hate speech classification project

Andrea Beretta

Andrea.beretta8@studenti.unimi.it, 27201A, Data Science for Economics, Università degli Studi di Milano.

Abstract

The goal of this project is to develop a model that classifies text content as hate speech, offensive speech, or neither, while also providing a lexical explanation for the classifier's decisions. Through solid text preprocessing, the model achieves good performance when compared to existing results. However, it still lacks robust lexical explainability.

Keywords: Hate speech, NLP, Machine Learning, Lexical explainability

1 Introduction

There is no universally agreed-upon definition of hate speech, which complicates detection for both humans and machines. However, those invested in the issue, such as social platform owners or policymakers, attempt to address this challenge by proposing their own definitions. According to the authors of the paper from which the dataset used in this project is sourced, hate speech is defined as language that expresses hatred towards a specific group or is meant to be derogatory, humiliating, or insulting to its members. [1].

In the relevant literature, it is noted that achieving strong results with traditional machine learning and deep learning is challenging, as even top-performing models often show their weaknesses when tasked with generalizing to unseen data. Beyond the inherent complexity of handling and accurately processing text data, there are frequently issues related to the methodologies employed in research papers and biases within the datasets. [2]. There is also a problem of explaining the reasons behind the choice of a classifier and there seems to be a trade-off between performance and explicability [3].

In this project, a popular data set is used [1] and a classifier is built to be able to distinguish among hate speech, offensive or neither of the two. The data are processed following precise indications [4] in order to extract as more useful information from the text as possible. Then the feature extraction is performed with the TF-IDF method and data are passed to a XGBoost classifier that is able to achieve better results overall for the data labeled as hate speech than the best model of the paper of the data set, with some drawbacks [1]. Finally, the SHAP library is used to study which is the relevant terminology for the choice of the classifier. The outcome is not amusing but still shows some interesting insights.

2 Research question and methodology

The goal of the project is to build a classifier that is able to correctly classify data labeled as hate speech, offensive or neither of the two. Furthermore, the aim is also to study the terminology related to the three categories in order to determine relevant aspects in the hate speech and especially what makes it different from offensive con-tent.

Once chosen and imported the data set, text data are subjected to a thorough prepro-cessing that makes use of various techniques. They are split in training and test set and, after undersampling the majority class, feature extraction is performed through TF-IDF method. Then the chosen model, an XGBoost, is trained and tested with unseen data. Thanks to the SHAP library, it is possible to extract the most relevant terminology and to try to draw some conclusions about the peculiarities of the hate speech.

Formally, given a string t, the problem is to build a classifier X such that X(t) is either hate speech, offensive speech or neither of the two with the minimum number of misclassifications with respect to the assigned label, in particular regarding the hate class. Furthermore, the problem is also to extract the most relevant substrings of t for the choice of the classifier among the three classes.

3 Experimental Results

In order to find the suitable data set for hate speech detection, a good start could be this website that provides a list of available choices in different languages and for each gives additional useful information. Furthermore, there is a summary paper that describes the practical steps that should be followed for choosing the right one in order to increase the quality of the trained model [5]. As already mentioned, the chosen resource is a very popular one, often used or cited in other papers. It can be found in this GitHub repository.

As described in the related paper [1], making use of an English lexicon of hate speech, the authors retrieve more than 24 thousand tweets with the Twitter API. Afterwards, they label each tweet as one of the three categories according to their definition. The dataset, as used in this project, is therefore composed of two columns:

1. class: an integer that is associated with one of the three classes: 0 with hate speech, 1 with offensive speech or 2 with neither of the two. It is easy to discover that

there is a strong class imbalance, as the occurences of the classes are respectively 1430, 19190 and 4163;

2. tweet: a string retrieved as described above from Twitter.

Following some Exploratory Data Analysis, the crucial step of text preprocessing is carried out. There are various techniques available, some of which serve as alternatives to one another. In this project, I adhere to the guidelines of a specific paper that provides valuable insights into the impact of each preprocessing technique on tweets, particularly concerning the quality of classification results for hate speech detection. [4]. I try to reason about the empirical results obtained in it by including or not a technique and if in my particular setting a certain choice could fit.
Below the list of techniques taken into account in the precise order:

- lowercase conversion: converting all the characters to lowercase. It is considered one of the basic processes as it reduces variability of the words and it is therefore implemented also in this project.
- hashtags handling: Hashtags are very common in tweets, and thus, they must be handled appropriately. They can be either removed, tokenized (converting them into a special token, like $HASHTAG$), or hashtag segmentation can be applied (breaking the hashtag into its constituent tokens and removing the # symbol). In this case, the decision is to remove them, as they are considered unnecessary for the analysis.
- URLs handling: A URL is a reference or address used to access resources on the internet. In the context of tweets, users may occasionally include URLs. These can either be removed or tokenized similarly to hashtags. The decision here is to remove them, as they are considered superfluous and not essential for the analysis.superfluous.
- mentions handling: a tweet may contain the @ symbol followed by an username: this is a mention and it is used to catch the attention of another user on the tweet. Again, mentions can be removed or tokenized. The choice is to tokenize them as SPECIALTOKEN because this feature may be related with one particular class, maybe hate speech, as it can be used to target someone.
- digits handling: there are three ways to deal with digits: removing, tokenizing or converting them to words. The choice is the latter as digits could be of interest for the kind of speech.
- punctuation and special characters removal: this is considered a basic technique, as all of the removed does not bring any semantic significance and may cause only an increase of noise. It is worth to mention that also emojis and emoticons are removed, whereas one may decide to tokenize or convert them to words.
- removing character repetitions: this technique consists of shortening all the sequences of the same characters longer than two to one, as in English any such spelling is incorrect, whereas words with the same character repeated twice are allowed because they are a normal part of the language. It is worth to say that this means that any misspelling such as soo instead of so is kept as it is. This is done because I think that such cases are rare and so should not affect deeply the performance of the model, whereas there may be a thorough and computationally expensive work to identify and remove them.

- expanding contractions: as in colloquial English it is common to observe contractions, it may be useful to replace them with the full words in order to reduce the noise, as ’re and are mean the same.
- tokenization: this might not even be considered a preprocessing technique like the others as it is mandatory in Natural Language Processing. It is listed especially to signal that it is performed at this point.
- stopwords removal: This technique aims to remove very common words in English, such as "the" or "on," as their high frequency tends to make them noisy. While the paper mentioned above suggests not removing them because they can be significant for the model, particularly since tweets are usually short, i decided to remove them. This decision is based on two reasons: first, it is unclear how these words could be related to any form of speech, and second, I do not want them to appear in the most relevant terminology once the model is tested, as they would add noise and reduce interpretability.
- stemming: lastly, two basic techniques usually performed are lemmatization and stemming. They are generally alternatives to each other. Lemmatization is more precise, as it reduces the words to their lemma, but it is also more computationally intensive, whereas stemming is approximate but faster. The choice is to perform stemming as, according to the paper mentioned above, there is not a clear best option between the two in a case like this.

Once the text preprocessing part is completed, data can be prepared for performing Machine Learning. The first thing to do is splitting train and test set. Only once this is done, undersampling can be performed on the training set in order to deal with class imbalance [2]. Regarding the exact number of instances to drop, I experimented with percentages between 75% and 85%, as within this range, the majority class becomes roughly as prevalent as the second most frequent class. By examining the number of columns in the output matrix—representing the tokens in the vocabulary—and the distribution of word lengths, it appears that any percentage within this range does not significantly impact the performance of the TF-IDF method.
The next step is precisely applying the TF-IDF method and some considerations made about this choice are noteworthy. First of all, it is a common approach for extracting feature in hate speech detection [6]. Secondly, TF-IDF in combination with XGBoost seems to obtain very solid results, comparable to transformers or Deep Learning approaches, without their complexity that usually requires time and com-putational resources [2][7]. Thirdly, it is crucial to carefully apply the method. TF-IDF should not be performed on the entire dataset, as this would lead to data leakage, with the model being trained on features extracted from the test set as well. This compromises the validity of the model's evaluation and can lead to overly optimistic performance estimates. [2]. On the other hand, if the method is applied separately to the two sets, the result reflects not only the model's performance but also the effectiveness of the method itself. Additionally, since the test set is smaller than the training set, the estimate would likely be less accurate. The solution is to apply the method to the training set, then use the resulting vocabulary and IDF estimates for the test set to ensure a consistent outcome. Lastly, when instantiating the method, I chose to remove tokens from the TF-IDF matrix that appear in fewer than 0.1% of the tweets, known as the cut-off, to reduce noise.

Once the method is applied and the corresponding matrix is computed, the next step is to instantiate the model. I create a custom function to pass to XGBoost, aiming to maximize the F1 Score for the hate speech class, as I want the model to be especially effective at detecting this type of speech. The F1 Score is the harmonic mean of precision (the proportion of true positive predictions out of all instances predicted as positive) and recall (the proportion of true positive predictions out of all actual positive instances). I choose to use it because I believe it provides a more reliable measure of the model's performance compared to precision and recall alone.

After all this work, finally the model is trained and tested. In order to be able to compare the results more clearly, data are processed and plotted in a confusion matrix in the same way as it is done in the paper (1 and 2). Each row sums up to 1 and each value is the percentage of the predictions made for each of the three classes on the total of instances of the true class. This means that the recall scores for each label are represented on the main diagonal [1].

Finally, the last step is to extract the relevant terminology with the library SHAP. This step does not require any great effort as it is only needed to pass the model to the built-in functions of the library. Below the plots for each class are reported (3, 4 and 5). The words at the top are the most influential for the choice of the classifier.

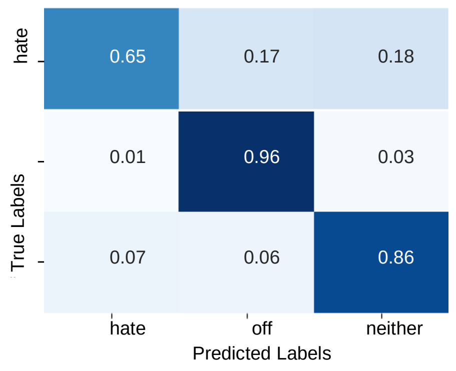

Fig. 1 Confusion Matrix of the project

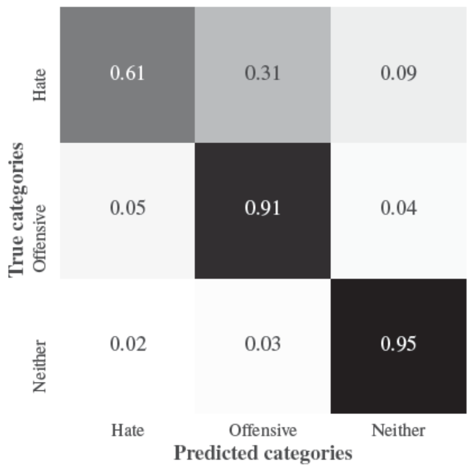

Fig. 2 Confusion Matrix of the paper

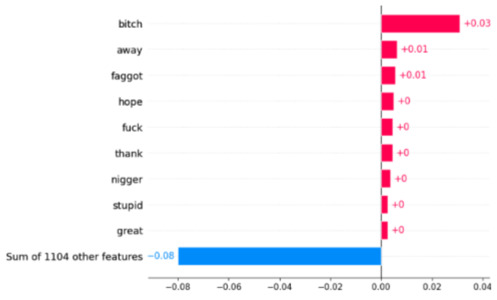

Fig. 3 Relevant terminology for hate speech

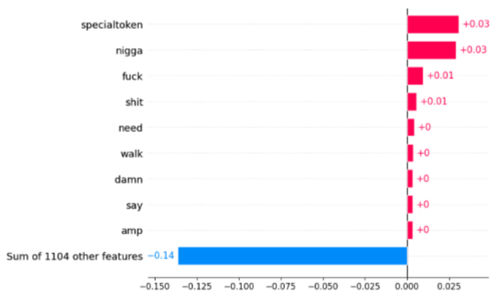

Fig. 4 Relevant terminology for offensive speech

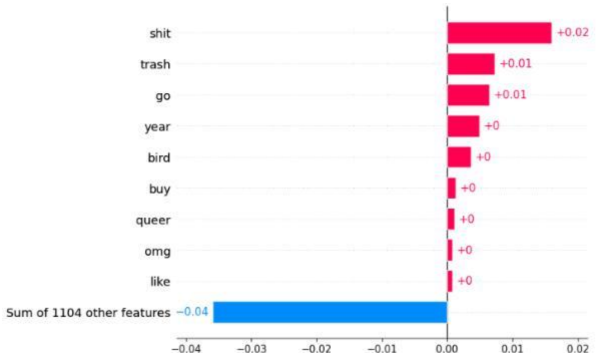

Fig. 5 Relevant terminology for normal speech

4 Concluding Remarks

With the project overview now complete, we can critically discuss the results. Compared to the best model reported in the paper [1], the overall performance of the XGBoost model is slightly lower. However, it demonstrates improved performance in the hate speech category, with higher precision (0.46 vs. 0.44) and recall (0.65 vs. 0.61) scores. From the above plots 1 and 2, it is possible to notice that the cost for the better results of my model for the class of the hate speech is that the number of misclassifications between the two furthest classes, hate and neither, is higher. This indicates that, while XGBoost generally performs better at accurately classifying hate speech, it struggles to fully grasp the hierarchy among the three classes. If it did, the results would more closely resemble those of the other matrix. This poses a problem, as not all misclassifications carry the same weight; errors between closely related classes are easier to address and explain. For the hate speech class, aside from some noisy words (e.g., "away" or "hope"), there are, as expected, harsh insults, particularly those related to racism and homophobia, along with vulgar expressions. This alignment is encouraging, as it corresponds with the definition outlined at the beginning of this paper. A special mention is needed for b*tch: according to the classifier,

it denotes strongly a speech as hateful. While from a strict semantic perspective this is accurate, as the word is highly misogynistic, it is frequently used in colloquial contexts today as a casual way to refer to others. Therefore, I would not expect it to be strongly associated with hate speech. Regarding the offensive class, in addition to some common words, there are three noteworthy insights. One is the dichotomy of n*gger and n*gga, that in my opinion are correctly associated by the model with the related classes: the first is a very harsh insult, the second, even if it is an insult, is commonly used by African Americans to refer to other components of the same community. The second is that there are two vulgar expressions, sh*t and f*ck, but no targeting insults. This is in line with the definition as one key factor that differen-tiates the hate speech from the offensive is the presence of a specific target group to insult. The third and last insight is that mentions are associated with offensive speech (specialtoken in 4). For the class of normal speech there is a vulgar expression, sh*t, that is not always associated with the purpose of offending, whereas it is not present f*ck, that usually has that purpose. Another noteworthy consideration concerns the word "queer": originally used as an insult, its meaning has evolved and it is now often used as a standard term to refer to certain members of the LGBT+ community. However, I would not anticipate it appearing in the relevant terminology for neutral or normal speech.
Overall, despite some noise affecting the results, certain aspects of hate speech can still be identified. Generally, it is easier to associate hate speech with harsh insults targeting specific groups, such as Black people, women, or the LGBT+ community. In contrast, more general insults or swear words are often linked to offensive speech, making their presence less indicative of hateful content. This obviously makes the detection more difficult. Furthermore, the presence of a swear word does not imply that a speech cannot be a normal one, as for the case of sh*t. From these considerations, one could design a lexicon pyramid for hate speech, with terms like "f*ggot" or "n*gger" at the top, representing the most explicit hate speech, and words like "sh*t" at the bottom, where the intensity of hate diminishes. It's crucial to remember that a word can be associated with multiple types of speech.
For future work, aside from improving model performance, three specific areas could be explored further:

1. definition of hate speech: I believe the primary challenge is the lack of a unanimous consensus on the definition of hate speech. This leads to results that are less generalizable across a broader framework and less comparable between studies, as researchers often develop their own definitions as a basis for their work. Establishing a unified definition, at least within the scientific community, would be essential for improving consistency and comparability in future research.
2. user identifier: as explained in [2], hate speech generation is typically attributed to a small number of individuals, but this is not reflected in most datasets, leading to a user bias. This bias often contributes to the overfitting observed in the literature. One potential solution is to include user identifiers, which could help mitigate this bias. It would be valuable to explore this topic further to understand how user-specific information could improve model performance and generalizability.

References

[1] Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. In: Proceedings of the International AAAI Conference on Web and Social Media. 11(1), 512–515 (2017)

[2] Arango, A., P´erez, J., Poblete, B.: Hate speech detection is not as easy as you may think: A closer look at model validation. In: Proceedings of the 42nd international acm sigir conference on research and development in information retrieval, 45–54 (2019, July)

[3] Matthew, B., Saha, P., Yimam, S.M., Biemann, C., Goyal, P., Mukherjee, A.: Hatexplain: A benchmark dataset for explainable hate speech detection. In: Proceedings of the AAAI Conference on Artificial Intelligence 35(17), 14867–14875 (2021, May)

[4] Glazkova, A.: A comparison of text preprocessing techniques for hate and offensive speech detection in twitter. Soc. Netw. Anal. Min. 13(155) (2023) https://doi. org/10.1007/s13278-023-01156-y

[5] Vidgen, B., Derczynski, L.: Directions in abusive language training data, a systematic review: Garbage in, garbage out. PLoS ONE 15(12) (2020) https://doi.org/10.1371/journal.pone.0243300

[6] Fortuna, P., Nunes, S..: A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR) 51(4), 85 (2018)

[7] Malik, J.S., Qiao, H., Pang, G., Hengel, A.V.D.: Deep learning for hate speech detection: a comparative study (2022) https://doi.org/arXivpreprintarXiv:2202.09517