Detect Hate and Offensive Content in English and Indo-Aryan Languages based on Transformer

Yongyi Kui

Information Institute of Yunnan University, Yunnan, China, 650504

Abstract

This paper describes my submission to the Subtask 1A and Subtask 1B tasks of the HASOC (2021) Hate Speech and Offensive Content Identification Challenge. In the experiment, I applied different pre-training and common neural network models for this task and integrated them. According to the official evaluation results, the test results of the solution proposed in this article are ranked fourteenth and fifteenth on English Subtask A and English Subtask B, the rankings on Hindi Subtask A and Hindi Subtask B are sixth and fifth, respectively, and Marathi Subtask A is ranked eleventh. The source code for the evaluated models in this paper is shared openly.

Keywords

Text Classification, Hate and Offensive Content Analysis, pre-trained model, Transformers

1. Introduction

In recent years, offensive language on social media platforms has surged. Because the Internet has a certain degree of anonymity, people are more likely to publish hate speech [1] on online platforms than in reality.

Hate speech will bring challenges to social civilization and harmony. Similarly, insulting offensive speech will lead to the radicalization of communication. Therefore, it is necessary to find an appropriate way to automatically recognize such content to enhance the public opinion environment of social media. Human beings are more sensitive to hate speech and offensive content, so people can easily identify such speech. However, the computer can only detect whether the text is hateful or offensive after learning via unsupervised, self-supervised, or supervised methods that are based on large amount of data.

In the challenges of HASOC 2019 [2] and HASOC 2020 [3], there is a task of identifying hate speech and offensive content in English and Hindi. In addition, in 2019, SemEval [4] has a task to identify the offensive and non-aggressiveness of English tweets. They use convolutional neural networks and the BERT model to solve them. SemEval proposed a task called OffensEval 2020 [5] in 2020, to identify offensive content in multiple languages, including English. In order to identify offensive content, Risch et al. [6] used a BERT model with distinct random seeds, while Subhanshu et al. [7] fine-tuned the BERT-based network model. Many existing text content recognition systems are based on Transformer [8] models.

https://github.com/kuiyongyi/hasoc_2021

Forum for Information Retrieval Evaluation, December 13-17, 2021, India

3964438@qq.com (Y. Kui)

The rest of the paper is structured as follows: In the second part of the paper, we give an overview of the tasks and datasets of this challenge; the third part describes the models used in this challenge; the fourth part describes the experimental process of Subtask A and Subtask B; in the fifth part, the official evaluation results of these two tasks are listed. In the last part, I summarized the evaluation results and the paper.

2. Task and Data Description

2.1. Subtasks

HASOC (2021) [9] Subtask 1 includes two subtasks, Subtask A and Subtask B. The main purpose is to detect the Hate Speech and Offensive Content of the text. The two subtasks are defined as follows:

Subtask A: Tweets predicted as hate speech and offensive speech are further divided into hate speech, offensive speech, and profane content. Therefore, it is a multiclass classification problem.

Subtask B: Tweets predicted to be hate speech and offensive speech in English and Hindi corpus are further identified into three categories: Hate speech, Offensive, and Profane. Subtask B is a text multi-classification task.

The evaluation standards of the prediction results of Subtask A and Subtask B are both Macro F1 and Macro Precision.

2.2. Dataset

The data for this challenge comes from comments on the Twitter platform, the corpus involves Marathi data [10], English and Hindi data . The Subtask A and Subtask B tasks of the HASOC (2021) Hate Speech and Offensive Content Identification Challenge [11] provide training datasets and testing datasets.

In order to get more data to train the model better, to make the model have a better generalization and reduce the risk of overfitting, I collected data on English and Hindi corpus in HASOC (2019) and HASOC (2020) challenges. After integrating the collected data, the amount of training data about English corpus and Hindi corpus is 12035 and 10215 respectively. Next, we used the shuffle function in the Sklearn package to shuffle the order of the data and finally divided the integrated data into Training Dataset and Validation Dataset at a ratio of 4/1. Table 1 specifically lists the data volume of the three languages in Subtask A and Subtask B.

2.3. Data pre-process

The datasets of Subtask A and Subtask B are sampled from Twitter, and the format of the data is informal. Tweets can be long or short, and there are a lot of emojis or URL links in the text, and even spelling errors in words.

Pre-processing step is applied on the data to make the model better extract the information carried by the text and help enhance the accuracy of the classifier. In this challenge, some of the

Table 1
The total data volume of English, Hindi, and Marathi corpus, and the training datasets and Validation datasets after dividing by the ratio of 4/1, as well as the data volume of Testing datasets provided by this challenge.

Language	Amount	Training Dataset	Validation Dataset	Testing Dataset
English	12035	9628	2407	1268
Hindi	10215	8172	2043	1469
Marathi	1874	1500	374	625

methods we used include: deleting URL links in the text, deleting emoticons and punctuation marks in the text.

3. System Description

3.1. Pre-trained Model

In this hate speech and offensive content detection challenge, I tried to use six pre-trained models: BERT, ALBERT, multilingual BERT (mBERT), DeBERTa, XLNet, and SqueezeBERT. Here is a brief introduction to each pre-trained model.

The BERT [12] model is a Deep Bidirectional model trained on the Transformer Encoder structure. The training process is divided into the pre-training stage and Fine-tuning stage, and its pre-training tasks include Masked LM (Language Model) and Next Sentence Prediction.

ALBERT [13] uses word embedding parameter factorization and hidden layer parameter sharing methods to reduce the amount of model parameters, and uses Sentence Order Prediction Loss to optimize Next Sentence Prediction Loss. Therefore, compared with the BERT model, it significantly reduces the amount of model parameters, while the performance loss is tiny.

mBERT is a Cross-language [14] model. It performs cross-language pre-training on data in 104 languages including English, Hindi, and Marathi. Texts in different languages share some common word blocks or vocabulary (such as numbers, links, etc.).

DeBERTa [15] model uses two methods to enhance BERT. The first is Disentangled Attention, and in this way, each word uses two vectors to encode the text and position respectively, and the attention weights between words are calculated separately by using a matrix of text and relative positions; the second technique is to introduce absolute positions in the Decode Layer to predict masked tokens.

Compared with the Masked of the BERT model, XLNet [16] introduces the new pre-training target of the Permutation Language Model during pre-training. In addition, XLNet introduces the Transformer XL mechanism, so it has an advantage over the Bert model for tasks where the input is a long text.

The SqueezeBERT [17] model applies experience in the computer vision field to the Natural Language Processing tasks. The SqueezeBERT model replaces several operations in self-attention layers with grouped convolutions. It replaces several operations in self-attention layers with grouped convolutions. This model has achieved high accuracy on the GLUE Dataset.

3.2. Common neural network

In this part, I will give a brief overview of several common neural networks used in the paper.

A Recurrent Neural Network (RNN) [18] is a neural network that can be used to process sequence data. The RNN model has a memory function, it can remember important words in the text.

Long Short-Term Memory (LSTM) [19] has the same memory function as RNN. LSTM uses a gate mechanism, so it can solve the problem of gradient disappearance to a certain extent. In this paper, the Bidirectional LSTM (BiLSTM) model is used, which can extract the contextual information of the text.

TextCNN [20] is a text classification model using convolutional networks. It passes the word vector through convolution and pooling operations, and finally sends the output to the softmax function to achieve classification. The structure of the TextCNN model is relatively simple, with fewer parameters, and good results can be achieved by introducing Pre-trained word vectors.

3.3. Integrated

In the integration process, we did not freeze the data initialized by the pre-trained model, but integrated the pre-trained model with a smaller-scale model (RNN, LSTM, or TextCNN) for training.

The pre-trained model uses a lot of data for training, it can get high-quality word and sentence embedding vectors. Therefore, we plan to add models such as RNN, BiLSTM, or TextCNN to the output layer of the pre-trained model to further extract high-dimensional features. Next, the results obtained by these relatively small-scale Neural Networks are sent to the Fully Connected Neural Network for classification. From our subsequent experimental results, we can see that the accuracy of this method is similar to that of the pre-trained model, but the result of this integration strategy increases the Macro F1 value in the official evaluation score by 0.3% to 1%. In fact, many downstream tasks have been achieved in this way.

4. Experiments

4.1. Subtask A Parameters Setting

In Subtask A, the optimizer selects AdamW; loss function uses Crossentropy; the epoch, max length, and batch size parameters are set to 10, 96, and 32 respectively; drop_out, learning rate, and weight_decay parameters are set to 0.4, 1e-5, and 1e-2, respectively.

4.2. Subtask A

First, I use six pre-trained models to conduct a text binary classification experiment under the parameters set above. These pre-trained models are BERT, ALBERT, mBERT, DeBERTa, XLNet, SqueezeBERT. Table 2 lists the specific performance of each pre-trained model on the respective validation datasets of the three language corpora.

The experimental results show that among the six pre-trained models, the DeBERTa model performs best on English corpus, and the mBERT model achieves the highest accuracy on the

Table 2
Evaluation results of six pre-trained models on subtask A's Validation datasets.

Model	Validation Accuracy		
	English	Hindi	Marathi
ALBERT	0.8064	0.6754	0.6905
BERT	0.8092	0.7911	0.7872
DeBERTa	**0.8212**	0.7533	0.7083
mBERT	0.8122	**0.8034**	**0.8908**
XLNet	0.8094	0.6825	0.6899
SqueezeBERT	0.8055	0.7588	0.7542

text binary classification task of Hindi and Marathi corpus. In the classification experiment of Subtask A, the six models all use a learning rate of 1e-5 in the training phase. Next, I use the common learning rates of 1e-6, 5e-6, 1e-5, 3e-5, and 5e-5 to train the DeBERTa+ BiLSTM model separately, and ensure that the remaining parameters remain unchanged. Table 3 lists the scores of the models trained with these five learning rates on the validation Dataset.

Table 3
The evaluation results of the DeBERTa + BiLSTM models trained with five common learning rates on the subtask A's Validation Dataset of the English corpus.

Learning Rate	1e-6	5e-6	1e-5	3e-5	5e-5
Accuracy	0.8143	0.8149	0.8201	0.8194	**0.8221**
Macro F1	0.8093	0.8088	0.8024	0.8019	**0.8182**

The results show that the DeBERTa+ BiLSTM model uses a 5e-5 learning rate to obtain the best performance on the Dataset of this challenge. So in the subsequent experiments of Subtask A and Subtask B, I chose to use the 5e-5 learning rate.

Next, I integrated the 4-layer RNN, 2-layer BiLSTM, and TextCNN models after the pre-trained model with the highest score in the three-language verification data set for experiments. During this experiment, the learning rate is set to 5e-5, and the other parameters are unchanged. Finally, the output results of the RNN, BiLSTM, or TextCNN model are sent to the fully connected layer for classification. The final models are obtained after DeBERTa and mBERT integrate RNN, BiLSTM or TextCNN models. Table 4,5 list the scores of the final models on their respective Validation Dataset.

From the experimental results listed in Table 4, it can be seen that the DeBERTa+ BiLSTM model performs best on the Validation Dataset of the English corpus, and the accuracy and Macro F1 score are improved compared to the DeBERTa model alone. Therefore, I use the prediction result of the DeBERTa+ BiLSTM model as the final submission result on the English Subtask A task. Similarly, it can be seen from Table 5 that the mBERT+ TextCNN model has the best performance on the Validation datasets of Hindi and Marathi corpus. So, I use the prediction results of the mBERT+ TextCNN model on the Hindi and Marathi test data sets as the final answer to the Hindi Subtask A and Marathi Subtask A tasks.

Table 4
The final models are obtained after the DeBERTa model integrates RNN, BiLSTM, and TextCNN models respectively, and the evaluation result of the final models on the English corpus Validation Dataset.

Model	English Validation Dataset	
	Accuracy	Macro F1
DeBERTa+RNN	0.8163	0.8020
DeBERTa+BiLSTM	**0.8201**	**0.8024**
DeBERTa+TextCNN	0.8188	0.8059

Table 5
The final models are obtained after the mBERT model integrates RNN, BiLSTM, and TextCNN models respectively, and the scores of the final models on the Validation datasets of Hindi and Marathi corpus.

	Hindi Validation Dataset		Marathi Validation Dataset	
Model	Accuracy	Macro F1	Accuracy	Macro F1
mBERT+RNN	0.8101	0.7906	0.8866	0.8712
mBERT+BiLSTM	0.8112	0.7951	0.8916	0.8748
mBERT+TextCNN	**0.8124**	**0.7964**	**0.8980**	**0.8831**

4.3. Subtask B

The experimental results of Subtask A show that the DeBERTa+ BiLSTM and mBERT+ TextCNN models perform best on the Validation datasets of English and Hindi corpus, respectively. Therefore, I still use these two models on the English and Hindi corpus in Subtask B.

The difference from Subtask A is that the fully connected layer of Subtask B outputs a matrix of batch_size * 4, while Subtask A outputs a matrix of batch_size * 2. Tables 6,7 respectively list the scores of the DeBERTa+ BiLSTM and mBERT+ TextCNN models on the Validation datasets of English Subtask B and Hindi Subtask B.

Table 6
DeBERTa+ BiLSTM model's accuracy and Macro F1 score on the Validation Dataset of the English corpus of Subtask B.

Model	English Validation Dataset	
	Accuracy	Macro F1
DeBERTa	0.7291	0.5968
DeBERTa+BiLSTM	**0.7356**	**0.6051**

5. Results

On Subtask A and Subtask B, among all teams, the solutions I put forward in the paper are ranked fourteenth and fifteenth on the English corpus, ranked sixth and fifth on the Hindi corpus, and ranked eleventh in the Marathi corpus. Table 8 lists the best scores on the corpus

Table 7
mBERT+ TextCNN model's accuracy and Macro F1 score on the Hindi Validation Dataset of Subtask B.

Model	Hindi Validation Dataset	
	Accuracy	Macro F1
mBERT	0.7266	0.5911
mBERT+TextCNN	**0.7337**	**0.5954**

of each language in Subtask A and Subtask B in the HASOC (2021) Challenge, as well as the official evaluation results of the answers I finally submitted.

Table 8
After the official evaluation, my final score on each task, my ranking, and the best result of each task.

Subtask	Macro F1		My Rank
	Best Score	My Score	
English Subtask A	0.8305	**0.8030**	6 / 56
English Subtask B	0.6657	**0.6116**	15 / 37
Hindi Subtask A	0.7825	**0.7725**	6 / 34
Hindi Subtask B	0.5603	**0.5509**	5 / 24
Marathi Subtask A	0.9144	**0.8611**	11 / 25

6. Conclusion

In this paper, I describe the solution I proposed in the HASOC (2021) challenge, including the pre-preprocessing of the data before training the model, the selection of the learning rate, and the construction of the final model. This challenge mainly includes the text classification tasks of English, Hindi, and Marathi. I solved the classification problem of English corpus by integrating the DeBERTa and BiLSTM models, and the classification problem of Hindi and Marathi corpus was solved by integrating the mBERT and TextCNN models.

The difficulties of Subtask A and Subtask B in this challenge are as follows. First of all, the text content is informal, the text length is generally short, and it lacks context, so it is difficult to obtain very high accuracy. Secondly, during the experiment, I found that the data distribution of each category in Subtask A and Subtask B is not uniform, which is also a reason why the model is biased to predict a category that appears more commonly. Finally, the amount of data in this challenge is not sufficient compared to large models like BERT, which leads to over-fitting, which makes the model perform poorly on the testing Dataset. In future research work, I will try to use a variety of fine-tuning strategies [21], or the idea of transfer learning [22] to continue to improve my solution.

References

[1] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys (CSUR) 51 (2018) 1–30.

[2] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, A. Patel, Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages, in: Proceedings of the 11th forum for information retrieval evaluation, 2019, pp. 14–17.

[3] 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, CoRR abs/2108.05927 (2021). URL: https://arxiv.org/abs/2108.05927. arXiv:2108.05927.

[4] 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).

[5] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, Ç. Çöltekin, Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020), arXiv preprint arXiv:2006.07235 (2020).

[6] J. Risch, R. Krestel, Bagging bert models for robust aggression identification, in: Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, 2020, pp. 55–61.

[7] S. Mishra, S. Mishra, 3idiots at hasoc 2019: Fine-tuning transformer neural networks for hate speech identification in indo-european languages., in: FIRE (Working Notes), 2019, pp. 208–213.

[8] S. A. Chowdhury, A. Abdelali, K. Darwish, J. Soon-Gyo, J. Salminen, B. J. Jansen, Improving arabic text categorization using transformer training diversification, in: Proceedings of the Fifth Arabic Natural Language Processing Workshop, 2020, pp. 226–236.

[9] T. Mandl, S. Modha, G. K. Shahi, H. Madhu, S. Satapara, P. Majumder, J. Schäfer, T. Ranasinghe, M. Zampieri, D. Nandini, A. K. Jaiswal, Overview of the HASOC subtrack at FIRE 2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages, in: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation, CEUR, 2021. URL: http://ceur-ws.org/.

[10] S. Gaikwad, T. Ranasinghe, M. Zampieri, C. M. Homan, Cross-lingual offensive language identification for low resource languages: The case of marathi, in: Proceedings of RANLP, 2021.

[11] S. Modha, T. Mandl, G. K. Shahi, H. Madhu, S. Satapara, T. Ranasinghe, M. Zampieri, Overview of the HASOC Subtrack at FIRE 2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages and Conversational Hate Speech, in: FIRE 2021: Forum for Information Retrieval Evaluation, Virtual Event, 13th-17th December 2021, ACM, 2021.

[12] 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).

[13] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, arXiv preprint arXiv:1909.11942

(2019).
[14] Z. Chi, L. Dong, F. Wei, N. Yang, S. Singhal, W. Wang, X. Song, X.-L. Mao, H. Huang, M. Zhou, Infoxlm: An information-theoretic framework for cross-lingual language model pre-training, arXiv preprint arXiv:2007.07834 (2020).
[15] P. He, X. Liu, J. Gao, W. Chen, Deberta: Decoding-enhanced bert with disentangled attention, arXiv preprint arXiv:2006.03654 (2020).
[16] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Advances in neural information processing systems 32 (2019).
[17] F. N. Iandola, A. E. Shaw, R. Krishna, K. W. Keutzer, Squeezebert: What can computer vision teach nlp about efficient neural networks?, arXiv preprint arXiv:2006.11316 (2020).
[18] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).
[19] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, Lstm: A search space odyssey, IEEE transactions on neural networks and learning systems 28 (2016) 2222–2232.
[20] T. Lei, R. Barzilay, T. Jaakkola, Molding cnns for text: non-linear, non-consecutive convolutions, arXiv preprint arXiv:1508.04112 (2015).
[21] S. Li, J. Wang, W. Xiang, K. Yang, Z. Li, W. Wang, An autoregulated fine-tuning strategy for titer improvement of secondary metabolites using native promoters in streptomyces, ACS synthetic biology 7 (2018) 522–530.
[22] N. Barhate, S. Bhave, R. Bhise, R. G. Sutar, D. C. Karia, Reducing overfitting in diabetic retinopathy detection using transfer learning, in: 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), IEEE, 2020, pp. 298–301.