HateDetectors at HASOC 2020: Hate Speech Detection using Classical Machine learning and Transfer learning based approaches

Varsha Reddy[a], Surendra Telidevara[b]

Abstract

In this paper, we describe our models submitted to the HASOC shared task conducted as part of FIRE 2020. We have presented two directions to approach the problem of hate speech detection which are based on the advancements in the field of natural language processing. We present classical machine learning approaches using Support Vector Machine (SVM) and transfer learning approaches at sentence level using BERT. We have shown through experimental results that Transfer learning based approaches beat Machine learning based approaches in identifying the nuances of hate speech in a sentence. We have performed experiments on the English and Hindi datasets. On the public test dataset provided, we obtain a macro F1 score of 0.90 and 0.67 on English and Hindi languages respectively for subtask A and scores of 0.54 and 0.44 for subtask B. We also observe a difference of around 0.05 macro F1 score between BERT based models and SVM based models on English public test data, while we see only a difference of 0.01 on Hindi public test data in subtask A. We have highlighted the importance of a monolingual model over a multi lingual BERT based model for hate speech detection. We also highlight the importance of having a large, balanced training dataset on model performance for hate speech detection.

Keywords

Hate Speech Detection, Transfer Learning, Machine Learning, SVM, BERT, CEUR-WS

1. Introduction

Social media has become a great platform for communication amongst people living distantly. It is also a platform for expressing one's ideas and thoughts. The number of users on different social media platforms is increasing day by day and hence they have a wide reach. While social media when used constructively is a wonderful tool, the cases of cyberbullying, harassment, targeted spread of hate and so on, have increased a lot.[1,2] Online hate can not only affect the mental health of a person, but can also translate to violence in the real world as well and hence this issue requires attention.[3,4]

FIRE '20, Forum for Information Retrieval Evaluation, December 16–20, 2020, Hyderabad, India

varsha.redla@students.iiit.ac.in (V. Reddy); surendra.telidevara@students.iiit.ac.in (S. Telidevara)

[1]https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons

[2]https://www.hindustantimes.com/analysis/it-is-time-to-regulate-hate-speech-on-social-media/story-x2JfnAcZ4mh404CM2wQLpO.html

[3]https://timesofindia.indiatimes.com/home/science/Hate-posts-on-social-media-can-affect-mental-health/articleshow/41570101.cms

[4]https://www.washingtonpost.com/nation/2018/11/30/how-online-hate-speech-is-fueling-real-life-violence/

Hate speech detection has gained a lot of importance in the NLP community. The HASOC 2020 [1] shared task which is part of FIRE 2020 is an attempt in this direction. It provides a forum for classifying text into hate speech. It has two subtasks, A and B, in three languages: English, Hindi and German. subtask-A is a binary classification task in which we are required to classify the post into hate speech (HOF) and non hate speech (NOT). subtask-B tries to further address the fine tuned versions of hate speech: Hate speech (HOF), Profanity (PRF) and Offensive speech(OFF). The posts which have been classified as HOF in the previous subtask have to be further classified into one of these 3 categories.
We have conducted our investigation on the English and Hindi datasets. Simple word based filtering approaches to tackle this problem are not efficient enough as a sentence could only be tagged as hateful based on the context in which that word has been spoken. So, with this simple swear word-based filtering approach, we might end up falsely tagging a lot of sentences hateful.
The BERT model [2] which is a pre trained language model based on the neural transformer networks [3], has achieved state of art results on most of the NLP tasks. There are two main pre trained model versions of BERT available: Monolingual models which has been pre trained on a specific language and Multilingual model which has been trained on multiple languages and the hypothesis is that it could be used on any language. Hence we wanted to utulize these pretrained models on this task. We also wanted to experiment the effectiveness of the multilingual model as opposed to the monolingual version. Another aspect we wanted to see was how well these pretrained transformer models perform when compared to classical models like SVM, Logistic regression and so on. We have experimented with Support vector machines with multiple embedding schemes and compared the performance with the BERT based models.
The rest of the paper is organized as follows: Section 2 deals with the data description. In Section 3, we present our model and in Section 4, 5 we present our results and discussion. In section 6, we mention our final submitted models and finally in section 7 we conclude.

2. Dataset Description

HASOC 2020 has offered two subtasks:
Subtask-A: Classification of the post as hate speech (HOF) and non hate speech (NOT). All categories of hate like profanity, offensive language, hate speech are labelled as HOF as part of this task. This is a binary classification task. The English dataset is almost perfectly balanced while the Hindi dataset is imbalanced with only 25% of the data being HOF.
For English subtask-A, we have used an additional dataset named OLID [4], proposed by OffensEval [5]. This is a dataset for hate speech detection. There are 4400 posts under the label of OFF denoting offensive and 8840 posts categorized as NOT denoting non-hateful posts.
Subtask-B: The posts with 'HOF' label have to be further classified into one of the fine tuned labels: HATE, OFFN, PRFN denoting hate speech, offensive content and profane content respectively. The posts with 'NOT' from the previous subtask-A are labelled as NONE in subtask-B. Here,unlike the case in subtask-A, both English and Hindi datasets were highly imbalanced, with the majority class being NONE in both cases. Additionally in the case of English langauge, PRFN class formed the majority among HATE, PRFN and OFFN classes.

Table 1
HASOC 2020 Dataset description

subtask-A (Binary Classification)		
Label	English Dataset	Hindi Dataset
HOF NOT	1856 1852	847 2116
subtask-B (Multi Class Classification)		
Label	English Dataset	Hindi Dataset
HATE NONE OFFN PRFN	158 1852 321 1377	234 2116 465 148

The table 1 has the number of posts per label for English and Hindi datasets.

3. Methodology and experiments

This section is divided as follows: We first present a common pre-processing pipeline which we follow for both subtasks A and B. We then present our method description for subtask-A and then describe our methodology for subtask-B.

3.1. Pre-processing

The dataset has been extracted from social media sites like twitter. So, it is expected that it would have a lot of social media jargon and hence contains unconventional social media style of writing. It is essential to pre-process the dataset before passing it on to the models. The following is the pre-processing pipeline we followed for English:
1. URL Removal: All the urls have been replaced with a special token.
2. User Mention Removal: All the user mentions have been replaced with another special token.
3. Conversion of emojis and emoticons into corresponding text.
4. Expansion of hashtags, removal of stop words and unnecessary symbol, case folding.
5. Map all elongations to the same word. (For example "yeah", "yeahhhh", "yeeaahhh" into "yeah")
For Hindi we additionally stem the words and we did not perform step-5. All other steps were performed. We have used the NLTK [6] library in python to perform some of the above tasks.

3.2. Subtask-A

We have experimented with two kinds of approaches:
1. Classical Machine Learning Approaches using SVM
2. Transfer Learning Approaches using Neural BERT

3.2.1. SVM Based Model

We perform all further processing on the pre-processed data. There are two steps in this:

1. **Sentence Encoding**: The resultant text after pre-processesing has to be converted into vector form for the classification models to train upon. The output of this step is a vector which tries to represent the sentence. For this, we had experimented with different sentence encoding mechanisms:
   a) Universal Sentence Encoder [7] (Used only for English dataset) We will use the abbreviation *USE* to denote Universal Sentence Encoder embedding scheme. We used the model available on TF Hub for Universal Sentence Encoder.[5]
   b) Averaging the word level fasttext [8] embeddings in the sentence. We will call this encoding scheme *Avg Fasttext* in this paper.[6] [7]
   c) TfIdf Vectorization. We will call this encoding scheme *TfIdf vect* in this paper.
2. **Classification**: We have experimented with SVM [9] classifier provided by scikit-learn [10] library in python. For the C value in parameter, we performed grid search on values from 0.0001 to 100, with a jump per interval of 10. We have used 5-fold cross validation to come up with the best model for the task. The kernel used was radial basis function ('rbf') [11]. We have experimented with two modes of loss calculation. First mode is the standard loss calculation. Second mode is giving more weightage to class with lesser number of samples in the loss. The weightage given is inversely proportional to the number of samples in that class. We will call this mode SVM_b in this paper.

While trying the above machine learning approaches on the Hindi dataset, we experimented on two forms of data:

1. Original devanagari form
2. Transliterated form: We performed transliteration on the Devanagari form to convert it to the standard Roman form. We have used mylanguages website [8] to perform the transliteration.

3.2.2. BERT Based Model

The pre-processing pipeline followed before performing the experiments mentioned in this section is same as that mentioned in Section 3.1. We describe our approach below:

1. **English**: For English, we have used the monolingual pre-trained cased BERT model [2] to get the sentence encoding. We will call this model $BERT_{mono}$ in this paper. The classifier used is a dense layer, with the output dimension as 1, representing the label. We used an additional dataset named OLID, to boost up the number of training samples.

[5]https://tfhub.dev/google/universal-sentence-encoder/1

[6]Here, embedding themselves might have been pre-trained using deep learning based approaches, but these have been reported under the encoding for the machine learning approaches as the title represents the nature of the classification algorithm used and not the embedding training approach.

[7]We are just using USE/Avg Fasttext as embedding schemes. We did not fine tune these architectures and hence we did not place these mechanisms in the Transfer Learning section.

[8]http://mylanguages.org/devanagari_romanization.php

We appended the OLID dataset to the 2020 English subtask-A dataset to obtain the final dataset. We took the pre-trained BERT model and fine tuned it with this final dataset.

2. **Hindi**: For Hindi, we have used the multilingual cased pre-trained BERT model [9] to get the sentence encoding. We will call this model $BERT_{multi}$ in this paper. The classifier again is a dense layer with the output dimension as 1, denoting the label itself. We took this pre-trained multilingual model and fine tuned it on the 2020 Hindi dataset for subtask-A.

3.3. Subtask-B

For subtask-B, the preprocessing pipeline is same as the one described in Section 3.1. The dataset for this subtask was pretty imbalanced. There were two strategies which we had tried, second one adding on to the first strategy. We follow the same approaches for English and Hindi. The sentence encoding schemes remain same as the ones mentioned in Section 3.2.1.

1. Approach 1: In this approach, we do not consider it as an extension to subtask-A. We simply take the encoded data and its corresponding label for subtask-B, for training, and fit an SVM on this data. This would translate to multiclass classification with 4 labels: NONE, HATE, PRFN and OFFN. We can see here that this classification is independent of the previous task's output.
2. Approach 2: In this approach, we first filter out all the non-NONE labelled data from the training data for subtask-B and fit an SVM on it. Henceforth, we will call this model *SVM_1* in the paper. Now we take the first subtask's output and directly label the 'NOT' predictions from the first task as 'NONE'. For the remaining posts, we predict using the *SVM_1* model.

4. Results

HASOC organizers have released public test data with actual ground truth labels this year. The final leaderboard ranks were decided based on a private test dataset. All the results reported in Tables 2-5 are on the 2020 public test data. We included macro precision, macro recall and macro F1 score in our results. Macro values were calculated by averaging the corresponding value for all the classes. Organizers of HASOC have used macro-F1 as the primary metric in deciding the leaderboard positions. Table 6 shows the performance of our model on the private test data. The results of Hindi subtask-B were not announced at the time of submission of this paper.
For English subtask-A, results labelled with OLID+2020 denote models trained on 2020 HASOC dataset appended with OLID dataset as mentioned in the methodology section. The results without that label are trained on the 2020 HASOC dataset only. The label *trans* in Hindi results denotes that the model has been trained on the transliterated version of the dataset. All the results for subtask-B are based on the two approaches described in section 3.3.

[9] https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip

Table 2
English Subtask-A results on public test data

Machine Learning Approaches			
Method	Macro Recall	Macro Precision	Macro F1 score
USE + SVM$_b$	0.86	0.86	0.86
USE + SVM	0.85	0.85	0.85
USE + SVM$_b$*(OLID+2020)*	0.86	0.86	0.86
USE + SVM *(OLID+2020)*	0.85	0.86	0.86
Avg Fasttext + SVM$_b$	0.87	0.87	0.86
Avg Fasttext + SVM	0.87	0.87	0.86
TfIdf Vect + SVM$_b$	0.86	0.85	0.85
TfIdf Vect + SVM	0.87	0.86	0.85
TfIdf Vect+SVM$_b$$(OLID + 2020)$	0.88	0.88	0.88
TfIdf Vect+SVM*(OLID+2020)*	0.87	0.86	0.85
Transfer Learning Approaches			
BERT$_{\mathbf{mono}}$*(OLID+2020)*	**0.87**	**0.93**	**0.90**
BERT$_{multi}$*(OLID+2020)*	0.87	0.88	0.88
BERT$_{mono}$	0.89	0.90	0.89
BERT$_{multi}$	0.88	0.84	0.88

Table 3
English Subtask-B results on public test data

Attempt-1 (Multi class Classification with NONE label)			
Method	Macro Recall	Macro Precision	Macro F1 score
TfIdf Vect + SVM$_b$	0.73	0.50	0.51
TfIdf Vect + SVM	0.61	0.47	0.45
USE + SVM$_\mathbf{b}$	**0.53**	**0.55**	**0.54**
USE + SVM	0.77	0.47	0.46
Avg Fasttext + SVM$_b$	0.54	0.58	0.54
Avg Fasttext + SVM	0.56	0.46	0.43
Attempt-2 (Filter NONE label)			
TfIdf Vect + SVM$_b$	0.58	0.51	0.51
TfIdf vect + SVM	0.48	0.45	0.42
USE + SVM$_b$	0.53	0.53	0.53
USE + SVM	0.74	0.48	0.48
Avg Fasttext + SVM$_b$	0.51	0.50	0.51
Avg Fasttext + SVM	0.52	0.47	0.44

5. Discussion

An important thing to notice in the case of BERT models is that when there is monolingual model available, it is outperforming the multilingual model. We saw a difference of around 0.012 macro F1 score between both the models. The multilingual model though turns out to be better than the classical SVM model. We saw a difference of around 0.02 macro F1 score again

Table 4
Hindi Subtask-A results on public test data

Method	Macro Recall	Macro Precision	Macro F1 score
Machine Learning Approaches			
Avg Fasttext + SVM_b	0.66	0.68	0.66
Avg Fasttext + SVM	0.81	0.56	0.53
TfIdf Vect + SVM_b	0.67	0.64	0.65
TfIdf Vect + SVM	0.73	0.54	0.50
Avg Fasttext + $\text{SVM}_b(trans)$	0.56	0.56	0.56
Avg Fasttext + SVM *(trans)*	0.35	0.50	0.41
TfIdf Vect + $\text{SVM}_b(trans)$	0.70	0.65	0.66
TfIdf Vect + SVM *(trans)*	0.77	0.52	0.46
Transfer Learning Methods			
$\text{BERT}_{\text{multi}}$	**0.68**	**0.67**	**0.67**

Table 5
Hindi Subtask-B results on public test data

Method	Macro Recall	Macro Precision	Macro F1 score
Attempt-1 (Multi class classification with NONE label))			
TfIdf Vect + SVM_b	0.57	0.36	0.39
TfIdf Vect + SVM	0.69	0.28	0.27
Avg Fasttext + SVM_b	0.39	0.38	0.39
Avg Fasttext + SVM	0.19	0.25	0.21
Attempt-2 (Filter NONE label)			
TfIdf Vect + SVM_b	**0.56**	**0.41**	**0.44**
TfIdf Vect + SVM	0.65	0.37	0.37
Avg Fasttext + SVM_b	0.41	0.46	0.42
Avg Fasttext + SVM	0.46	0.36	0.33

between the multilingual BERT model and the SVM model. This can be seen in Table 2. We can see from the results of subtask-A that BERT model has performed better than SVM model for both languages.
We could see significant improvement in macro F1 score between SVM_b model and the SVM model in cases where there was data imbalance. In subtask-B where the dataset is heavily imbalanced and is very small, we could see improvement upto 0.12 macro F1 score, which is quite significant. In subtask-A, there wasn't any significant improvement for English as the dataset was already balanced. There was an improvement upto 0.15 macro F1 score for Hindi subtask-A which is expected due to the imbalance in the dataset.
From Table 5 we can see that there is an improvement of 0.12 macro F1 score between SVM models in attempt-1 and attempt-2. We see that the second approach has indeed reduced the bias towards the majority class [NONE] and has improved the overall Macro F1 score. We could see a significant number of samples now getting classified into classes other than 'NONE'.
From experimental analysis, we could see that usage of additional data for English Subtask-A

Table 6
Performance on private test data

Subtask-A			
Language	Macro F1	Macro F1 of the best performing model	Leaderboard position
English	0.4981	0.5152	15th
Hindi	0.5129	0.5337	10th
Subtask-B			
English	0.2299	0.2652	17th
Hindi	0.2272	0.3345	13th

did help in improving the macro F1 score. As we can see in Table 2 there was an improvement of 0.01 macro F1 score in the case of BERT and 0.03 macro F1 score in the case of TfIdf Vect + SVM_b when we additionally use the OLID dataset.

6. Submitted models

We have submitted two runs for each subtask for English and Hindi. For English subtask-A, we submitted USE+SVM_b model and the $BERT_{mono}$ (OLID+2020) model described in the previous sections. For Hindi subtask-A, we submitted $BERT_{multi}$ based model and the *Avg Fasttext* model described above. For subtask-B, we submitted approaches 1 and 2 described in section 3.3 as part of the two runs for both the languages. For subtask-B, we submitted models with USE sentence encodings for English and for Hindi we submitted models with TfIdf sentence encoding scheme.

7. Conclusion

We have described two main directions for the hate speech detection problem: Classical machine learning approach and Transfer learning based approach. We have shown through experiments that BERT based models pre-trained on large English corpora and fine tuned on the dataset available yield good results. We have achieved a macro F1 of 0.90 and 0.67 for English and Hindi subtask A respectively which is 0.05 and 0.01 more than the best performing SVM counterpart. The proposed model performed reasonably well, it was off by very small margins compared to the best performing models in almost all cases. It was off by 0.01 and 0.02 F1 score for English and Hindi respectively in subtask A and 0.03 and 0.11 F1 score in subtask B. The poor performance in subtask-B could be attributed to the fact that the dataset was small and imbalanced. Furthermore, the classes individually being very fine grained makes it difficult for models to capture. We should try to build more robust models which capture these subtleties between classes and also possibly try to build a more balanced and larger dataset in order to improve the efficiency of the hate speech detection systems. Various approaches like LSTM, CNN, LSTM-Attention, etc can also be tried. Comparing the efficiency of these approaches with BERT and SVM based models is left to future work. We plan to experiment with BERT based architectures for subtask-B as part of future study. We hope our experimentation and analysis helps in tackling the problem of hate speech detection in some way.

References

[1] 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.

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017) 5998–6008.

[4] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting the type and target of offensive posts in social media, arXiv preprint arXiv:1902.09666 (2019).

[5] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval), arXiv preprint arXiv:1903.08983 (2019).

[6] E. Loper, S. Bird, Nltk: the natural language toolkit, arXiv preprint cs/0205028 (2002).

[7] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., Universal sentence encoder, arXiv preprint arXiv:1803.11175 (2018).

[8] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.

[9] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intelligent Systems and their Applications 13 (1998) 18–28.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[11] M. J. Orr, et al., Introduction to radial basis function networks, 1996.