Ezike); msivanes@redhat.com (M. Sivanesan)

## 2. Methodology and Data

### 2.1. Data Description

The task in the English Language challenge is divided into subtasks A and B. Subtask A is a binary classification challenge where we are to categorize the sentences in the English dataset into Hate-Offensive (HOF) and Non-Hate-Offensive (NOT) categories. In contrast, the subtask B is a fine-grained multiclass classification problem where we group the dataset into 3 categories; Hate speech (HATE), Offensive (OFFN), Profane (PRFN), and Neutral (NONE). For the training dataset in the English category, there are a total of 3708 tweets in the corpus. Further details about each task can be found on the competition homepage [2].

In total, there are 3708 tweets in the dataset. The two categories in subtask A have an almost equal number of approximately 185 tweets respectively. The categories in the subtask B however are highly imbalanced. The dataset distribution in both tasks can be seen in Figure 1 and Figure 2.

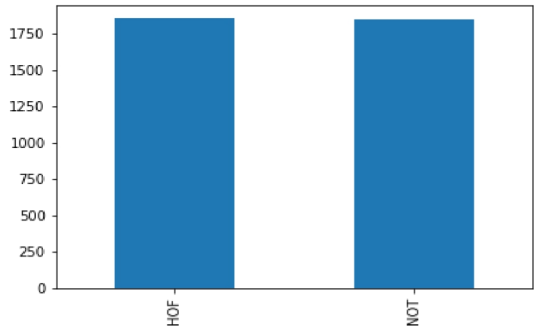


**Fig. 1.** Dataset Distribution for Subtask A (English)

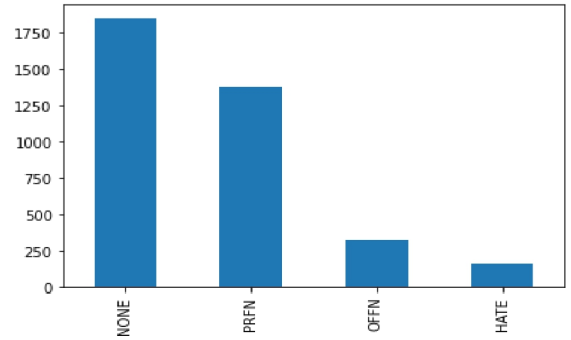


**Fig. 2.** Dataset Distribution for Subtask B (English)

### 2.2. Model Description

We used a pre-trained transformer-based Bert Model from the HuggingFace library. A concise architecture diagram of this model can be seen in Figure 3.

Based on the experiments performed, the pre-trained transformer-based Bert model provided better scores than other language models such as ULMFiT [11]. Bert-based models are currently state of the art in various NLP tasks including text classification. This is because transformers have higher expressive powers and are better universal approximators for sequential functions [9] than traditional methods. The pre-trained Bert model was trained on the large English Wikipedia corpus in an unsupervised manner. This allowed the model to learn contextual embedding representations of the words in the corpus. These learned contextual representations are the key reason Bert based models perform a lot better than context-free representations like word2vec or Glove[3]. We leveraged these learned contextual representations obtained through pre-training to initialize our weight. These weights were then further fine-tuned on subtasks A and B. It has been shown that this weight initialization method called Transfer Learning performs better and faster than random or universal initialization of the model's weights after training on the downstream task [10].

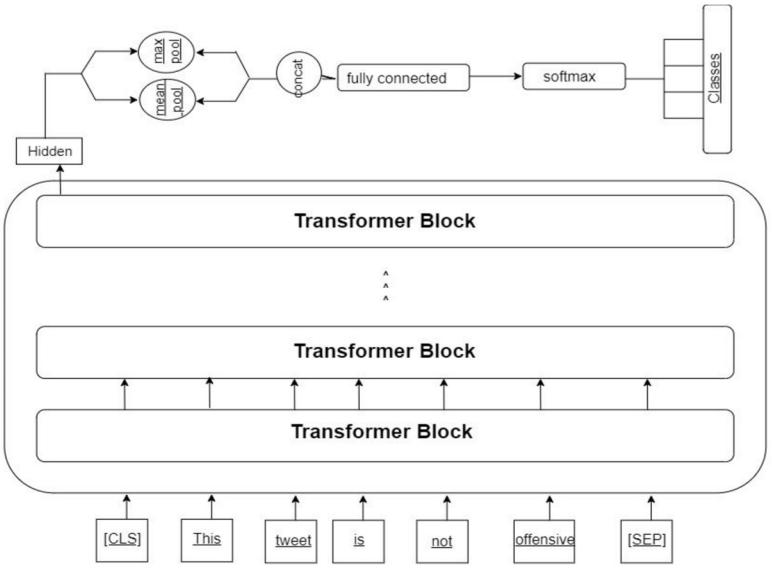


**Fig. 3.** Visual Architecture for Bert showing transformer layers for masked sentence prediction

The output sequences from the Bert model were pooled together to create a single representation using both a mean and max pool. This was done to ensure that strong representations were learned from the output sequences. These 2 pooled outputs were then concatenated and passed into an appropriately sized linear classifier for classification. This concatenation of pooled outputs gave a boost in the overall F1 score of the final model.

### 2.3. Ensembling

Our approach uses stratified K-fold across 5 folds as a cross-validation strategy due to the heavy imbalance amongst the classes in the dataset. This stratification was done to ensure that each fold had an equal number of samples. The intuition behind using this strategy is to randomly divide the dataset into 5 equal subsets and then use 4 out of these 5 subsets as training and the remaining subset as the hold out/validation dataset. This process is to be done 5 times across all the folds. This strategy is very useful as it helps the model to generalize better and not overfit to a single train-validation fold split. A diagram showing this ensembling approach can be seen in Figure 4.

Finally, after making predictions on our given test set using the models trained on each of the 5 folds, a simple mean across all the test set predictions from the 5 models is taken. This became our final prediction on the test set. The same validation strategy was used for subtasks A and B respectively.

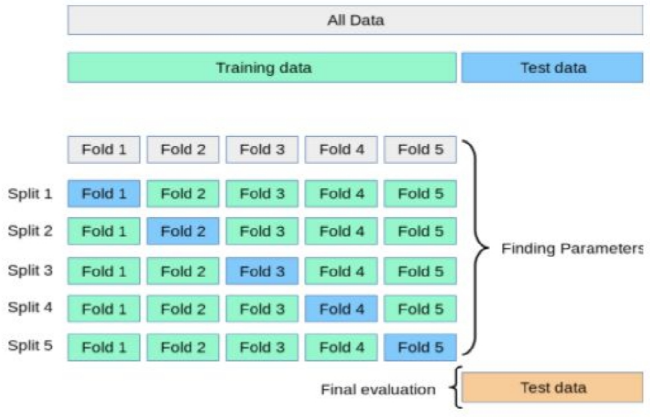


**Fig. 4**. Cross validation strategy using stratified K-fold across 5 folds.

## 3. Experiments and Results

### 3.1. Training

We used the "bert-based-uncased" model from the HuggingFace library and its corresponding tokenizer to maintain consistency with what was used during the model pre-training. All our fine-tuned models in the 2 subtasks were trained using Adam optimizer with weight decay [5]. The maximum sequence length in each batch sent into the model was set to 72. Categorical cross-entropy was used as a loss function in subtask A while Label smoothing cross-entropy was used for the subtask B. The loss functions were chosen based on the local cross-validation scores on the 2 tasks after experimentation. Maximum learning rates of 1e-5 and 1e-4 were used for the individual subtasks respectively. The learning rates were warmed up from 0 to their maximum values and then further decayed from this set maximum using a linear schedule.

Due to the high imbalance in the categories in subtask B, we used an upsampling strategy where the minority classes were duplicated until they matched the number of the majority classes. This was done with the BalanceClassSampler from the Catalyst library [8].

Lastly, a seed of 42 was set for reproducibility. All experiments were carried out with Pytorch [6] on Google Colab using Nbdev [7] to enforce an interactive development.

### 3.2. Result

**Table 1.** Variation in model performances on local validation set and test sets for subtasks A and B

| Experiment | Model | Subtask | Local CV macro f1-score | TEST CV macro f1-score |
|---|---|---|---|---|
| exp1 | ULMFiT | A | 0.49 | |
| | | B | 0.25 | |
| exp2.1 | Bert + mean pool | A | 0.522 | |
| | | B | 0.283 | |
| exp2.2 | Bert + concat pool | A | 0.532 | 0.4969 |
| | | B | 0.302 | 0.2652 |

According to the competition rules, the models for the 2 subtasks were evaluated on the validation set using the macro-f1 score. This exact metric was also used on the test set by the competition organizers to evaluate our models in order to rank them on the leaderboard. Experiments were conducted using Bert and ULMFiT. From Table 1, exp1 shows the results obtained with ULMFiT in subtasks A and B respectively. exp2.1 shows results from the Bert model using only a mean pool while exp2.2 shows results from the Bert model using a concatenation of a mean and max pool along the output sequences.

**Table 2.** Performance of final models on the leaderboard for subtasks 1 and 2

| Subtask | Model Name | LB macro f1-score | Highest LB macro f1-score | Ranking |
|---|---|---|---|---|
| A | Bert + concat pool | 0.4969 | 0.5152 | 24th |
| B | Bert + concat pool | 0.2652 | 0.2652 | 1st |

## 4. Conclusion

In this paper, we presented our solutions to the HASOC 2020 competition which was placed at the middle and at the top of the leaderboard for the English subtasks A and B respectively. Our solution involves fine-tuning a Bert model on the training dataset. We present a robust end-to-end pipeline for reproducing our solution effectively. All our code has been open-sourced in this repository https://github.com/tezike/Hasoc.

## References


[1] E. Whittaker, R.M. Kowalski. 2015. Cyberbullying via social media. Journal of School Violence, 14(1):11C29

[2] T. Mandl, S. Modha, G. K. Shahi, A. K. Jaiswal, D. Nandini, D. Patel, P. Majumder, J. Schäfer, Overview of the HASOC track at FIRE 2020: Hate Speech and Offensive Content Identification in Indo-European Languages), in: Working Notes of FIRE 2020 - Forum for Information Retrieval Evaluation, CEUR, 2020.

[3] J. Devlin, M.W. Chang, K. Lee, K. Toutanova: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423

[4] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771. G oogle Scholar

[5] I. Loshchilov, F. Hutter. "Decoupled weight decay regularization. arXiv 2017." arXiv preprint arXiv:1711.05101.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, A. Desmaison (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems (pp. 8026-8037

[7] J. Howard, S. Gugger (2019, December 6). n bdev: use Jupyter Notebooks for everything . https://www.fast.ai/2019/12/02/nbdev/

[8] S. Kolesnikov, (2018). Catalyst: Accelerated deep learning R&D. https://github.com/catalyst-team/catalyst.

[9] C. Yun, S. Bhojanapalli, A. Rawat, S. Reddi, S. Kumar (2020). Are Transformers universal approximators of sequence-to-sequence functions? ArXiv, abs/1912.10077.

[10] Tan, Chuanqi, et al. "A Survey on Deep Transfer Learning." ArXiv:1808.01974 [Cs, Stat], Aug. 2018. arXiv.org, h ttp://arxiv.org/abs/1808.01974

[11] J. Howard, S. Ruder. "Universal Language Model Fine-Tuning for Text Classification." ArXiv:1801.06146 [Cs, Stat], May 2018. arXiv.org, http://arxiv.org/abs/1801.06146.