Detecting Offensive Language in English, Hindi, and Marathi using Classical Supervised Machine Learning Methods and Word/Char N-grams

Yaakov HaCohen-Kerner[1] and Moshe Uzan[2]

[1] *Computer Science Department, Jerusalem College of Technology, Jerusalem 9116001, Israel*
[2] *Computer Science Department, Bar Ilan University, Ramat-Gan 5290002, Israel*

Abstract

In this paper, we describe our submissions for the HASOC 2021 contest. We tackled subtask 1A that addresses the problem of hate speech and offensive language identification in three languages: English, Hindi, and Marathi. We developed different models using six classical supervised machine learning methods: support vector classifier, binary support vector classifier, random forest, ada-boost classifier, multi-layer perceptron, and logistic regression. Our best submission was a model we built for offensive language identification in Marathi using random forest. This model was ranked in 6th place out of 25 teams. Our result is lower by only 0.0059 than the result of the team that was ranked in 3rd place. Our ML models were applied on various combinations of character and/or word n-gram features from uni-gram to 8-gram.

Keywords

Hate Speech, offensive language, supervised machine learning, word/char n-grams

1. Introduction

There is no universal definition of "offensive language". Jay and Janschewitz [1] define offensive language as vulgar, pornographic, and hateful language. Xu and Zhu [2] noted that the definition of offensive language can be subjective because different people interpret differently the same content. Xu and Zhu accepted the definition of offensive language given by the Internet Content Rating Association (ICRA)[2], as text that includes gutter language, sexually explicit material, racist, graphic violence, or any other content that may be considered offensive on social, religious, cultural or moral grounds. Another popular and general definition for offensive language is any implicit or explicit attack or insult against one person or a group of people.

Offensive language is one of the main problems of online communities and their users. Offensive language posts in social networks (e.g., Twitter, Facebook, and post blogs) are spreading quickly and easily. This phenomenon undermines the reputation of online communities, impairs their growth of alienates their users.

Distinguishing offensive language and hate speech from non-offensive language and non-hate is challenging because of various reasons, such as (1) hate speech not always includes offensive slurs and offensive language does not always express hate; (2) the large variety of implicit and explicit ways to verbally attack a target person or group; (3) too short tweets; and (4) incoherent tweets.

One of the recent results of this challenge was the organization of several tournaments about the detection of various types of offensive language in different languages (e.g., English, German, Hindi, Tamil, and Malayalam): HASOC 2019 [3], HASOC 2020 [4], SemEval-2019 [5], and SemEval-2020 [6].

Forum for Information Retrieval Evaluation, December 13-17, 2021, India
EMAIL: kerner@jct.ac.il; uzanmacho@gmail.com
ORCID: 0000-0002-4834-1272


[2] http://www.icra.org/

In these tournaments, the use of natural language processing (NLP) and machine learning (ML) models to detect offensive language has been found useful.

Users, especially weak populations such as old age, children, youth, women, and some of the minorities are exposed to various dangers such as fear, panic, and hatred of specific people or population groups and in some cases even harm to their physical or mental health.

The motivation for offensive language detection research is quite clear. There is a need for high-quality systems, which can detect offensive language posts, prevent their spread, and report them to the responsible authorities. Such systems will help to improve the protection and security of the people, especially in terms that are related to physical or mental health.

The structure of the rest of the paper is as follows. Section 2 introduces the general background concerning offensive language. Section 3 describes the HASOC 2021 Subtask 1A. In Section 4, we present the applied models and their experimental results. Section 5 summarizes, concludes, and suggests ideas for future research.

2. Related Work

Early studies [7-8] referred to hate speech as abusive and hostile messages or flames. Recent authors [9-11] preferred to employ the term cyberbullying. However, more terms related to hate speech are often used in the NLP community, such as discrimination, flaming, abusive language, profanity, toxic language, or comment [12].

Several notable articles in the field of offensive language detection are Davidson et al. [13], Fortuna and Nunes [14], and Pitsilis et al. [15]. A large-scale semi-supervised dataset for offensive language identification was introduced by Rosenthal et al. [16].

In many cases, preprocessing can “clean” the data and improve its quality. There are various basic types of preprocessing methods e.g., conversion of uppercase letters into lowercase letters, HTML tag removal, punctuation mark removal, and stop-word removal.

HaCohen-Kerner et al. [17] investigated the impact of all possible combinations of six preprocessing methods (spelling correction, HTML tag removal, converting uppercase letters into lowercase letters, punctuation mark removal, reduction of repeated characters, and stopword removal) on TC in three benchmark mental disorder datasets. In one dataset, the best result showed a significant improvement of approximately 28% over the baseline result using all six preprocessing methods. In the other two datasets, several combinations of preprocessing methods showed minimal improvements over the baseline results.

In another study, HaCohen-Kerner et al. [18] explored the influence of various combinations of the same six basic preprocessing methods mentioned in the previous paragraph on TC in four general benchmark text corpora using a bag-of-words representation. The general conclusion was that it is always advisable to perform an extensive and systematic variety of preprocessing methods, combined with TC experiments because this contributes to improving TC accuracy.

The authors of this paper published two workshop papers about offensive language detection [19-20].

3. Task Description

The HASOC 2021 Subtask 1A "Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages" addresses the problem of hate speech and offensive language identification for English, Hindi, and Marathi. Sub-task A is coarse-grained binary classification in which participating systems are required to classify tweets into two classes:

- (NOT) Non-Hate-Offensive - This post does not contain any Hate speech, profane, offensive content.
- (HOF) Hate and Offensive - This post contains Hate, offensive, and profane content.

The overview of the HASOC Sub-track at FIRE 2021 is described in [21]. Additional information about the Subtask-1 in English and Hindi is described in [22]. The dataset for Subtask 1A in the Marathi language is described in [23]. The HASOC 2021 train and test datasets for English, Hindi, and Marathi are located at https://hasocfire.github.io/hasoc/2021/dataset.html.

4. Applied Models and Their Experimental Results

We used the given training and test datasets (see the end of the previous section). Due to time limitations, we applied only one preprocessing method - converting uppercase letters into lowercase letters (LC) and only six classical supervised ML methods: Support Vector Classifier (SVC), Binary Support Vector Classifier (BSVC), Random Forest (RF), Ada-Boost Classifier (ABF), Multi-Layer Perceptron (MLP), and Logistic Regression (LR) using classical features like char n-gram features and word n-gram features.

Random forest (RF) is an ensemble learning method for classification and regression [24]. Ensemble methods use multiple learning algorithms to obtain improved predictive performance compared to what can be obtained from any of the constituent learning algorithms. RF operates by constructing a multitude of decision trees at training time and outputting classification for the case at hand. RF combines Breiman's "bagging" (Bootstrap aggregating) idea in [25] and a random selection of features introduced by Ho [26] to construct a forest of decision trees.

SVC is a variant of the support vector machine (SVM) ML method [27] implemented in SciKit-Learn. SVC uses LibSVM [28], which is a fast implementation of the SVM method. SVM is a supervised ML method that classifies vectors in a feature space into one of two sets, given training data. It operates by constructing the optimal hyperplane dividing the two sets, either in the original feature space or in higher dimensional kernel space.

A Binary Support Vector Classifier (BSVC) is based on a class of linear hyperplanes, to separate elements into two specific classes, based on class specifying attributes using a hyperplane. The basic principle of a BSVC is as follows: given a dataset comprising data from two different classes it constructs an optimal linear classifier in the form of a hyperplane, which has the maximum margin [29].

The Ada-Boost Classifier (ABF) is one of the ensemble boosting classifiers [30]. AdaBoost is an iterative ensemble method that combines multiple weakly performing classifiers to increase the accuracy of classifiers. The basic principle is to set the weights of classifiers and train the data sample in each iteration such that it ensures the accurate predictions of unusual observations.

Multi-layer perceptron (MLP) is a deep, artificial neural network [31]. This model is based on a network of the computational unit, called perceptron, interconnected in a feed-forward way. The network is composed of layers of perceptron that each one has directed connections to the neurons of the subsequent layer. Usually, these units apply a sigmoid function, called the activation function, on the input they get and feed the next layer with the output of the function. This model is very useful especially when the data is not linearly separable.

Logistic Regression (LR) [32-33] is a linear model for classification. It is known also as maximum entropy regression (MaxEnt), logit regression, and the log-linear classifier. In this model, the probabilities describing the possible outcome of a single trial are modeled using a logistic function.

These ML methods were applied using the following tools and information sources: The Python 3.7.3 programming language and Scikit-learn – a Python library for ML methods.

In our experiments, we test dozens of TC models for each language. Tables 1-3 present 10/11 best F-Measure results of our models for English, Hindi, and Marathi, respectively. The results were calculated for the test sub-dataset according to models that were built using the training sub-dataset. As mentioned above, we applied six different supervised ML methods for various combinations of character and/or word n-gram features from uni-gram to 8-gram. We fine-tuned the number of character and/or word n-grams. However, we did not fine-tune the parameter values. We used their default values. The best result of each ML method (column) in each Table is colored in red. The best result in each Table is colored in bold red.

Table 1

10 Best F-Measure Results of our Models for English

Preproc-essing	Features	n of n-grams (uni-gram /bi-gram ...)	Number of features	Minimal number of occurrences	SVC	BSVC	RF	ABF	MLP	LR
LC	chargram	5	750	2	0.6884	0.6868	0.7023	0.7448	0.7238	0.6914
LC	chargram	4	800	3	0.7147	0.7132	0.7168	**0.7617**	0.7276	0.7165
LC	wordgram	1	175	2	0.7375	0.7385	0.7197	0.7439	0.7223	0.7429
LC	chargram chargram	5 4	750 800	2 3	0.6933	0.6870	0.7193	0.7223	0.7255	0.7023
LC	chargram wordgram	5 1	750 175	2 2	0.6992	0.6973	0.7323	0.7503	0.7348	0.7005
LC	chargram wordgram	4 1	800 175	3 2	0.6889	0.6992	0.7202	0.7545	0.7318	0.7086
LC	chargram wordgram chargram	4 1 5	800 175 750	3 2 2	0.6818	0.6790	0.7120	0.7323	0.7427	0.6941
LC	Wordgram-TfIdf	1	225	2	0.7453	0.7302	0.7120	0.7256	0.7227	0.7538
LC	Chargram-TfIdf	6	2500	2	0.6966	0.6926	0.7096	0.7094	0.7438	0.7054
LC	Chargram-TfIdf	8	1075	2	0.6404	0.6366	0.6968	0.7369	0.7004	0.6557

Table 2

11 Best F-Measure Results of our Models for Hindi

Preproc-essing	Features	n of n-grams (uni-gram /bi-gram ...)	Number of features	Minimal number of occurrences	SVC	BSVC	RF	ABF	MLP	LR
LC	Chargram	3	2975	3	0.6711	0.6671	0.7538	0.7374	0.7261	0.6960
LC	Chargram	4	2375	3	0.7026	0.6839	0.7452	0.7353	0.7254	0.6966
LC	Wordgram	1	1575	2	0.6012	0.6079	0.6978	0.6802	0.6637	0.6375
LC	Chargram Chargram	4 3	2375 2975	3 3	0.6914	0.6850	0.7409	0.7308	0.7338	0.7145
LC	chargram wordgram	3 1	2975 1575	3 2	0.6426	0.6426	0.7406	0.7211	0.6823	0.6723
LC	chargram wordgram	4 1	2375 1575	3 2	0.6833	0.6824	0.7363	0.7298	0.7377	0.7088
LC	chargram wordgram chargram	4 1 3	2375 1575 2975	3 2 3	0.6777	0.6739	0.7399	0.7270	0.7181	0.7192
LC	Chargram-TfIdf	2	4125	3	0.6118	0.6142	0.7484	0.7308	0.6911	0.6451
LC	Chargram-TfIdf	2	825	3	0.6853	0.6811	0.7423	0.7164	0.7084	0.6997
LC	Wordgram-TfIdf	1	1575	2	0.6012	0.6079	0.6978	0.6802	0.6637	0.6375
LC	Chargram-TfIdf	6	1250	2	0.6791	0.6791	0.7271	0.7561	0.7244	0.7009

Table 3
10 Best F-Measure Results of our Models for Marathi

Preproc-essing	Features	n of n-grams (uni-gram /bi-gram ...)	Number of features	Minimal number of occurrences	SVC	BSVC	RF	ABF	MLP	LR
LC	chargram	3	1825	3	0.7773	0.7874	0.8917	0.8293	0.8342	0.8359
LC	chargram	4	3550	3	0.7746	0.7750	**0.9016**	0.8517	0.8238	0.8038
LC	wordgram	1	475	2	0.6803	0.6707	0.7865	0.7288	0.7341	0.7021
LC	chargram chargram	3 4	1825 3550	3 3	0.7838	0.7868	0.8955	0.8401	0.8385	0.8316
LC	chargram wordgram	3 1	1825 475	3 2	0.8047	0.8047	0.8858	0.8374	0.8311	0.8284
LC	chargram wordgram	4 1	3550 475	3 2	0.7863	0.7859	0.8893	0.8147	0.8507	0.8273
LC	chargram wordgram chargram	4 1 3	3550 475 1825	3 2 3	0.7863	0.7970	0.8866	0.8252	0.8514	0.8273
-	Chargram TfIdf	3	4450	3	0.8356	0.8278	0.8768	0.8260	0.8212	0.8155
-	Chargram TfIdf	5	2450	2	0.7885	0.7841	0.8866	0.8641	0.8390	0.8007
-	Chargram TfIdf	6	3900	2	0.7289	0.7400	0.8685	0.8360	0.7853	0.7585

For each language, we submitted the top three models according to their F-Measure results. Our best F-Measure results (BIU's results) in the competition were as follows: English (F1 = 0.7388, 44th place) using RF with 175 word-unigrams, Hindi (F1 = 0.7400, 19th place) using RF with 2,975 tri-char grams, and Marathi (F1 = 0.8697, 6th place). Our best submission was the model we built for offensive language identification in Marathi using RF with 3,550 char 4-grams, 325 word uni-grams, and 400 word tri-grams. This model was ranked in 6th place out of 25 teams. Our result is lower by only 0.0059 than the result (0.8756) of the team that was placed in 3rd place.

Error analysis of the test dataset showed four types of errors: (1) too short tweets, e.g., “shag” or “Bloody hell”, which do not provide enough context information; (2) tweets that do not provide necessary world knowledge to understand the meaning, e.g., “#ChineseVirus”; (3) use of common offensive vocabulary with non-hateful intent, e.g. “Where are the fucking vaccines”; and (4) incoherent tweets, e.g., “So now Ram mandir is mudizee s achievement? SC bench is bjp I guess”

5. Summary, Conclusions, and Future Work

In this paper, we described our submitted models for subtask 1A of the HASOC 2021 competition that addresses the problem of hate speech and offensive language identification in three languages: English, Hindi, and Marathi. Due to time limitations, we applied only classical ML methods (i.e., no deep learning methods): SVC, BSVC, RF, ABF, MLP, and LR. These ML methods were applied on various combinations of character and/or word n-gram features from uni-gram to 8-gram.

Two interesting phenomena have been discovered. (A) While for Marathi the use of a classical learning method such as RF was sufficient for a relatively high result and a relatively high place (6). (B) In the English and Hindi languages, the use of classical learning methods such as RF, LR and SVC was not enough for good results. The models who achieved first places (at least according to some of their names) in these languages used deep learning methods such as various BERT variants.

Many Twitter messages contain ambiguous acronyms. Future research may apply the acronym disambiguation [34-35] that might enable better classification. It is also suggested to examine the usefulness of skip character n-grams that can serve as generalized n-grams, which allow overcoming frequent problems in Twitter messages, e.g., noise and sparse data [36]. Other ideas that may lead to better classification are to use for the TC process: stylistic feature sets [37], key phrases [38], and summaries [39]. Application of various deep learning models is of course potential for improving the TC results, especially, in languages such as English and Marathi, where the best models in the competition seem to have used different BERT variants and other deep learning models.

6. Acknowledgments

We thank the Bar-Ilan Data Science Institute for kindly providing a server for training our models. Without their support, this research would not have been possible. We are grateful to two anonymous reviewers and the organizers for their fruitful comments and suggestions.

7. References

[1] T. Jay, K. Janschewitz, The pragmatics of swearing, Journal of Politeness Research 4 (2008) 267-288.

[2] Z. Xu, S. Zhu, Filtering offensive language in online communities using grammatical relations. In Proceedings of the Seventh Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, 2010, pp. 1-10.

[3] T. Mandl, S. Modha, P.,Majumder, Patel, D., Dave, M., Mandlia, C., & Patel, A. (2019, December). 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 (pp. 14-17).

[4] T. Mandl, S. Modha, M, A. Kumar, B. R. Chakravarthi, (Overview of the hasoc track at fire 2020: Hate speech and offensive language identification in tamil, malayalam, hindi, english and german. In Forum for Information Retrieval Evaluation, 2020, pp. 29-32.

[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), 2019, arXiv preprint arXiv:1903.08983.

[6] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, ..., Ç. Çöltekin, SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020), 2020, arXiv preprint arXiv:2006.07235.

[7] E. Spertus, Smokey: Automatic recognition of hostile messages, in: Aaai/iaai, 1997, pp.1058–1065.

[8] D. Kaufer, Flaming: A white paper, Department of English, Carnegie Mellon University, Retrieved July 20 (2000) 2012.

[9] J. M. Xu, K. S. Jun, X. Zhu, A. Bellmore, Learning from bullying traces in social media, in: Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies, 2012, pp. 656–666.

[10] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, S. Mishra, Detection of cyberbullying incidents on the instagram social network, 2015, arXiv preprint arXiv:1503.03909.

[11] H. Zhong, H. Li, A. C. Squicciarini, S. M. Rajtmajer, C. Griffin, D. J. Miller, C. Caragea, Content-driven detection of cyberbullying on the instagram social network, in: IJCAI, 2016, pp. 3952–3958.

[12] A. Schmidt, M. Wiegand, A survey on hate speech detection using natural language processing, in: Proceedings of the Fifth International workshop on natural language processing for social media, 2017, pp. 1–10.

[13] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated hate speech detection and the problem of offensive language, In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 11, No. 1), 2017.

[14] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys (CSUR), 51(4) (2018) 1-30.

[15] G. K. Pitsilis, H. Ramampiaro, H. Langseth, Detecting offensive language in tweets using deep learning, 2018, arXiv preprint arXiv:1801.04433.

[16] S. Rosenthal, P. Atanasova, G. Karadzhov, M. Zampieri, P. Nakov, A large-scale semi-supervised dataset for offensive language identification, 2020, arXiv preprint arXiv:2004.14454.

[17] Y. HaCohen-Kerner, Y. Yigal, D. Miller, The impact of Preprocessing on Classification of Mental Disorders, in Proc. of the 19th Industrial Conference on Data Mining, (ICDM 2019), New York, 2019.

[18] Y. HaCohen-Kerner, D. Miller, Y. Yigal, The influence of preprocessing on text classification using a bag-of-words representation, PloS one, vol. 15, p. e0232525, 2020.
[19] M. Uzan, Y. HaCohen-Kerner, JCT at SemEval-2020 Task 12: Offensive language detection in tweets using preprocessing methods, character and word n-grams, in: Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2017–2022.
[20] M. Uzan, Y. HaCohen-Kerner, Detecting Hate Speech Spreaders on Twitter using LSTM and BERT in English and Spanish, CLEF 2021 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings (CEUR-WS.org), 21-23/Sep 2021, Bucharest, Romania.
[21] S. Modha, T. Mandl, G. K. Shahi, H. Madhu, S. Satapara, T. Ranas-inghe, 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.
[22] T. Mandl, S. Modha, G. K. Shahi, H. Madhu, S. Satapara, P. Majumder, J. Schäfer, T. Ranas-inghe, M. Zampieri, D. Nandini, A. K. Jaiswal, Overview of the HASOC subtrack at FIRE2021: 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/.
[23] 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.
[24] L. Breiman, Random forest, Machine Learning 45(1) (2001) 5-32.
[25] L. Breiman, Bagging predictors, Machine Learning 24(2) (1996) 123-140.
[26] T. K. Ho, Random decision forests, In Proceedings of 3rd International Conference on Document Analysis and Recognition, 1995, Vol. 1, pp. 278-282, IEEE.
[27] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.
[28] C.-C., Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM transactions on intelligent systems and technology (TIST) 2 (2011) 1–27.
[29] S. Mahadevan, S. L. Shah, Fault detection and diagnosis in process data using one-class support vector machines. Journal of process control 19(10) (2009) 1627-1639.
[30] Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm. In icml Vol. 96, 1996, pp. 148-156).
[31] S. K. Pal, S. Mitra, Multilayer perceptron, fuzzy sets, classification, IEEE transactions on Neural Networks 3(5),1992, 683-697.
[32] D. R. Cox, The regression analysis of binary sequences, Journal of the Royal Statistical Society: Series B (Methodological), 20 (1958) 215–232.
[33] D. W. Hosmer Jr, S. Lemeshow, R. X. Sturdivant, Applied logistic regression, Vol. 398, John Wiley & Sons. Applied logistic regression (Vol. 398). John Wiley & Sons, 2013.
[34] Y. HaCohen-Kerner, A. Kass, A. Peretz, Combined one sense disambiguation of abbreviations. In Proceedings of ACL-08: HLT, Short Papers, Association for Computational Linguistics, Columbus, Ohio, 2008, pp. 61-64, URL: https://aclanthology.org/P08-2.
[35] Y. HaCohen-Kerner, A. Kass, A. Peretz, Haads: A hebrew aramaic abbreviation disambiguation system, Journal of the American Society for Information Science and Technology 61(9) (2010) 1923–1932.
[36] Y. HaCohen-Kerner, Z. Ido, R. Ya'akobov, Stance classification of tweets using skip char Ngrams, In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2017, pp. 266-278, Springer, Cham.
[37] Y. HaCohen-Kerner, H. Beck, E. Yehudai, M. Rosenstein, D. Mughaz, Cuisine: Classification using stylistic feature sets and/or name-based feature sets, Journal of the American Society for Information Science and Technology 61(8) (2010) 1644-1657.
[38] Y. HaCohen-Kerner, I. Stern, D. Korkus, E. Fredj, Automatic machine learning of keyphrase extraction from short html documents written in hebrew, Cybernetics and Systems: An International Journal, 38(1) (2007) 1–21.
[39] Y. HaCohen-Kerner, E. Malin, I. Chasson, Summarization of jewish law articles in hebrew, Proceedings of the 16th International Conference on Computer Applications in Industry and Engineering (CAINE), November 11-13, 2003, Imperial Palace Hotel, Las Vegas, Nevada, USA, 2003, pp. 172–177.