Exploiting Contextualized Word Representations to Profile Haters on Twitter

Notebook for PAN at CLEF 2021

Tanise Ceron[1,2], Camilla Casula[1,2]

[1]University of Trento, Italy
[2]Fondazione Bruno Kessler, Italy

Abstract

In this paper, we present our submission to the Profiling Haters on Twitter shared task at PAN@CLEF2021. The task aims at analyzing Twitter feeds of users in two languages, English and Spanish, in order to determine whether these users spread hate speech on social media. For English, we propose an approach which exploits contextualized word embeddings and a statistical feature extraction method, in order to find words which are used in different contexts by haters and non-haters, and we use these words as features to train a classifier. For Spanish, on the other hand, we take advantage of BERT sequence representations, using the average of the sequence representations of all tweets from a user as a feature to train a model for classifying users into haters and non-haters.

Keywords

BERT, word embeddings, hate speech, statistical feature extraction

1. Introduction

The rise of social media in the past decade has undoubtedly changed interaction among people and made the world more inter-connected. It has provided a way for people to keep constantly in contact even when being far apart geographically, united people who have not seen each other for years or who had never met before, helped numerous volunteer associations to gather aid or recruit more volunteers, provided a place for entire communities with common interests to interact with one another and share content, resources and ideas, and the list of benefits continues relentlessly. However, on the flip side of the coin, the growing amounts of user-generated content online are tied to an increased presence of hateful content on social media. Content moderation online is therefore important to identify and limit the spread of hate speech.

The Profiling Haters on Twitter task [1] at PAN 2021 [2] aims at determining whether a user spreads hate speech based on their Twitter feed. This shared task tackles the problem of identifying hate spreaders from a multilingual perspective, including Twitter feeds in English and Spanish.

In this paper, we present our submission to the Profiling Haters on Twitter shared task, which consists of two different approaches. First, we propose a novel approach to hate speech detection for the English data set, which derives from the assumption that certain words are

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
taniseceron@gmail.com (T. Ceron); ccasula@fbk.eu (C. Casula)

used in different contexts by haters as opposed to non-haters. The idea is to exploit statistical feature selection techniques in order to find words whose embedding vectors extracted from BERT differ the most between classes, then use these words as features to train a classifier. The Spanish model, on the other hand, is inspired by text classification models, as it allows us to tackle the challenge of having a single representation for long sequences. Therefore, we build the features as though all tweets of a given user were a unique text, without losing information from any tweet. In order to do this, we use a Spanish pre-trained version of BERT for extracting a single vector representation of each tweet by a user. These representations are then averaged and fed into a classifier.

2. Related Work

The present work follows the definition of hate speech as described in the overview of this edition's shared task [1] and claimed by Nockleby [3] - it is defined as "any communication that disparages a person or a group on the basis of some characteristic, such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or others".

Most studies carried out on hate speech within natural language processing (NLP) so far have focused on the detection of hate speech in single messages. The singularity of this shared task lies in the fact that, instead, it focuses on the quite novel approach of classifying users who disseminate hateful messages (haters) and users who do no spread any type of hateful messages (non-haters) on Twitter. To the best of our knowledge, a similar task was proposed only once [4]. However, it is developed differently, given that the features of their model are based on the interaction among users and network metrics rather than linguistic features as proposed in the present work. User information has been used to boost the performance of hate speech detection in messages in other works as well [5, 6].

In the past years, many approaches have been proposed for the detection of hate speech in single messages extracted from various social media channels, such as Twitter [6, 7], Reddit [8], and YouTube [9]. A number of shared tasks have been organized on the topic, both from a monolingual [10, 11, 12, 13] and a multilingual perspective [14, 15]. They vary from more linear machine learning approaches with Naive Bayes [9], Logistic Regression [7], Support Vector Machine [16] to non-linear approaches fed with features from non-contextualized word embeddings [17] and the latest deep learning models consisting of contextualized word vector representations as features [18].

As in many other NLP and, more in general, supervised learning methods, feature selection is one of the most crucial parts of the task. In addition to this, the task of hate speech detection is particularly complex because messages can involve sarcasm, irony and neutral sentiments that are challenging for NLP systems to identify. In early models, as Schmidt and Wiegand [19] put it, simpler surface-level features such as n-gram and character n-grams have been implemented [20, 6, 7]. In addition to that, linguistic and lexical features have also been employed for this task, the former with the addition of part-of-speech or dependency information [21, 22] and the latter with terms that are related to hatred against a certain community or general profanities [9, 20]. Yet, other models have made use of features reliant on other common NLP tasks such as sentiment analysis [23].

Nobata et al. [20] experiment with features derived from static word embeddings with annotated data of comments on Yahoo! in three ways. Two of them consist of averaging the vector representation of all words in a comment derived from two types of word embeddings, they are the *pretrained* and *word2vec* models, both containing 200 dimensions. Their third approach is based on the representation of paragraph embeddings [24] following the work of Djuric et al. [25], who use the same approach for abusive language detection. In this case, every word of the comment is mapped to a matrix representing words, and every comment is mapped into a vector in a matrix of comments. Finally, words and comment vectors are concatenated forming a *single* representation of the comment. Besides the distributional semantic features, character and token n-grams, linguistic features (such as length of comment in tokens, average length of words, number of punctuation marks and so on), syntactic features, namely part-of-speech and dependency parsing relations, are also included in the model. The combination of all these features yield better results than the use of the *paragraph2vec* technique alone [25].

The latest classifiers for hate speech detection take advantage of models such as BERT, RoBERTA and other large multilingual language models [15]. They usually feed the sentence vector representation, the [CLS] token in BERT, into more recent deep learning architectures such as convolutional neural networks, recurrent neural networks and gated recurrent units and they reach very impressive results. In the last SemEval task for detection of offensive language, the best team reached a F1 score of 0.9204 and the other teams have mostly reached very similar performance in a tight competition.

Our model proposes to work on this line of features because of its potential to capture meaning beyond a restricted list of words, besides the great number of successful NLP applications that are based on non-contextualized vector representations of words, for instance GloVe [26] and word2vec [27], and more recently contextualized representations of text with Deep Bidirectional Transformers such as BERT [28]. The development of language models based on transformer mechanisms is an important milestone in advancements of NLP, given that it has improved the state of the art of many well-established NLP tasks. One of its greatest advantages is its capacity to encompass the representation of a text in a single vector. Secondly, the vector representation of each word is dynamic and contextualized, meaning that it is has the potential to adapt the embeddings of a word according to its context. Whereas our Spanish model benefits from the former advantage, the English model uses the latter in its favor.

3. Methods

Both the English and Spanish training sets are balanced, consisting of 100 haters and 100 non-haters. The dataset provided by the task organizers contains 200 tweets and a ground truth label for each user. In both datasets provided, user mentions, URLs, and hashtags have been replaced with tags in the form of *#HASHTAG#*. We remove all hash marks (#) while keeping the accompanying words (USER, URL, and HASHTAG).

We use different models to perform the task on English and on Spanish data. Both models exploit contextualized word representations and implement a support vector machine for the task of binary classification.

Both models were tested by the task organizers using the TIRA tool [29].

3.1. English

3.1.1. Features exploration

The idea underlying our English model is that haters and non-haters might use certain words in different contexts. Example (1) below shows a tweet found in the hater class which mentions the word *gun* in an aggressive circumstance, whereas tweet (2) illustrates an instance from the non-hater class and mentions the same word in a more political context. Thus, the feature selection for the model involves the identification of words and, in this case, of BERT embeddings that significantly vary from one class to another.

1. *This money can't fit in my pockets but I bet that gun fit.*
2. *New state laws for the new year: California limits gun rights, minimum wages increase #url... #url.*

To verify whether there are significant differences in the vector representation of words between the two classes, we first carry out an experimental coarse analysis with t-SNE [30], a technique of dimensionality reduction that is able to reduce the space to two dimensions so that it can be plotted and interpreted.

We first make a list of most frequent tokens by selecting the ones that occur at least 25 times in both classes. In total there are 788 of them. Note that they are BERT WordPiece tokens [31] taken from BertTokenizer [32] [1], meaning that tokens not correspondent to complete words are also included in the list. However, even though words are not complete, they should still have a rich contextualized vector representation, considering that BERT is able to distinguish the different contexts of split words as well [33]. Throughout the whole experiment, we use the uncased base version of BERT [34].

For the t-SNE analysis, we feed each tweet of a given $user_j$ of the class hater into the BERT model and retrieve the vector representation of a given token (t_i) present in the most frequent list. Then, we average all the vectors of t_i of $user_j$. More formally, let t_i be the token that occurs in a tweet {tw_1, tw_2,... tw_n} of a given $user_j$. Thus, the vector representation ($\vec{E}$) of t_i in $user_j$ is:

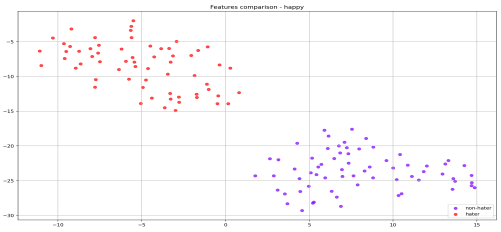
$$\vec{E}_{user_j}[t_i] = \frac{\sum_{n=1}^{N} tw_n[t_i]}{N} \tag{1}$$

where N is the number of occurrences of t_i in all tweets by $user_j$, and $\vec{tw}_n[t_i]$ is the vector of t_i in tw_n. We then repeat the same procedure for the non-hater class and reduce the dimensions. For example, $\vec{E}_{user}$[*gun*], which is a matrix of [Nx768], is reduced to a matrix with 2 components [Nx2]. Some of the results can be seen in Figure 1, where each dot represents a $\vec{E}_{user_j}[t_i]$.

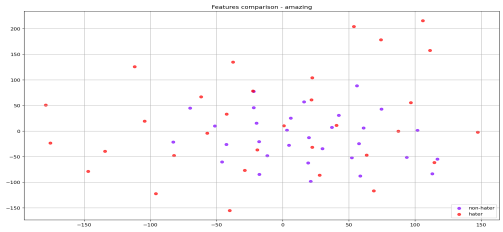
This coarse evaluation shows that some tokens form well-defined clusters between the two classes such as *happy* (Figure 1a) and *world* (Figure 1b). In contrast, others words like *amazing* (Figure 1c) and indeed even the word *gun* (Figure 1d) are sprawling and occupy overlapping spaces in both classes, suggesting that they do not have distinguishing vector representations.

Given the results of this coarse analysis with t-SNE, and considering that the reduction of vectors from 768 to 2 dimensions may cause the vector to lose a large amount of relevant information, we turn to a more statistical approach to select the words for our model.

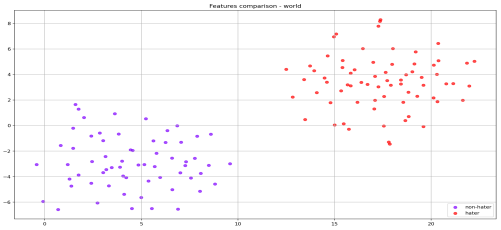
[1] https://huggingface.co/transformers/main_classes/tokenizer.html



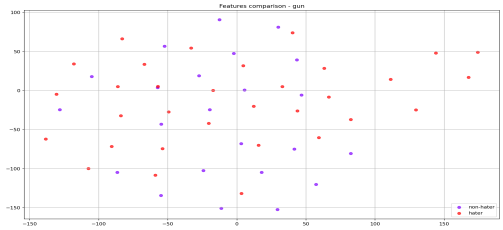
(a) Happy



(c) Amazing



(b) World



(d) Gun

Figure 1: t-SNE applied on words as a coarse analysis to check the difference in the semantic space of the hater and non-hater classes. t-SNE has a perplexity of 15, 2 components and 3500 iterations.

Instead of using predefined term lists, we employ a technique called *filter approach* for selecting the features (in our case the words) that most diverge between the two classes in terms of word embeddings. This technique requires two steps. First of all, a statistical test measures the difference in the vector representation of the tokens, and returns the p-value for the difference in vector between the two classes for each token. Then, a p-value (our threshold) is chosen, in order to pick the k most relevant features/tokens. In this study, we analysed the difference in vectors using the Kolmogorov-Smirnov (K-S) test. Biesiada and Duch [35] suggest that the K-S test helps in feature selection of high-dimensions and can significantly improve the performance of classifiers such as the one used here (SVM). The K-S test allows us to understand the maximum difference between the cumulative distribution of two random variables. Therefore, we assume that the more dissimilar the vectors are, as determined by a two-tailed K-S test, the easier it is for the classifier to distinguish between classes.

To start with, we retrieve the same vector representation for each user presented in Equation 1. After that, considering that in this case we want to have a single representation of a token t for each class, we average $\vec{E}_{user}[t_i]$ of all users to get the final representation of t_i, as in:

$$\vec{E}_{ks}[t_i] = \frac{\sum_{n=1}^{N} \vec{E}_{user}[t_i]}{N} \tag{2}$$

In this case N is the number of users that have at least one occurrence of the given t_i and we call the vector $\vec{E}_{ks}$ because it is used for the statistical test. We reach this point with two dictionaries, one for each class, with t_i as key and its corresponding $\vec{E}[t_i]$ as the value and are ready to run the K-S test. For example, the $\vec{E}[gun_{hater}]$ from the hater label as variable x and $\vec{E}[gun_{non-hater}]$ as variable y, letting the K-S test be the function $K - S_{two-tail}(x, y)$.

The results of the K-S test for each token are p-values very close to 1 for most tokens, since they are drawn from the same distribution. However, this is not a problem in our case, because we do not want to know whether they are statistically significant according to the confidence interval. The goal instead is to find out which $\vec{E}_{ks}[t_i]$ are lower compared to others, meaning that the $\vec{E}_{ks}[t_i]$ between classes are more dissimilar.

3.1.2. Model implementation

Now that we have the p-values for each token in our most frequent list, we must decide on a threshold that will select the number of relevant features to be fed into the classifier. This is done by various runs with different thresholds in the model. That is, we pick a p-value, get a single vector representation that is the average of all the $\vec{E}_{user}$ with t under the p-value of the K-S test, such as:

$$\vec{E}_{feat}[user_j] = \frac{\sum_{n=1}^{N} \vec{E}_{user_j}[t < threshold]}{N} \tag{3}$$

The embeddings is called *feat* because it is the feature representation of each user. $\vec{E}_{feat}[\textit{user}]$ is fed to the classifier. The performance is evaluated in terms of accuracy with 5-fold cross-validation. We finally choose the set of features that results in the model's best performance. The threshold selected in our submission is 0.998 and the set of tokens consists in a total of 394 tokens/features for the model. As a matter of fact, some of them can be very relevant semantically in the context of hate speech detection such as *war*, *liberal*, *black*, *woman*, *violence*, *racism*, *bitch* and so forth (all the tokens are presented in Appendix A - List of wordpieces used in the English model).

After having selected the features, we also try different layers of representations from BERT's outputs given that it is has been observed that each of the 12 layers capture different features of the input text [36]. More precisely, we experiment with the last three layers because they seem to be the ones that encapsulate more context-specific representations [37]. Hence, we run the classifier with feature vector representations from the 10th, 11th and 12th layer to see which performs the best. The 12th layer shows to produce better results even though the different in performance between one layer and another is not statistically significant.

As a final step, we add to the features the averaged CLS tokens of each tweet because we notice that even though the set of tokens is large, there are some users from the test set who do not contain any of those tokens. Again, we run tests to see which layer of the CLS token is more advantageous to the model and choose the 12th layer. The experiments are conducted with two kernels of the support vector machine, the radial basis function (rbf) and the polynomial kernel. We utilize Bayesian optimization technique [38] for finding the best hyper-parameters in order to spare some time and computational power in executing the traditional grid search approach. The best performing model is the rbf (C≈14.0749, gamma≈0.0095).

3.2. Spanish

During the experimental phase, we tested the same approach used in the English data set on Spanish as well. However, in our final submission, we opted for a simpler model, which in our

experiments worked better on the Spanish data.

This model follows a more straightforward approach inspired by text classification [36] with BERT representations. It is based on text classification systems because all tweets of each user are treated like a long text. Besides that, given that the 200 tweets are longer than 512 tokens, such as in the case of text classification, in order to have a single representation of the whole text, we use the strategy of averaging the sequence representation of every tweet of the user. For that, we use the uncased and base pre-trained Spanish version of BERT called BETO [2] [39] throughout the training and testing of the Spanish data set.

After pre-processing, every tweet {tw_1, tw_2,... tw_n} of a given user is fed into BETO. Then, we extract the vector representation of the CLS token, which encapsulates a single representation of the whole sequence. Lastly, we average these vectors to create the feature representation of each user such as in:

$$\vec{E}_{feat}[user_j] = \frac{\sum_{n=1}^{N} \vec{tw}_n[cls\_token]}{N} \quad (4)$$

Where N is always 200 given that this is a fixed number of tweets in $user_j$. The $\vec{E}_{feat}$ is fed into the support vector machine.

We also experimented with summing the CLS tokens and found that results are very similar, given that there is no confound with the frequency of tokens. Similarly to the English model, we test the last three layers of the CLS token in rbf and polynomial functions with Bayesian optimization, and verify that the 11th layer trained on the polynomial kernel (C≈7.3588, gamma≈0.0285, degree≈1.2859) and the 10th layer trained on rbf give the best result in the 5 fold cross-validation, so we submitted both for the shared task, and indeed they have even returned the same labels for the test set.

4. Results

Table 1
The 5-fold cross-validation results for the English model. Our final submission model is the underlined model.

Kernel	Layer	Feature (K-S test)	Additional feature	Acc. (train set)	Acc. (test set)
rbf	11th	210 tokens	CLS token	74%	69%
rbf	12th	394 tokens	CLS token	72%	73%
rbf	10th	210 tokens	CLS token	71%	-
rbf	10th	517 tokens	CLS token	70,5%	-

Table 1 shows that the best accuracy in the training set in English is reached by the model with fewer features (210 tokens) compared to the second place. However, a paired t-test with the results of the 5 fold CV showed that the first and second model are not statistically significantly different in the training set (p-value=0.1998). The results of the test test, on the other hand, differ considerably from one another with the 394-token model reaching 4 points better accuracy. One reason for the difference in classification of the test set is that a broader range of tokens included in the features can enhance the performance on unseen data.

[2] https://github.com/dccuchile/beto

Alternatively to averaging, we try to sum the vectors based on the idea that the frequency with which words occur in the tweets may help the classifier to discriminate better between classes. However, despite having indeed performed better (accuracy 7% higher that our final submission model) in the training set overall, the classification in the test set was overwhelmingly imbalanced with 97 haters out of 100 users and the accuracy was also very imbalanced within the 5 fold cross-validation, showing that it did not generalize well in all folds. It suggests, though, that the classifier learns from the frequency and that there should be a similar number of occurrences for a reasonable performance.

For what concerns the Spanish model, even though we chose the model that extracts the CLS token representation from the 11th layer, the three models actually perform similarly in terms of accuracy as seen in Table 2. The first and second model have even returned the same labels for the classification.

Table 2
The 5-fold cross-validation results for the Spanish model. Our final submission model is the underlined model.

Kernel	Layer	Feature	Acc. (training set)	Acc. (test set)
poly	11th	CLS token	84%	80%
rbf	10th	CLS token	84%	80%
rbf	12th	CLS token	82%	-

Moreover, we have attempted to apply same approach we used for the English data set, but the results in the training set drop considerably reaching lower performance than with the CLS token approach. One hypothesis for the difference in results could be related to the corpora on which the pre-trained language models are trained. It might be that the language used in the Spanish tweets is more similar to the language in the corpora used for training BETO compared to the English data set and BERT, which would help encompassing the meaning and context of the dataset more easily, therefore prompting better results. Nonetheless, it is difficult to know precisely the reason why they perform so differently, because of the lack of interpretability of these large language models.

In terms of layer selection, we observed that both models perform quite similarly when trained in each of the three last layers, suggesting that they have very similar representations of tokens.

5. Conclusion

We present two novel approaches to profile haters on Twitter. The English approach relies on the idea that a set of words can be used in different contexts by the hater and non-hater users. The state-of-the-art language model BERT is adopted to capture the contextualized embeddings of tokens. Then, the difference of the vector representation in both classes is measured through the K-S statistical test. And finally, the relevant features are chosen by feeding a set of tokens from different thresholds of the test into the support vector machine.

In contrast, we have seen that the same approach does not work as well for the Spanish model. Therefore, inspired by text classification methods, we use the averaged vector representation of

all CLS tokens from every tweet of each user as input for the support vector machine. Despite being a simpler model, it yields impressive results considering the amount of training data available.

As a future step, it would be interesting to test the same models with more training data to check whether it boosts their performance, and perhaps replace the SVM approach with a deep learning model. In addition, in this shared task we only process textual information, but in a real scenario other features related to metadata could be included to have more informative and characteristic features, which may improve classification. Lastly, more related to the English model, other types of statistical tests might be experimented as well, in order to distinguish better features for the model. Otherwise, after the statistical test, the model could be trained iteratively with the ablation technique to select best performing features among the ones already selected by the threshold.

References

[1] F. Rangel, P. Rosso, G. L. D. L. P. Sarracén, E. Fersini, B. Chulvi, Profiling Hate Speech Spreaders on Twitter Task at PAN 2021, in: CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.

[2] J. Bevendorff, B. Chulvi, G. L. D. L. P. Sarracén, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, , E. Zangerle, Overview of PAN 2021: Authorship Verification,Profiling Hate Speech Spreaders on Twitter,and Style Change Detection, in: 12th International Conference of the CLEF Association (CLEF 2021), Springer, 2021.

[3] J. T. Nockleby, "Hate Speech", Encyclopedia of the American Constitution, ed. Leonard W. Levy and Kenneth L. Karst, vol. 3. (2nd ed.), Detroit: Macmillan Reference US. Cited in "Library 2.0 and the Problem of Hate Speech," by Margaret Brown-Sica and Jeffrey Beall, Electronic Journal of Academic and Special Librarianship, vol. 9 no. 2 (Summer 2008), 2000.

[4] M. Ribeiro, P. Calais, Y. Santos, V. Almeida, W. Meira Jr, Characterizing and detecting hateful users on twitter, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 12, 2018.

[5] P. Mishra, M. Del Tredici, H. Yannakoudakis, E. Shutova, Author profiling for abuse detection, in: Proceedings of the 27th international conference on computational linguistics, 2018, pp. 1088–1098.

[6] Z. Waseem, D. Hovy, Hateful symbols or hateful people? predictive features for hate speech detection on twitter, in: Proceedings of the NAACL student research workshop, 2016, pp. 88–93.

[7] 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, volume 11, 2017.

[8] A. Olteanu, C. Castillo, J. Boy, K. Varshney, The effect of extremist violence on hateful speech online, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 12, 2018.

[9] P. Burnap, M. L. Williams, Cyber hate speech on twitter: An application of machine

classification and statistical modeling for policy and decision making, Policy & internet 7 (2015) 223–242.

[10] 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, 2019, pp. 75–86.

[11] C. Bosco, F. Dell'Orletta, F. Poletto, M. Sanguinetti, M. Tesconi, Overview of the evalita 2018 hate speech detection task, in: EVALITA@CLiC-it, 2018.

[12] M. Sanguinetti, G. Comandini, E. Nuovo, S. Frenda, M. Stranisci, C. Bosco, T. Caselli, V. Patti, I. Russo, Haspeede 2 @ evalita2020: Overview of the evalita 2020 hate speech detection task, 2020.

[13] 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.

[14] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, M. Sanguinetti, SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA, 2019, pp. 54–63. URL: https://www.aclweb.org/anthology/S19-2007. doi:10.18653/v1/S19-2007.

[15] 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), in: Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, 2020, pp. 1425–1447. URL: https://www.aclweb.org/anthology/2020.semeval-1.188.

[16] S. Malmasi, M. Zampieri, Challenges in discriminating profanity from hate speech, Journal of Experimental & Theoretical Artificial Intelligence 30 (2018) 187–202.

[17] P. Mishra, H. Yannakoudakis, E. Shutova, Neural character-based composition models for abuse detection, arXiv preprint arXiv:1809.00378 (2018).

[18] M. Mozafari, R. Farahbakhsh, N. Crespi, A bert-based transfer learning approach for hate speech detection in online social media, in: International Conference on Complex Networks and Their Applications, Springer, 2019, pp. 928–940.

[19] 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.

[20] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, Abusive language detection in online user content, in: Proceedings of the 25th international conference on world wide web, 2016, pp. 145–153.

[21] 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.

[22] 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.

[23] N. D. Gitari, Z. Zuping, H. Damien, J. Long, A lexicon-based approach for hate speech detection, International Journal of Multimedia and Ubiquitous Engineering 10 (2015) 215–230.

[24] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International conference on machine learning, PMLR, 2014, pp. 1188–1196.
[25] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, N. Bhamidipati, Hate speech detection with comment embeddings, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 29–30.
[26] 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.
[27] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).
[28] 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).
[29] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), Information Retrieval Evaluation in a Changing World, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1_5.
[30] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (2008).
[31] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, Google's neural machine translation system: Bridging the gap between human and machine translation, CoRR abs/1609.08144 (2016). URL: http://arxiv.org/abs/1609.08144.
[32] T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.
[33] K. Zhou, K. Ethayarajh, D. Jurafsky, Frequency-based distortions in contextualized word embeddings, arXiv preprint arXiv:2104.08465 (2021).
[34] I. Turc, M.-W. Chang, K. Lee, K. Toutanova, Well-read students learn better: On the importance of pre-training compact models, arXiv preprint arXiv:1908.08962v2 (2019).
[35] J. Biesiada, W. Duch, Feature selection for high-dimensional data: A kolmogorov-smirnov correlation-based filter, in: Computer Recognition Systems, Springer, 2005, pp. 95–103.
[36] C. Sun, X. Qiu, Y. Xu, X. Huang, How to fine-tune bert for text classification?, in: China National Conference on Chinese Computational Linguistics, Springer, 2019, pp. 194–206.
[37] K. Ethayarajh, How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings, arXiv preprint arXiv:1909.00512 (2019).
[38] F. Nogueira, Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL: https://github.com/fmfn/BayesianOptimization.
[39] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, J. Pérez, Spanish pre-trained bert model and evaluation data, in: PML4DC at ICLR 2020, 2020.

Appendix A - List of wordpieces used in the English model

cls_token, act, actually, ad, age, ah, air, al, almost, also, always, americans, amp, anti, around, attack, automatically, away, b, behind, better, big, bitch, black, block, body, border, boy, br, break, bro, buy, ca, california, call, cannot, car, care, change, check, checked, children, christmas, city, class, close, cnn, co, con, congress, control, cr, crazy, crime, cu, cut, da, days, de, dem, democrat, democrats, deserve, di, die, different, donald, drop, dude, dumb, e, election, else, em, end, energy, even, evidence, ex, f, fact, facts, fake, family, far, fast, feeling, fight, find, fine, folks, following, food, forever, fox, fr, friend, friends, full, g, ga, gas, gave, george, get, getting, give, glad, gone, great, guy, guys, h, ha, hair, half, hands, happen, happy, hard, hash, hell, help, high, history, hit, ho, hold, hot, hu, human, id, idea, im, imagine, imp, important, ins, interesting, j, k, keep, kid, kids, kind, last, late, law, less, let, liberal, listen, live, lives, living, lo, longer, look, lord, lot, mad, made, make, makes, mark, mask, matter, may, men, military, mine, minutes, miss, mom, month, months, mother, move, mr, ms, mu, n, ne, never, ni, night, obama, ok, okay, old, om, ones, open, order, others, p, paid, pan, past, pe, per, place, plan, play, playing, please, po, point, poor, post, power, pp, pray, press, put, question, r, race, racism, racist, rape, rather, read, ready, reason, red, republican, rest, right, room, sad, safe, save, say, saying, sc, second, see, seems, seen, self, send, sense, share, shit, shot, show, shut, side, sign, single, sit, sm, smoke, social, someone, song, soon, speak, special, stand, start, state, stay, step, stop, story, straight, strong, stuff, suck, sure, system, take, taking, team, test, thanks, thing, things, three, ti, time, times, took, top, tried, trip, try, trying, twitter, type, un, understand, user, using, va, vaccine, via, vibe, video, violence, vote, voted, voting, vs, wall, want, wanted, war, water, wear, wearing, whatever, white, whole, win, wish, wit, woman, wonder, word, world, worse, worst, would, wrong, x, ye, yeah, year, yes, yesterday, yet, yo, youtube, ##aa, ##al, ##c, ##ce, ##ck, ##d, ##e, ##ea, ##er, ##es, ##f, ##fs, ##ful, ##gga, ##h, ##ha, ##i, ##ie, ##ies, ##in, ##k, ##llo, ##n, ##na, ##o, ##ot, ##p, ##r, ##rs, ##ss, ##t, ##tf, ##ting, ##v, ##w, ##wed, ##wee, ##x, ##y, ##z, 0, 000, 1, 19, 20, 2021, 3, 30, 4, 5, 50, 6, 7, 8, 9