Leveraging Multilingual Transformers for Hate Speech Detection

Sayar Ghosh Roy, Ujwal Narayan, Tathagata Raha, Zubair Abid and Vasudeva Varma

Information Retrieval and Extraction Lab, International Institute of Information Technology, Hyderabad, India

Abstract

Detecting and classifying instances of hate in social media text has been a problem of interest in Natural Language Processing in the recent years. Our work leverages state of the art Transformer language models to identify hate speech in a multilingual setting. Capturing the intent of a post or a comment on social media involves careful evaluation of the language style, semantic content and additional pointers such as hashtags and emojis. In this paper, we look at the problem of identifying whether a Twitter post is hateful and offensive or not. We further discriminate the detected toxic content into one of the following three classes: (a) Hate Speech (HATE), (b) Offensive (OFFN) and (c) Profane (PRFN). With a pre-trained multilingual Transformer-based text encoder at the base, we are able to successfully identify and classify hate speech from multiple languages. On the provided testing corpora, we achieve Macro F1 scores of 90.29, 81.87 and 75.40 for English, German and Hindi respectively while performing hate speech detection and of 60.70, 53.28 and 49.74 during fine-grained classification. In our experiments, we show the efficacy of Perspective API features for hate speech classification and the effects of exploiting a multilingual training scheme. A feature selection study is provided to illustrate impacts of specific features upon the architecture's classification head.

Keywords

HASOC 2020, Hate Speech Detection, XLM-RoBERTa, Perspective API

1. Introduction

With a rise in the number of posts made on social media, an increase in the amount of toxic content on the web is witnessed. Measures to detect such instances of toxicity is of paramount importance in today's world with regards to keeping the web a safe and healthy environment for all. Detecting hateful and offensive content in typical posts and comments found on the web is the first step towards building a system which can flag items with possible adverse effects and take steps necessary to handle such behavior.

In this paper, we look at the problem of detecting hate speech and offensive remarks within tweets. More specifically, we attempt to solve two classification problems. Firstly, we try to assign a binary label to a tweet indicating whether it is hateful and offensive (class HOF) or not

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

sayar.ghosh@research.iiit.ac.in (S. Ghosh Roy); ujwal.narayan@research.iiit.ac.in (U. Narayan); tathagata.raha@research.iiit.ac.in (T. Raha); zubair.abid@research.iiit.ac.in (Z. Abid); vv@iiit.ac.in (V. Varma)
https://sayarghoshroy.github.io/ (S. Ghosh Roy); https://www.ujwalnarayan.ml/ (U. Narayan); https://github.com/tathagata-raha/ (T. Raha); https://zubairabid.com/ (Z. Abid); https://irel.iiit.ac.in/vasu/index.html (V. Varma)

(class NOT). Secondly, if the tweet belongs to class HOF, we classify it further into one of the following three possible classes: (a) HATE: Contains hate speech, (b) OFFN: Is offensive, and (c) PRFN: Contains profanities.

The language in use on the web is in a different text style as compared to day-to-day speech, formally written articles, and webpages. In order to fully comprehend the social media style of text, a model needs to have knowledge of the pragmatics of emojis and smileys, the specific context in which certain hashtags are being used, and it should be able to generalize to various domains. Also, social media text is full of acronyms, abbreviated forms of words and phrases, orthographic deviations from standard forms such as dropping of vowels from certain words, and contains instances of code mixing.

The escalation in derogatory posts on the internet has prompted certain agencies to make toxicity detection modules available for web developers as well as for the general public. A notable work in this regard is Google's Perspective API[1] which uses machine learning models to estimate various metrics such as toxicity, insult, threat, etc., given a span of text as input. We study the usefulness of these features for hate speech detection tasks in English and German.

In recent years, utilizing Transformer-based [1] Language Models pre-trained with certain objectives on vast corpora [2] has been crucial to obtaining good representations of textual semantics. In our work, we leverage the advances in language model pre-training research and apply the same to the task of hate speech detection. Lately, we have witnessed the growing popularity of multilingual language models which can work upon input text in a language independent manner. We hypothesize that such models will be effective on social media texts across a collection of languages and text styles. Our intuition is experimentally verified as we are able to obtain respectable results on the provided testing data for the two tasks in question.

2. Related Work

In this section, we will provide a brief overview of the variety of methods and procedures applied in attempts to solve the problem of hate speech detection. Approaches using Bag of Words (BoW) [3] typically lead to a high number of false positives. They also suffer from data sparsity issues. In order to deal with the large number of false positives, efforts were made to better characterize and understand the nature of hate speech itself. This led to the formation of finer distinctions between the types of hate speech [4]; in that, hate speech was further classified into "profane" and "offensive". Features such as N-gram graphs [5] or Part of Speech features [6] were also incorporated into the classification models leading to an observable rise in the prediction scores.

Later approaches used better representation of words and sentences by utilizing semantic vector representations such as word2vec [7] and GloVe [8]. These approaches outshine the earlier BoW approaches as similar words are located closer together in the latent space. Thus, these continuous and dense representations replaced the earlier binary features resulting in a more effective encoding of the input data. Support Vector Machines (SVMs) with a combination of lexical and parse features have been shown to perform well for detecting hate speech as well. [6]

[1] https://www.perspectiveapi.com/

Language	Train	Test
English	3708	814
German	2373	526
Hindi	2963	663

Table 1
Dataset Size (in number of tweets)

The recent trends in deep learning led to better representations of sentences. With RNNs, it became possible to model larger sequences of text. Gated RNNs such as LSTMs [9] and GRUs [10] made it possible to better represent long term dependencies. This boosted classification scores, with LSTM and CNN-based models significantly outperforming character and word based N-gram models. [11] Character based modelling with CharCNNs [12] have been applied for hate speech classification. These approaches particularly shine in cases where the offensive speech is disguised with symbols like '*', '$' and so forth. [13]

More recently, attention based approaches like Transformers [1] have been shown to capture contextualized embeddings for a sentence. Approaches such as BERT [2] which have been trained on massive quantities of data allow us to generate robust and semantically rich embeddings which can then be used for downstream tasks including hate speech detection.

There have also been a variety of open or shared tasks to encourage research and development in hate speech detection. The TRAC shared task [14] on aggression identification included both English and Hindi Facebook comments. Participants had to detect abusive comments and distinguish between overtly aggressive comments and covertly aggressive comments. OffensEval (SemEval-2019 Task 6) [15] was based on the the Offensive Language Identification Dataset (OLID) containing over 14,000 tweets. This SemEval task had three subtasks: discriminating between offensive and non-offensive posts, detecting the type of offensive content in a post and identifying the target of an offensive post. At GermEval, [16] there was a task to detect and classify hurtful, derogatory or obscene comments in the German language. Two sub-tasks were continued from their first edition, namely, a coarse-grained binary classification task and a fine-grained multi-class classification problem. As a novel sub-task, they introduced the binary classification of offensive tweets into explicit and implicit.

3. Dataset

The datasets for the tasks were provided by the organizers of the HASOC '20[2]. [17] The data consists of tweets from three languages: English, German and Hindi, and was annotated on two levels. The coarse annotation involved a binary classification task with the given tweet being marked as hate speech (HOF) or not (NOT). In the finer annotation, we differentiate between the types of hate speech and have four different formal classes:

1. **HATE**: This class contains tweets which highlight negative attributes or deficiencies of certain groups of individuals. This class includes hateful comments towards individuals

[2]https://competitions.codalab.org/competitions/26027

Language	NOT	HOF
English	1852	1856
German	1700	673
Hindi	2116	847

Table 2
Training set label distribution: Task 1

Language	NONE	HATE	OFFN	PRFN
English	1852	158	321	1377
German	1700	146	140	387
Hindi	2116	234	465	148

Table 3
Training set label distribution: Task 2

based on race, political opinion, sexual orientation, gender, social status, health condition, etc.
Example: "RT @Lubchansky: good to know rich people have always been dumb as shit https://t.co/otdmH0wquk"

2. **OFFN**: This class contains tweets which are degrading, dehumanizing or insulting towards an individual. It encompasses cases of threatening with violent acts.
Example: "By shitting yourself and taking the backdoor out, instead of fronting up to the public."
3. **PRFN**: This class contains tweets with explicit content, profane words or unacceptable language in the absence of insults and abuse. This typically concerns the usage of swearwords and cursing.
Example: "@HermesCxbin turn that shit off"
4. **NONE**: This class contains the tweets which do not fit into the above three classes i.e it does not contain instances of hate and offence.
Example: "@AskPlayStation I can't get the 14 days free trial please fix I don't have money for ps plus I need this."

In table 1, we list the data size in number of tweets, and in tables 2 and 3, we provide the number of instances of different classification labels.

4. Approach

In this section, we outline our approach towards solving the task at hand.

4.1. Preprocessing

We utilized the python libraries tweet-preprocessor[3] and ekphrasis[4] for tweet tokenization and hashtag segmentation respectively. For extracting English and German cleaned tweet texts, tweet-preprocessor's clean functionality was used. For Hindi tweets, we tokenized the tweet text on whitespaces and symbols including colons, commas and semicolons. This was followed by removal of hashtags, smileys, emojis, URLs, mentions, numbers and reserved words (such as @RT which indicates Retweets) to yield the pure Hindi text within the tweet.

4.2. Feature Engineering

In addition to the cleaned tweet, we utilize tweet-preprocessor to populate certain information fields which can act as features for our classifiers. We include the hashtag text which is segmented into meaningful tokens using the ekphrasis segmenter for the twitter corpus. We also save information such as URLs, name mentions such as '@derCarsti', quantitative values and smileys. We extract emojis which can be processed in two ways. We initially experimented with the emot[5] python library to obtain the textual description of a particular emoji. For example, ' 😅 ' maps to 'smiling face with open mouth & cold sweat' and ' 🐼 ' maps to 'panda'. We later chose to utilize emoji2vec [18] to obtain a semantic vector representing the particular emoji. The motivation behind this is as follows: the text describing the emoji's attributes might not capture all the pragmatics and the true sense of what the emoji signifies in reality. As a concrete example, consider ' 👅 ', the tongue emoji. The textual representation will not showcase the emoji's association with 'joking around, laughter and general goofiness' which is its real world implication. We expect emoji2vec to capture these kinds of associations.

4.3. Perspective API Features

We perform experiments with features extracted from the Perspective API. The API uses machine learning models to estimate various numerical metrics modeling the perceived impact which a post or a comment might have within a conversation. Right now, the Perspective API does not support Hindi natural language text in Devanagari script. Thus, our experiments are on German and English. On German text, the API provides scores which are real numbers between 0 and 1 for the following fields: 'toxicity', 'severe toxicity', 'identity attack', 'insult' and 'profanity and threat'. For English text, in addition to the fields for German, the API provides similar scores for the fields: 'sexually explicit', 'obscene' and 'toxicity fast' (which simply uses a faster model for computing toxicity levels on the back-end).

For both English and German tweets, we extract perspective API scores for all available fields using (a) the complete tweet as is, and (b) the extracted cleaned tweet text excluding emojis, smileys, URLs, mentions, numbers, hashtags and reserved words. Thus, we have 18 features for English tweets and 12 features for German tweets to work with.

We trained multi-layer perceptron classifiers for English and German using a concatenation of these features as the input vector. In addition to these classifiers trained in the monolingual

[3] https://github.com/s/preprocessor
[4] https://github.com/cbaziotis/ekphrasis
[5] https://github.com/NeelShah18/emot

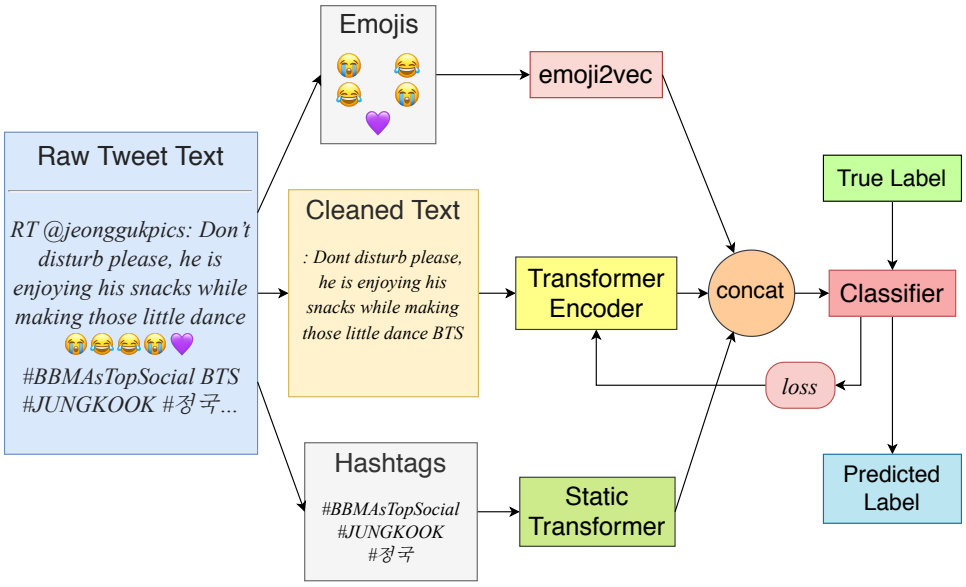

Figure 1: Model Overview

setting, we trained an English-German multilingual classifier using the 12 perspective API features which are common to English and German. The datapoints in the corresponding training sets were randomly shuffled and standardized. The same standardization values were used on the test set during inference. We tried out multiple training settings with different activation functions and optimization techniques. The best results with Perspective features are presented in Section 5.

4.4. Proposed Transformer-based Models

We leverage Transformer-based [1] masked language models to generate semantic embeddings for the cleaned tweet text. In addition to the cleaned tweet's embedding, we generate and utilize semantic vector representations for all the emojis and segmented hashtags available within the tweet. The segmented hash embeddings are generated using the same pre-trained Transformer model such that the text and hashtag embeddings are grounded in the same latent space. emoji2vec is used to create the emojis' semantic embeddings. The Transformer layers encoding the cleaned tweet text are updated during the fine-tuning process on the available training data. For classification, we use the concatenation of the cleaned tweet's embedding with the collective embedding vector for segmented hashtags and emojis.

We are required to encode a list of emojis & a list of segmented hashtags, both of which can be of variable lengths. Therefore, we average the vector representations of all the individual emojis or segmented hashtags as the case may be, to generate the centralised emoji or hashtag representation. This is simple, intuitive, and earlier work on averaging local word embeddings to generate global sentence embeddings [19] has showed that this yields a comprehensive vector representation for sentences. We assume the same to hold true for emojis and hashtags as well.

	Activation	Optimization	English		German	
			Task 1	Task 2	Task 1	Task 2
monolingual	identity	adam (early-stop)	**89.68**	**53.90**	75.40	41.84
	tanh	adam (early-stop)	88.93	47.07	**79.25**	43.00
multilingual	identity	sgd (adaptive LR)	88.82	47.02	72.89	38.86
	identity	adam (early-stop)	88.44	46.00	72.63	42.83
	tanh	sgd (adaptive LR)	87.69	44.86	75.38	38.80
	tanh	adam (early-stop)	87.95	46.03	76.68	**46.40**

Table 4
Perspective API Experiments (Best results highlighted in bold)

The concatenated feature-set is then passed to a two layer multi-layer perceptron (MLP). The loss from the classifier is propagated back through the cleaned tweet Transformer encoder during training. We experimented with XLM-RoBERTa (XLMR) [20] as our pre-trained Transformer in various training settings. XLM-RoBERTa has outperformed similar multilingual Transformer models such as mBERT(multilingual BERT) [2] and multilingual-distilBERT [21] on various downstream tasks. We therefore chose XLMR as our base Transformer model for the purpose of the shared task. A high level overview of our model flow is shown in figure 1.

For fine-tuning our XLMR Transformer weights, we perform learning rate scheduling based on the actual computed macro F1-scores on the validation split instead of using the validation loss. As opposed to simply using early-stopping to prevent overfitting, we consider the change in validation performance at the end of each training iteration. If the validation performance goes down across an iteration, we trace back to the previous model weights and scale down our learning rate. Training stops when the learning rate reaches a very small value ϵ[6]. Although expensive, this form of scheduling ensures that we maximize our Macro F1-score on the validation split. For further details on specific implementation nuances and choice of hyperparameters, refer to Section 6.

5. Results

In this section, we provide quantitative performance evaluations of our approaches on the provided testing-set, the evaluation metric used throughout being the macro F1-score.

In table 4, we present our study on usage of Perspective API features with a multi-layer perceptron classifier for English and German tasks. We notice that these features are able to provide respectable results on the hate and offensive content detection but cannot compete with the Transformer-based models when fine-grained classification is required. In the monolingual mode, our exhaustive grid search showed that the use of identity activation for English and tanh activation for German are the most effective MLP hidden layer activation settings. Table 4 lists the best activation functions and optimization techniques for particular (task, language) pairs. We observe that German Task 2 benefits from the multilingual mode and we attribute this

[6]Set to 1e-12 in our experiments.

Model	English		German		Hindi	
	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2
XLMR-freeze-mono	83.92	52.38	66.85	41.52	68.25	40.45
XLMR-freeze-multi	82.02	51.02	68.34	48.60	66.27	41.59
XLMR-adaptive	**90.29**	59.03	81.04	52.99	**75.40**	45.87
XLMR-tuned	90.05	**60.70**	**81.87**	**53.28**	74.29	**49.74**

Table 5
Performance of Proposed Transformer-based Models (Best results highlighted in bold)

	Features	English		German		Hindi	
		Task 1	Task 2	Task 1	Task 2	Task 1	Task 2
monolingual	cleaned text	83.27	49.12	69.23	39.12	**68.45**	45.18
	cleaned text + emoji	**83.60**	49.78	68.05	40.54	68.21	45.48
	cleaned text + hashtag	83.17	**53.60**	66.23	41.91	66.98	**50.08**
multilingual	cleaned text	80.47	47.88	71.07	46.66	64.84	44.39
	cleaned text + emoji	82.73	51.90	**72.73**	43.24	67.83	41.83
	cleaned text + hashtag	81.22	50.06	68.52	**47.31**	68.19	44.71

Table 6
Feature Selection Study (Best results highlighted in bold)

to the additional data from the English training examples which allow the model to generalize better. However, a drop in the English results is witnessed which might be due to the reduction in the number of available features.

In table 5, we present results using our proposed Transformer-based models. We present XLMR-freeze-mono and XLMR-freeze-multi as baselines in which we use the pre-trained XLM-RoBERTa Transformer weights without any fine-tuning[7]. Only the classifier head is trained in these models. We train six separate models for the three languages (two tasks per language) and report corresponding results in the monolingual mode. In multilingual mode, we only train two models on the aggregated training data for the two tasks and use that for inference across the three languages.

The models: XLMR-adaptive and XLMR-tuned use our proposed adaptive learn rate scheduling. In XLMR-tuned, the epsilon value of the Adam optimizer was set to 1e-7 as this experimental setting provided gains on the validation split in our hyper-parameter tuning phase. In both of these models, we jointly fine-tune the XLM-RoBERTa Transformer weights and the classifier head in a multilingual setting. Our proposed models significantly outperform baselines with frozen Transformer weights which is both intuitive and expected.

Finally, in table 6, we show results for a study on feature selection using pre-trained XLM-RoBERTa as the Transformer architecture for generating text embeddings. Note that our primary models including XLMR-freeze utilize all of the discussed features. Like XLMR-freeze,

[7]We used UKPLab's sentence-transformers library's pre-trained model: 'xlm-r-100langs-bert-base-nli-mean-tokens' for this task. The model is available at https://github.com/UKPLab/sentence-transformers.

the Transformer layers are frozen and not fine-tuned during the training process. The table is separated into monolingual and multilingual modes of training. Results are showed using different feature collections, namely, 'cleaned tweet text only', 'cleaned tweet + hashtags', and 'cleaned tweet + emojis' as inputs to the classifier. We observe a performance drop for English and Hindi and a considerable performance gain for German while moving from monolingual to multilingual training settings.

6. Experimental Details

We used Hugging Face's[8] implementation of XLM-RoBERTa in our proposed architecture. Our architectures using Transformer models with custom classification heads were implemented using pytorch[9]. We used Adam optimizer for training with an initial learning rate of 2e-5, dropout probability of 0.2 with other hyper-parameters set to their default values. We updated weights based on cross-entropy loss values. For studies with Perspective API Features and experiments where we do not fine-tune the Transformer weights, we used scikit-learn's [22] implementation of a multi-layer perceptron and UKPLab's sentence-transformers library [23] whenever applicable.

In our Perspective API experiments, we used deep multi-layer perceptrons with 12 and 9 hidden layers for the binary and multi-class classification modes respectively. Across all our experimental settings, we used a batch size of 200 with other hyper-parameter values set to default. We performed an exhaustive grid search for every multi-layer perceptron model varying the activation function, size of hidden layer, optimization algorithm and type of learning rate scheduling. We reported results using the grid search settings which performed the best on a 4-fold cross validation on the training set. Our experimentation code is publicly available at https://github.com/sayarghoshroy/Hate-Speech-Detection.

7. Conclusion

In this paper, we have leveraged the recent advances in large scale Transformer-based language model pre-training to build models for coarse detection and fine-grained classification of hateful and offensive content in social media posts. Our experiments showcase the utility and effectiveness of language models pre-trained with multi-lingual training objectives on a variety of languages. Our studies show the efficacy of Perspective API metrics by using them as standalone features for hate speech detection. Our best model utilized semantic embeddings for cleaned tweet text, emojis, and segmented hashtags as features, and a customized two-layer feed-forward neural network as the classifier. We further conducted a feature selection experiment to view the impact of individual features on the classification performance. We concluded that the usage of hashtags as well as emojis add valuable information to the classification head. We plan to further explore other novel methods of capturing social media text semantics as part of future work.

[8] https://huggingface.co/
[9] https://pytorch.org/

References

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

[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] I. Kwok, Y. Wang, Locate the hate: Detecting tweets against blacks, in: AAAI, 2013.

[4] W. Wang, L. Chen, K. Thirunarayan, A. P. Sheth, Cursing in english on twitter, in: Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, 2014, pp. 415–425.

[5] S. Themeli, Hate Speech Detection using different text representations in online user comments, Ph.D. thesis, 2018. doi:10.13140/RG.2.2.12991.25764.

[6] Y. Chen, Y. Zhou, S. Zhu, H. Xu, Detecting offensive language in social media to protect adolescent online safety, in: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing, IEEE, 2012, pp. 71–80.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[8] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[9] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[10] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[11] P. Badjatiya, S. Gupta, M. Gupta, V. Varma, Deep learning for hate speech detection in tweets, in: Proceedings of the 26th International Conference on World Wide Web Companion, WWW '17 Companion, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2017, p. 759–760. URL: https://doi.org/10.1145/3041021.3054223. doi:10.1145/3041021.3054223.

[12] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in neural information processing systems, 2015, pp. 649–657.

[13] Y. Mehdad, J. Tetreault, Do characters abuse more than words?, 2016, pp. 299–303. doi:10.18653/v1/W16-3638.

[14] R. Kumar, A. K. Ojha, M. Zampieri, S. Malmasi (Eds.), Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018. URL: https://www.aclweb.org/anthology/W18-4400.

[15] 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), in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA, 2019, pp. 75–86. URL: https://www.aclweb.org/anthology/S19-2010. doi:10.18653/v1/S19-2010.

[16] J. Struß, M. Siegel, J. Ruppenhofer, M. Wiegand, M. Klenner, Overview of germeval task 2, 2019 shared task on the identification of offensive language, 2019.
[17] 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.
[18] B. Eisner, T. Rocktäschel, I. Augenstein, M. Bosnjak, S. Riedel, emoji2vec: Learning emoji representations from their description, CoRR abs/1609.08359 (2016). URL: http://arxiv.org/abs/1609.08359.
[19] S. Arora, Y. Liang, T. Ma, A simple but tough-to-beat baseline for sentence embeddings (2016).
[20] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, 2020. arXiv:1911.02116.
[21] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. arXiv:1910.01108.
[22] 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.
[23] N. Reimers, I. Gurevych, Making monolingual sentence embeddings multilingual using knowledge distillation, arXiv preprint arXiv:2004.09813 (2020). URL: http://arxiv.org/abs/2004.09813.