hate speech classification project

Andrea Beretta

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

Abstract

The aim of this project is to create a model to classify text contents as hate speech, offensive or neither of the two and to give a lexical explanation with respect to the decision of the classifier. Thanks to a solid text preprocessing, the model is able to achieve a good performance considering the results in the literature, but still lacks robust lexical explicability.

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 pertinent literature it is observed that achieving good results with traditional machine learning and deep learning is tough as usually best-in-class models reveal their weaknesses when they are asked to generalize on unseen data. Beyond the diffi-culty per se of handling and correctly processing text data, often there are also issues related to the methodology followed in the papers and data set biases [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 the presented 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.

After some Exploratory Data Analysis, the fundamental part of text preprocessing is performed. There are many techniques that can be chosen and some are alternatives to each other. I follow the directions of a very specific paper that gives some insights about the impact of each technique performed on tweets on the quality of the classification results regarding 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 so one has to deal with them. They can be simply removed, tokenized (it means converting each of them to a special token, for example $HASHTAG$) or hashtag segmantation can be performed (it means breaking a hashtag into its constituent tokens and removing the # symbol). The choice is to remove them as they are considered unnecessary;
- URLs handling: an URL is a reference or address used to access resources on the internet. It may happen that a person adds one when tweeting. One can remove or tokenize them similarly to hashtag. The choice is to remove them as they are considered 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 delete all the very common words in English such as the or on. This is usually done precisely because their high prevalence makes them just noisy. Actually in the paper mentioned above it is suggested not to remove them, as they can be significant for the model because tweets are usually very short. I choose to remove them anyway as firstly it is not very clear how they could be related to any form of speech and secondly I do not want them to appear in the most relevant terminology once the model is tested because they would be just noisy and lack of 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 precise number of instances to drop, I try some percent-ages between 75% and 85%, as around these numbers the majority class is more or less as common as the second most present. By retrieving the number of columns of the output matrix, that are the tokens that compose the vocabulary, and the distri-bution of the length of the words, it can be stated that any number within this range does not seem to affect very much 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, one has to be careful about how the method is applied. TF-IDF cannot be performed on the entire data set, otherwise there would be data leakage as the model would have been trained with features extracted also from the test set [2]. On the other hand, if one performs the method separately on the two sets, the outcome is a measure of the goodness of not only the model but also the method itself. Furthermore, as the test set is smaller than the training, the estimate would be probably worse. The solution is to perform the whole method on the train set and to use the resulting vocabulary and the IDF estimates for the test, in order to obtain a coherent outcome. Lastly, while instancing the method, I choose to remove from the TF-IDF matrix all the tokens that do not appear in at least the 0.1% of the tweets in order to reduce the noise (this value is known as cut-off).

Once the method is performed and the correspondent matrix computed, it is time to instance the model. I create a custom function to pass to the XGBoost in order to maximze the metric F1 Score for the class hate as I want the model to be particularly able to detect this kind of speech. F1 Score is the harmonic mean of precision (how many true positive predictions out of all the instances predicted as positive) and recall (how many true positive predictions out of all the true positive instances). I use it because I think it is a more reliable measure of the quality of the model with respect to the other two.

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.

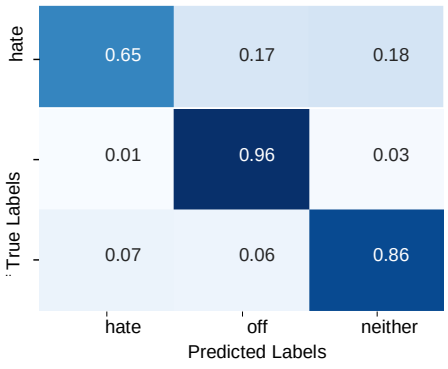

Fig. 1 Confusion Matrix of the project

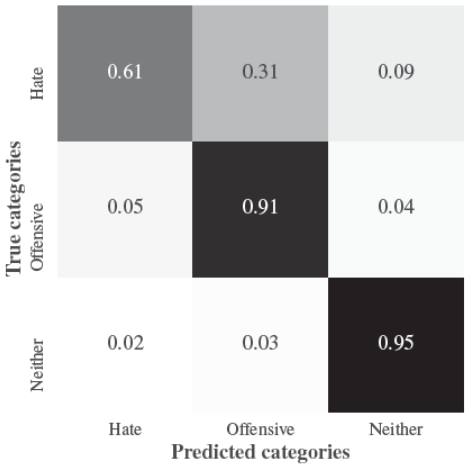

Fig. 2 Confusion Matrix of the paper

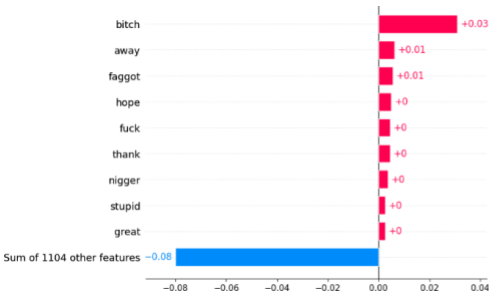

Fig. 3 Relevant terminology for hate speech

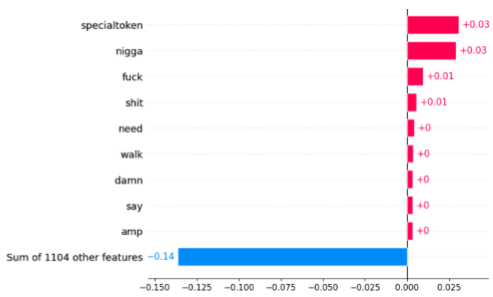

Fig. 4 Relevant terminology for offensive speech

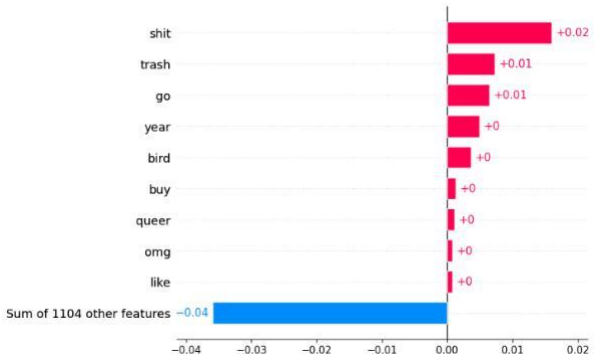

Fig. 5 Relevant terminology for normal speech

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 means that generally XGBoost is better at correctly classifying hate speech but it does not fully comprehend the hierarchy among the three classes, other-wise the results would have been more similar to the other matrix. This is a problem because not all mistakes are equal and it is easier to fix and explain a misjudgment between two classes that are close.

Regarding the relevant terminology, the plots 3, 4 and 5 are helpful. For the class of the hate speech, besides some noisy words (e.g. away or hope), there are as expected some harsh insults, particularly related to racism and homophobia, and vulgar expressions. This is a good sign, because it is in line with the definition reported at the begin-ning of this paper. A special mention is needed for b*tch: according to the classifier,

it denotes strongly a speech as hateful. Even if from a strict semantic point of view this is correct, as this word is strongly misogynist, nowadays, especially in colloquial circumstances, it is often used as a normal way to refer to other people. Consequently, I would not expect it to be so related to the hate speech. For the offensive class, besides some normal words, there are three interesting 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. The only other interesting consideration can be made about queer: this word was originally used as an insult but recently its meaning has been subjected to some changes and can be considered also a normal way to refer to some components of the LGBT+ community. Anyway, I would not expect it in the relevant terminology for the normal speech.

Overall, even if there is some degree of noise that affects the results, there are still some aspects of the hate speech that can be extracted. In general it is easier to associate it with harsh insults that target a specific group (black people, women or LGBT+ com-munity). On the contrary other insults or swear words that are more general can be also associated with offensive speech and so their presence does not denote so strongly a content as hateful. 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. hate speech definition: I think that the first thing to deal with is that there is no unanimous consent on what is hate speech. The consequence is that the results are less generalizable in a broad framework and less comparable with each other because authors create their own definition from which they start their research. It would be necessary to study a unique definition to use at least in the scientific field;
2. user identifier: as explained in [2], usually hate speech generation is associated with a few individuals, but in a normal data set there is no trace of this. This leads to a user bias that often is responsible for a portion of overfitting that can be found in the literature. There are some solutions that can be deployed, for example providing user identifiers. It would be interesting to elaborate further this topic.

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