HATE SPEECH DETECTION

Classification of textual data from white supremacist forum Stormfront on a Streamlit dashboard

Martina Viggiano (954603)

MSc Data Science and Economics - Università degli Studi di Milano

Abstract. This project presents an analysis of textual data, with the aim to detect hate speech.
In the first part of the study, we did a deep data cleaning, removing errors and tokens that did not correspond to words. Then, we analyzed data in terms of most common words and parts of speech to study the type of lexicon used on the forum and then compare the results between sentences labelled as hate speech and those set as neutral. Afterwards, we implemented some supervised statistical methods, trained with the 80% of the dataset and tested on the remaining 20%. By employing five different statistical methods we modelled and tested them by considering for each of them a different set of variables, to add beneficial information and find the model that better classified the label of interest.
In particular, we found three of models - trained of different kinds of data - that reached levels of precision and f1-score above 0.75: even if they have a similar performance on test set, they work differently, thus it may happen that they classify sentences in opposite ways.
Furthermore, to give an additional overview of the analysis and the results we obtained, we developed a dashboard with *Streamlit* to display outcomes and findings. On GitHub repository *textsent_project* it is possible to find the commands to run to open the corresponding *Streamlit* page.

1 Introduction

Through the growing use of social media, the amount of hate speech is also increasing: the common opinion is to consider internet and social networks as a sort of "free-zone" of rights, without the limits that typically people respect in non-virtual circumstances. The distance between people pushes discussions beyond socially acceptable boundaries, both in terms of lexicon used and manners.
The purpose of this project is to create a model able to classify sentences as *hate speech* or *not hate speech*. To do so, we used the dataset containing text extracted from *Stormfront*, a white supremacist forum. In particular, data were taken from the Github repository *hate-speech-dataset* by *Vicomtech*[1].
A random set of forums posts have been sampled from several subforums and split into sentences. Those sentences have been manually labelled as containing hate speech or not, according to certain annotation guidelines.

From the repository we took *all_files* folder, which is composed by the list of all forum posts, and the csv file (*annotation_metadata.csv*) containing the actual label for each file in the other folders together with other information about each sentence, such as user id or subforum id.
Even if the division in test and train sets was also provided, we implemented a random split in train and test subsets by using the function of *sklearn*.
We have not used any column from the *annotation* file, other than the one reporting the label of interest. This is a categorical variable and consists of four classes: *hate, no hate, relation, idk/skip*.
Since the last category consists of strings of alphanumeric meaningless tokens, we dropped them. We also removed sentences classified as *relation* because they were not numerically significant and did not add knowledge to our study.
Then, we only considered *hate speech* and *no hate* as categories of the variable *label*; for simplicity we changed the values in 1 and 0, respectively.

1.1 Data Cleaning

Before moving to the exploratory part of the project, we needed to deeply clean textual data. In particular, we faced some difficulties working with this dataset: it turned out to be full of contractions and entities that created some bias, such as links and metadata. Without cleaning, these alphanumerical items would have been read as actual words, thus producing systematic distortion into our analysis. For this reason we implemented several functions to recognize and remove them.
What we did was to, first, expand some common contractions we found in sentences using *regular expressions*. Then we ran the results into a function in which we implemented operations of the two libraries *pproc* and *cleantext*. Lastly, we deployed further corrections by replacing some parts of the strings that we were not able to get and manage in the previous steps. The last thing was to remove the strings which contained less than 2 characters, that is to say that we did not consider "sentences" composed by only one character.
After cleaning textual data, we focused on a first exploratory analysis of the dataset.
We looked at the number of items in each sentence before and after cleaning - creating three new columns *word_count_before, word_count, word_cleaning*.
Then we employed functions from the *spaCy* library for lemmatizing and tokenizing the sentences, we filtered the sentence that were not recognized as written in English and we created eight new columns, each for a given part of speech. For every sentence we run a function looking for one of the parts of speech – for example, adjectives – that put all the tokens corresponding to that part of speech in one separate column. Then we added a new column that counted the number of items present in each of those columns.
In this way, we re-organized the data frame that we used for the project, dropping some original columns and creating new variables that would be useful for the following parts of the study.

You can find all the steps of the *Introduction* part in the *Jupyter notebook* called *0.1 Pre-processing* on the GitHub repository.

2 Research question and methodology

In the following chapters we are going to address the research questions. In particular, we are interested in finding the most common terminology - mainly as regards *hate speech* sentences - also with respect to various parts of speech, and design some models trained on sets with different composition.
After importing in *0.2 Data Analysis* notebook the data we prepared in the cleaning section, we looked at basic data structure, starting from the distribution of sentences between the two labels 0 and 1. We learned that the dataset is highly unbalanced: *hate speech* sentences are around 1/9 with respect to *no hate speech* ones – respectively only 1196 and 9374. For this reason, we decided to implement dataset balancing techniques later in the project.
Then, we defined a function that, starting from the cleaned sentences, vectorized them by using *CountVectorizer sklearn* function - which converts the corpus to a matrix of token counts - and defined the bag of words of the entire corpus, counting the number of times each word is present within the collection. After that, the function sorted the words with respect to frequency retrieving the most common words of the cleaned – or lemmatized – dataset: in our case, we provided the lemmatized set as corpus.
Lastly, we used all the information collected to design and explore several models, in order to find the best ones in terms of performance.

3 Experimental results

3.1 Descriptive analysis

As we anticipated, we first worked on frequencies. Initially, we plotted the distribution of the labels in the target variable, and then we looked at the histograms showing the frequencies of amounts of words[1] – tokens – for every sentence before and after the cleaning step. As we expected, the number of words decreases after cleaning, but not by a sensible number: the average amount goes from 17.83 to 15.14 tokens per sentence.
Then, after implementing the function *get_top_n_words()* which vectorized and sorted terms, we plotted the bar plot displaying the top 20 most common words. Since we did not remove stop words in the cleaning step, the function provided very general words, such as prepositions, auxiliary verbs, and adjectives. Besides, we can see from Fig. 2 that within the most common words there is also the word "white": since *Stormfront* is a white supremacists' forum, it is self-evident this is likely to happen, because a lot of the posts talk about "whites", even without hate or prejudice purpose.

[1] When in this case we mention "words" we refer to tokens, i.e. singular instances, that may also not correspond to words in terms of verbal expressions.

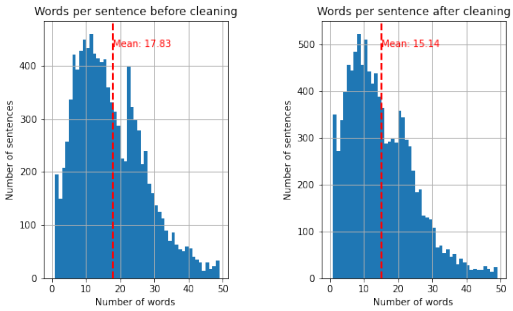


Fig. 1: Word count before and after cleaning.

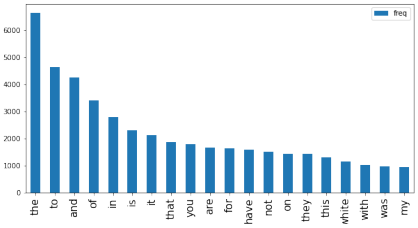
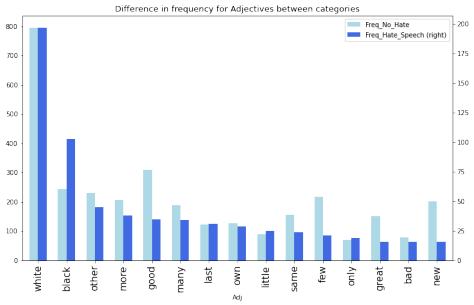
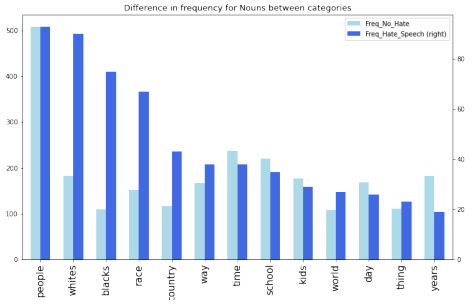


Fig. 2: Most common words.

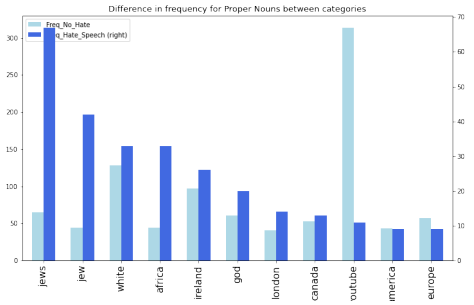
To have an overview of the composition of the dataset, we defined a further function similar to the previous one, filtering words on *spacy* parts of speech: the aim was to collect the most common words in four parts of speech – adjectives, nouns, proper nouns and verbs. Then, to search for peculiarities of the *hate speech* subset, we compared the most common words in the two categories. Please note that in the following bar plots comparing the two categories of variable *label*, we displayed the interjection between the top 20 most common words within the two subsets – in descending order for *hate speech*. This implies that we may have different numerousness based on the number of words present in both *hate speech* and *no hate speech* top words categories.
As regards adjectives, it is clear from Fig. 3 (a) that the most common one is "white"; we will also see in the next steps that this term has been classified by *spacy* in more than one part of speech, on the basis of the context, and in all circumstances it constitutes one of the most common words of that category. Besides "white", we can also find "black" which is more prevalent in hate speech context. The other ones are general adjectives such as "good" and "many" that are not really able to describe and identify the peculiarities of our dataset, since they are very common.
Talking about the main differences between nouns, it can be clearly seen on Fig. 3 (b) that hate speech sentences differ from the others in terms of frequency for



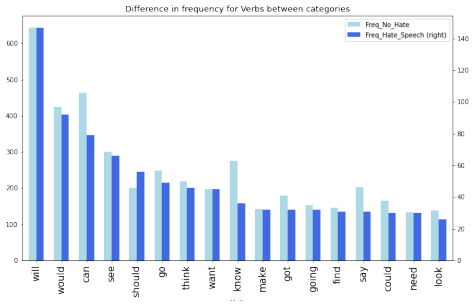
(a) Top adjectives.



(b) Top nouns.



(c) Top proper nouns.



(d) Top verbs.

Fig. 3: Frequencies of Top common terms based of POS.

mainly four words: “whites”, “blacks”, “race” and “country”. It is no wonder these words are more commonly used in discriminatory speeches, since they determine a differentiation between groups of people, based on the ethnicity and color of the skin. Referring to what we said before about the lemma “white”, here we can notice that “whites” has been labelled as a noun, referring to group of people having white skin, often representing the subject and/or object of the sentences. Proper nouns include personal names, geographical names – such as cities, countries – but also words that *spacy* identify as the name to define a group of people or entities, for example “Saint”, “jews” and “white” (Fig. 3 (c)). In this case we can notice that the most common proper nouns in hate speech are “jews” and “jew” – they appear as two different tokens since it was decided to run the POS classifier on non-lemmatized sentences. These two words – and also “africa”, “ireland” and so on – depict a situation in which it is clear that the author of the speech wants to clearly define and distinguish some ethnic group, thus is probably discriminating precise classes of people or religions or ethnicities.
To conclude, we looked at verbs: in Fig. 3 (d) plot we can see that are mainly present auxiliary verbs that, clearly, represent the most common ones with respect to other verbs. In this case there is no such difference between frequencies in *hate speech* and *no hate speech* sentences. As we said, the main verbs we observe are auxiliary, like “will”, “would”, “can” and so on, which are not capable to distinguish our sentences from others, since they are used almost in every kind of speech. In the case in which we applied stop words removal in the cleaning section, we would not have found auxiliaries, but other verbs that would have better identified our corpus.

3.2 Models

In *03. Models* notebook we tested several methods in order to reach the goal of hate speech detection. We employed five statistical methods applying them to different sets of data and variables: in particular, the methods we modelled are Multinomial Naïve Bayes, Support Vector Machine, Logistic Regression, Random Forest, Decision Tree. While the evaluation method we used to judge models’ performance are: accuracy, precision, recall, f1 score.
We started from considering the simplest cleaned dataset, with all the sentences in it. We run the models on lemmatized data, without balancing it with respect to the categories *no hate speech* and *hate speech*. This circumstance led us to an unsatisfactory outcome, in which neutral sentences are decently classified, but hate speech ones are not, since they have a strongly lower numerousness. To solve this issue, we employed *imblearn* random sampling.
Initially, we run the *RandomUnderSampler* over the cleaned dataset: in this way we obtained a smaller dataset – of 1196 units each, instead of (9374, 1196) – where we removed some of the sentences classified as *not hate*. In fact, as we said in data exploration chapter, our dataset is highly unbalanced, so we reduced the numerousness of the bigger category. In this way we obtained better results in terms of classification, both in terms of hate speech recognition, and

also in terms of the overall performance of the models themselves. As you can

	Accuracy Score	Precision Score	Recall Score	F1 Score
Multinomial Naive Bayes	0.680585	0.651408	0.680585	0.677805
Support Vector Machine	0.753653	0.781395	0.753653	0.753007
Logistic Regression	0.743215	0.781553	0.743215	0.741953
Random Forest	0.649269	0.723270	0.649269	0.639072
Decision Tree	0.605428	0.606838	0.605428	0.605376

Fig. 4: Performance of models for UnderSampled dataset.

see from Fig. 4, the best performing method is the Support Vector Machine with f1 score of 0.7530, which is a good result for our classification.
Then we tested the *RandomOverSampler* of *imblearn*: unlike the previous method, it balances the dataset enlarging the minority class. Clearly it is not a good solution for our problem, because there exists the risks of adding bias to our model. As a matter of fact, it works in an odd way, by reaching performance scores really high. This happens because the model is overfitting, that is to say that is good working with this data, but will not work properly on different data.
Thus we decided to modify the data we are giving to test and train our models: we selected two combination composed of only some parts of speech and we used the *RandomUnderSampler* balancing. We performed several tests, but we choose to display only two versions of them in the project.
The first one is composed of nouns, verbs, adjectives, pronouns and proper nouns. Even if this model learns from a smaller number of words with respect to the previous ones, the performance seems to be even better. In particular, the Lo-

	Accuracy Score	Precision Score	Recall Score	F1 Score
Multinomial Naive Bayes	0.680585	0.651408	0.680585	0.677805
Support Vector Machine	0.768267	0.801887	0.768267	0.767501
Logistic Regression	0.778706	0.806452	0.778706	0.778216
Random Forest	0.703549	0.793939	0.703549	0.696198
Decision Tree	0.707724	0.705394	0.707724	0.707722

Fig. 5: Performance of models for the first combination of POS.

gistic Regression method classifies with precision 0.8065 (0.76 for *no hate speech* and 0.81 for *hate speech* sentences) and f1 score of 0.7782. These results are so good that we will also consider this model for the final tests of our project.
The second combination of parts of speech includes nouns, proper nouns, verbs, adjectives, pronouns, subordinating conjunctions, prepositions, coordinating con-

junctions, and "other" defined by *spacy*. Also in this case, the best model between the five methods provided is the Logistic Regression, which unfortunately shows lower performance scores with respect to the previous model - the first POS combination - even if in this case we added new parts of speech, which should have brought further information. This shows that, even if we consider a higher number of elements in our model – as it happens in this case with respect to the previous one – it does not imply that the model will work better.

	Accuracy Score	Precision Score	Recall Score	F1 Score
Multinomial Naive Bayes	0.684760	0.659420	0.684760	0.682919
Support Vector Machine	0.768267	0.796296	0.768267	0.767708
Logistic Regression	0.774530	0.804651	0.774530	0.773939
Random Forest	0.682672	0.763636	0.682672	0.674803
Decision Tree	0.691023	0.702222	0.691023	0.690740

Fig. 6: Performance of models for the second combination of POS.

In conclusion, we performed some tests on the two selected models - the one trained by lemmatized under sampled dataset and the one trained by the first combination of POS also unders-sampled. To display the outcomes and to give the possibility to interact with data and visualize the results, we created a dashboard on *Streamlit*; it can be found on the GitHub repository *textsent_project*. In the home page you can read the introduction and have an overview of the entire dashboard; in the second page – Data Exploration – it is possible to use the interactive tools to display plots and tables; in the last page – Classification – you can actively test the models, by writing the sentence you want the two methods to classify, labelling them as *hate speech* or *not hate speech*.
You will find a default sentence "I hate all of you!"[2] which has been classified correctly as *hate speech* by the first model, but wrongly by the second one.
However, you can also find another suggestion to test, in order to show a situation in which the second model is more capable of classifying the sentence.

4 Concluding remarks

In summary, through this projects we analyzed the *Stormfront* dataset and we designed some models capable of classifying sentences as *hate speech* or *not hate speech* with a good performance - based on the results on test set.
Although, we found several issues: the dataset on which we based our study is highly unbalanced and, moreover, the number of sentences provided is inadequate. It implies that our models will have some difficulties in working with

[2] Disclaimer: the examples used in this project are inspired by the sentences in the dataset itself. Some changes were made to disguise the sentences and alter the appearance, to make it less easy for the models to classify them.

external sentences, composed of terminology not present in the training set or belonging from different contexts.
This could be easily proven by typing a sentence in the cell present in the third page of the dashboard. If the sentence written is not clearly neutral/positive, or contains terms mainly used in the *hate speech* sentences, the two models are unlikely to classify properly the sentence written. For example, the term "black/s" is highly biased, since it is more often present in sentences labelled as *hate speech*; thus if you provide the models a sentence in which this word - or one of its declension - is present, the model are likely to classify it as *hate speech*, even if it is clearly not.
One possible solution can be to expand the training size by adding new labelled sentences, even coming from different social networks, forums and also contexts: enlarging the range of terminology and diversifying the users writing - or saying - the sentences, will clearly provide more information to the models and will help them improving their classification ability.

References

1. Stormfront dataset on Github, Vicomtech, https://github.com/Vicomtech/hate-speech-dataset.
2. Examples and code chunks for the course of Text Mining and Sentiment Analysis, Professor Ferrara A., https://github.com/afflint/textsent.
3. VADER Sentiment Analysis, GeeksforGeeks, https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/, Last Updated : 23 Jan, 2019.
4. Hate speech classification project, Sachin Jain, GitHub repository Detection And Classification Of Hate Speech In Social Media Using Python, https://github.com/Sachin-Jain-98/Detection-And-Classification-Of-Hate-Speech-In-Social-Media-Using-Python/blob/master/Hate_speech_detection_Final_code.ipynb.