Astralis@Hasoc 2020:Analysis On Identification Of Hate Speech In Indo-European Languages With Fine-Tuned Transformers.

Hiren Madhu[a], Shrey Satapara[b], and Harsh Rathod[c]

LDRP-ITR, Gandhinagar, India

Abstract

The detection of hate speech in online social media platforms is of great importance in text classification. There is a need to research languages other than English. In this paper, we describe our team Astralis' combined effort in the shared task HASOC. We analyzed various models such as Naive Bayes, SVM, ANN, CNN, and embeddings such as TF-IDF, Multilingual BERT, and OPENAI-GPT2. Our relative performance was better in Subtask B for all languages, with our best-performed system ranked in second position in German Subtask B.

Keywords [1]

Hate Speech Detection | Text Classification | CNN | Deep Learning| Transformers

1. Introduction

In today's world, many of us rely on social media platforms such as Facebook, Instagram and Twitter to find and connect with each other. While each has its benefits, it is essential to remember that it can never be a replacement for real-world human interaction. One such negative impact of using social media platforms is continuously getting affected by offensive comments and disrespectful behaviour from a complete stranger. In 2014, a Pew Research Center study found that about one-in-five (22%) internet users that had been victims of online harassment reported that it had happened in the comment section of a website. An average time spent by a person on social media is around 142 minutes a day. One thing which plays a significant role in spreading hate content is that a person can be anonymous and this anonymity causes the so-called online disinhibition effect, a psychological phenomenon that occurs when accepted social norms cease to exist in online contexts. When people are allowed to post without disclosing their identity (as in many forums and websites), their comments debase noticeably. It also turns out that exposure to online negativity makes our own thinking negative – reading uncivil comments can immediately increase readers' own hostile cognitions. We define offensive language as the offence of using language in a way which could offend a reasonable person. Comments and tweets are having an adverse effect on one's life. The various forms of abusive and offensive content online are a constant threat to users who use the platform. Hence it has become utterly essential to detect these unknown sources who are spreading hate among these social platforms so that necessary action can be taken.

HASOC[1] provided a multilingual track joining Indo-European languages such as Hindi, English, and German. The tweets classification is done in two sub-tasks, namely A and B, which are then further classified. This paper emphasizes the problem of offensive language detection. Sub-task A is a coarse-grained binary classification. Participating systems are required to classify tweets into two classes: Hate and Offensive (HOF) and Non- Hate and offensive (NOT).

- (NOT) Non-Hate-Offensive - This subcategory does not comprise of hate, offensive, and profane posts

FIRE '20, Forum for Information Retrieval Evaluation, December 16–20, 2020, Hyderabad, India.
EMAIL: hirenmadhu16@gmail.com (H. Madhu); shreysatapara@gmail.com (S. Satapara); harshrathod6874@gmail.com (H. Rathod)
ORCID: 0000-0002-6701-6782 (H. Madhu); 0000-0001-6222-1288 (S. Satapara);0000-0001-8599-2501 (H. Rathod)

- (HOF) Hate and Offensive - This division consists of Hate and offensive content.

Sub-task B is a fine-grained classification offered for English, German, Hindi. Hate-speech and offensive posts from the sub-task A are further classified into three categories.

- (HATE) Hate speech:- This contains a class of hate speech content.
- (OFFN) Offensive:- Posts under this class contain offensive content.
- (PRFN) Profane:- This subcategory contains profane content.

2. Related Work

Hate speech detection is a vast field of research and attracts many. Here we briefly describe some of the works done in this area.

The GermEval presented a task that identifies offensive language. The performance was measured by F1 score, precision, and recall.[2] This was a competition comprising 20 teams working on the shared task. The best results were achieved using five disjoint sets to train three different classifiers and then combining them, resulting in a meta-level classifier.[3]

SemEval 2019 focuses on studying the type and target of the offensive language. They presented a shared task called OffensEval[4]. A schema is defined for taking into account the class and target. The dataset used is Offensive language identification(OLID). Three sub-tasks were given according to their annotation schema, which the participating team had to use. Sub-task A was offensive language identification, Sub-task B was Automatic categorization of offence types Subtask C was Offense target identification.

Work was done on text classification using CNN by Yoon kim[5]. The CNN models discussed herein improve upon state of the art on 4 out of 7 tasks, including sentiment analysis and question classification. These vectors were trained by Mikolov et al. (2013) on 100 billion words of Google News and are publicly available.[6]

Nobata et al.[7] came up with a model that uses regression techniques to detect hate and offensive content from a speech. Djuric et al.[8] presented a model that used LR classifiers to identify hate content. Besides using conventional techniques, this research also used comment embeddings as one of their features.

3. Dataset

Below we briefly describe the dataset used for HASOC 2020. Given below is the class-wise distribution of the dataset provided to us during this task.

Table 1
Distribution of Train Set classes

	Subtask 1		Subtask 2				
Language	HOF	NOT	NONE	PRFN	OFFN	HATE	Total
English	1856	1852	1852	1377	321	158	3708
Hindi	847	2116	2116	148	465	234	2963
German	673	1700	1700	387	140	145	2373

Table 2
Distribution of Test Set classes

	Subtask 1		Subtask 2				
Language	HOF	NOT	NONE	PRFN	OFFN	HATE	Total
English	391	423	414	293	82	25	814
Hindi	197	466	493	27	87	56	663
German	134	392	378	88	36	24	526

4. Approach

Here we describe the various methodologies we used in different steps of the experiment.

4.1. Preprocessing

For preprocessing, we followed a few simple traditional steps. At first, all of the Twitter handles were removed. After that, the links in the tweets were removed. All the retweets in the data had "RT" at the start. So we removed that. Then we removed all the residual blanks and kept the emojis. As the transformers[9] model we used had emoji support.

4.2. Embeddings

Here we define the methodology that we used to analyze two different subtasks that were given to us. We have used various models to get the best possible results.

4.2.1. TF-IDF

It is short for the term frequency-inverse document frequency. One gram and bigrams were used to create this vectorizer with minimum occurrences of 5 for English. The length of the vectors derived was 1281

4.2.2. BERT[10]

BERT stands for Bidirectional Encoder Representation from Transformers. The hugging face[11] transformers library has made many transformers models available for use. From BERT, we used two models. bert-base-uncased for English and bert-base-multilingual-uncased for English, Hindi, and German. Matrices of length 768 * MAX_LEN was received for this. MAX_LEN is a parameter that shows the Max Length of tokenized tweets. Post padding with '0' was used.

4.2.3. OPENAI GPT-2[12]

Similarly, shaped matrices as BERT were received from this transformer model for Hindi, English, and German.

4.3. Models

We have used various Machine Learning algorithms and Deep Neural Networks, and here we describe them in detail. We used tensorflow[18] keras[19] to make all the Neural Networks.

4.3.1. SVM[13]

SVM performs exceptionally well in specific NLP scenarios. To implement it, we used the scikit-learn[14] library. We used SVC with RBF kernel. To implement this with matrices that we got from transformers, we used the Continuous Bag Of Words method, in which the mean of embeddings of each word is taken. So the input to SVM was 768 length vectors.

4.3.2. KNN

This was also implemented using scikit-learn library with neighbours set to 3. Here again, the CBOW method was used.

4.3.3. Naive Bayes[15]

Naive Bayes also works well for NLP. Here, the CBOW method was used to get the embeddings.

4.3.4. ANN

The input to this ANN was the CBOW embeddings and tf-idf for English. It was a five-layer NN with 128, 128, 256, 512 neurons and the last layer had 1 or 4 neurons depending upon the subtask.

4.3.5. CNN[16]

Here we have used a similar approach as the work produced by Yoon Kim. The architecture of CNN has layers in the following order.

1. Input: For initializing the input tensors
2. BatchNormalization[20]: To normalize each batch that is being processed.
3. Dropout[17]: Dropout helps to reduce the possibility of overfitting
4. After this, it is divided into various branches. Each branch computes x-gram.
    a. Conv1D: The kernel_size is set to x to compute x-gram.
    b. GlobalMaxPool1D: To get a vector of length of the number of filters
    c. BatchNormalization
5. The above three layers produce one x-gram. After this, all of the x-grams are merged into one Tensor.
6. The merged tensor is then passed into a Dense layer of 128 neurons.
7. And then, finally, a dense classifier layer with 1 or 4 neurons based on the subtask.

The architecture described above can be seen in Figure 1.

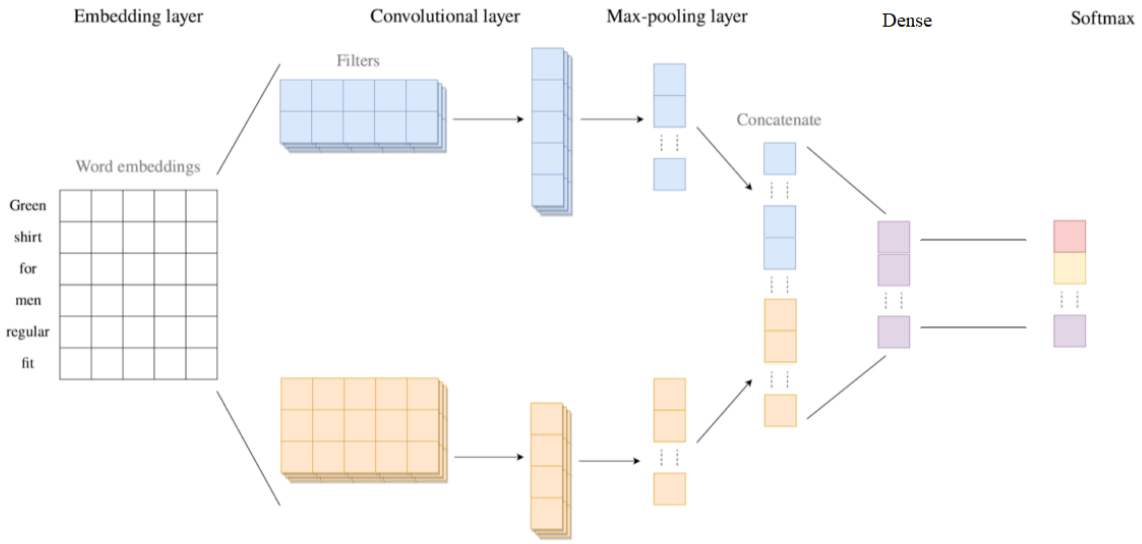

Figure 1: Architecture of Bigram + Trigram CNN model

First, we experimented with CNN with Bigram, Trigram, and Four-gram. Then after with Bigram and Trigram. In most cases, the Bigram + Trigram model was working better than the Bigram + Trigram + Four-gram model. So, for further analysis, we considered only the Bigram+Trigram model.

For the above models, training Adam optimizer [21] was used. We also used mild kernel regularization[22] in English subtask 2. We have also used class weights[23] in subtask B for all languages because there was an imbalance of classes in the dataset.

Table 3
Hyperparameters

Hyperparameter	Value
Learning Rate	1e-3
L1 regularization	1e-5
L2 Regularization	1e-4
Dropout	0.5
Filters	64
Activation	Relu
Activation in classifier neuron	SB1- Sigmoid SB2- Softmax
Loss	SB1- binary_crossentropy SB2 - categorical_crossentropy
Epochs in Subtask 1	English- 10 German- 17 Hindi- 25
Epochs in Subtask 2	English- 25 German- 25 Hindi- 25
TF-IDF Features	1281
Batch_size	256
SVM kernel	RBF
SVM regularization	0.025
Neighbours	3

5. Analysis

Experimental results in the available test set show that CNN(bigram + trigram) outperforms all other models. CNN can exceed most of all baseline models, precisely because of the nature of tweets. For example, tweets can be indirect texts. (e.g., sarcasm), full of noise and may not follow proper grammatical structure.

CNN can identify many small and large patterns in a tweet; if some are impacted by the noise[6], it can still use other patterns to determine the class, which can be seen in Table 4 and 5, which displays Embedding vs Model F1 Macro scores which is the metric used for scoring in HASOC 2020. The TF-IDF for vectors for English subtask A works well enough, but the subtask B performance is lower, which is caused by the imbalance in the dataset's distribution. The bert-based-uncased Bigram + Trigram model gives the best performance in English. Models for which CBOW was used do not perform on par with the CNN model for both the subtasks. The bert-base-multilingual-cased and the gpt2 transformers embedding gives a reasonably similar performance in some situations, and for

some, BERT performs a little better. bert-based-uncased performs better than bert-base-multilingual-cased in both the subtasks of English.

Table 4
Subtask A F1 Macro Scores of various models on Public Test Data

Embedding	TF-IDF	bert-base-uncased	bert-base-multilingual-cased			OPENAI GPT-2		
Language	English	English	English	Hindi	German	English	Hindi	German
SVM	0.618	0.738	0.719	0.412	0.427	0.827	0.412	0.513
Naive bayes	0.712	0.578	0.558	0.536	0.561	0.679	0.530	0.574
KNN	0.554	0.655	0.602	0.555	0.570	0.698	0.534	0.564
DNN	0.790	0.823	0.729	0.435	0.647	0.815	0.534	0.639
CNN(bi)	-	0.868	0.859	0.427	0.629	0.865	0.543	0.679
CNN(bi + tri)	-	0.826	0.858	0.482	0.665	0.874	0.508	0.724

Table 5
Subtask B F1 Macro Scores of various models on Public Test Data

Embedding	TF-IDF	bert-base-uncased	bert-base-multilingual-cased			GPT-2		
Language	English	English	English	Hindi	German	English	Hindi	German
SVM	0.246	0.347	0.351	0.213	0.209	0.412	0.210	0.275
Naive bayes	0.284	0.289	0.150	0.218	0.162	0.371	0.188	0.236
KNN	0.285	0.337	0.309	0.288	0.318	0.345	0.261	0.241
DNN	0.470	0.403	0.296	0.187	0.320	0.501	0.119	0.305
CNN(bi)	-	0.513	0.498	0.370	0.381	0.438	0.310	0.409
CNN(bi+tri)	-	0.543	0.528	0.384	0.410	0.495	0.354	0.428

So from the above work, we concluded to submit the CNN with Bigram + Trigram model with bert-based-uncased transformer for English subtasks and bert-base-multilingual-cased transformer for Hindi and German subtasks. The label wise F1 scores for the submitted models are shown in Table 6. OFFN and HATE categories from subtask 2 have relatively lower F1 scores due to relatively lower occurrences in the dataset. The final results on the private dataset on which the final ranks were given are displayed in Table 7.

Table 6
Label wise analysis of models submitted on the public test set

	Subtask 1		Subtask 2			
Language	HOF	NOT	NONE	PRFN	OFFN	HATE
English	0.855	0.852	0.789	0.783	0.188	0.000
Hindi	0.305	0.834	0.826	0.203	0.187	0.209
German	0.282	0.875	0.880	0.546	0.046	0.071

Table 7
Final results

Language/Subtask	Score	Rank	Score of rank 1
English/A	0.5017	10	0.5152
English/B	0.2484	5	0.2652
German/A	0.4789	16	0.5235
German/B	0.2627	2	0.2831
Hindi/A	0.4293	23	0.5337
Hindi/B	0.2644	3	0.3345

6. Conclusion

This paper describes offensive text identification into three Indo-European languages. We have shown our methodology for classifying tweets and posts from social media using multiple models in given three languages categorizing hate and offensive speech. After analyzing different models, we observed that the bert-based-uncased Bigram + Trigram model gives the best performance in English and bert-base-multilingual-cased transformer for Hindi and German subtasks. The results indicate that organizing profane and hate content is a strenuous task. In future work, we hope that better models and methods can be used to improve the effect of identifying hate speech.

7. References

[1] Mandl, Thomas and Modha, Sandip and Shahi, Gautam Kishore and Jaiswal, Amit Kumar and Nandini, Durgesh and Patel, Daksh and Majumder, Prasenjit and Schäfer, Johannes. 2020. Overview of the HASOC track at FIRE 2020: Hate Speech and Offensive Content Identification in Indo-European Languages. In Proceedings of the 12th annual meeting of the Forum for Information Retrieval Evaluation. Working Notes of FIRE 2020 - Forum for Information Retrieval Evaluation

[2] Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In Proceedings of GermEval.

[3] Montani, Joaquın Padilla.2018. Tuwienkbs at germeval 2018: German abusive tweet detection. In14thConference on Natural Language Processing KONVENS 2018, page 45

[4] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval).

[5] Kim, Y.: Convolutional neural networks for sentence classification. CoRR abs/1408.5882 (2014), http://arxiv.org/abs/1408.5882

[6] Mikolov, Tomas & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

[7] Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., Chang, Y.: Abusive Language Detection in Online User Content. In: WWW 2016. pp. 145–153. Montreal (2016)

[8] Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., Bhamidipati, N.: Hate Speech Detection with Comment Embeddings. In: WWW 2015. pp. 29–30. Florence, Italy (2015)

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, "Attention Is All You Need". arXiv:1706.03762 https://arxiv.org/pdf/1706.03762.pdf

[10] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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

[11] Wolf, Thomas & Debut, Lysandre & Sanh, Victor & Chaumond, Julien & Delangue, Clement & Moi, Anthony & Cistac, Pierric & Rault, Tim & Louf, Rémi & Funtowicz, Morgan & Brew, Jamie. (2019). Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771v5 [cs.CL] https://arxiv.org/pdf/1910.03771.pdf

[12] Radford, Alec, and Wu, Jeff, and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya, Language Models are Unsupervised Multi-Task Learners, (2019)

[13] Marti A. Hearst. 1998. Support Vector Machines. IEEE Intelligent Systems 13, 4 (July 1998), 18–28. DOI:https://doi.org/10.1109/5254.708428

[14] Scikit-learn: Machine Learning in Python, Pedregosa, et al., JMLR 12, pp. 2825-2830, 2011.

[15] Naïve Bayes, Webb, Geoffrey I., Sammut, Claude, Webb, Geoffrey I., Encyclopedia of Machine Learning, 2010, Springer US, Boston, MA, 978-0-387-30164-8, Webb 2010, 10.1007/978-0-387-30164-8_576, https://doi.org/10.1007/978-0-387-30164-8_576, 713-714

[16] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Back-propagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958

[18] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian

Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike, Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon, Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software is available from tensorflow.org.

[19] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[20] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15). JMLR.org, 448–456.

[21] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[22] Andrew Y. Ng. 2004. Feature selection, L1 vs. L2 regularisation, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (ICML '04). Association for Computing Machinery, New York, NY, USA, 78. DOI: https://doi.org/10.1145/1015330.1015435

[23] Adaptive Weight Optimization for Classification of Imbalanced Data, Huang, Wenhao, Song, Guojie, Li, Man, Hu, Weisong, Xie, Kunqing, Sun, Changyin, Fang, Fang, Zhou, Zhi-Hua, Yang, Wankou, Liu, Zhi-Yong, Intelligence Science and Big Data Engineering, 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 978-3-642-42057-3, 10.1007/978-3-642-42057-3_69