Hate Speech Detection

Gregorio Luigi Saporito - ID 941503

Università degli Studi di Milano, Data Science and Economics
gregorioluigi.saporito@studenti.unimi.it
https://github.com/gregorio-saporito/hate-speech-detection

Abstract. The aim of this project is (1) to build a hate speech detection classifier and (2) to identify distinctive terminologies. Different learning algorithms are tested coupled with various approaches for feature extraction such as TF-IDF, document embeddings with neural networks, and estimation of the sentiment polarity. A logistic classifier with features extracted through latent semantic analysis (LSA) performed the best among the experiments carried out. The results are discussed coupled with some approaches to improve the interpretability of the model which are achieved by identifying associations between important features and relevant terminology in each category. We then use non-parametric bootstrap to explore how the classifier behaves when given as input samples containing relevant terminology. To conclude, an app prototype for hate speech detection is presented.

Keywords: Multi-class classification · Hate speech · Latent semantic analysis.

1 Introduction

Detecting hate speech is vital to prevent the uncontrolled spread of hate online [1]. The intrinsic complexities of the human language make this task arduous. This is particularly true when it comes to distinguishing between hateful versus offensive content [2] since their boundaries are hard to define quantitatively. In this project, multi-class classification is performed on textual data to detect hateful and offensive contents. Different experiments are carried out to identify the best performing model among a logistic classifier, a random forest, XGBoost, and support vector machines. In particular, a logistic classifier with l2 regularisation and features extracted through LSA performed the best among the different classifiers. Relevant terminology associated with each class is explored coupled with an app prototype for hate speech detection.

2 Research Question and Methodology

The research question that this project intends to tackle is:

(1) to train a classifier to detect contents considered hateful, offensive, or neither of the two and (2) to identify the most relevant terminology associated with each category

Text Preprocessing To address the research question, the tweets are preprocessed to make them more digestible for the classifiers. We normalise constructs such as tags and URLs to make them easier to recognise by the classifier. An example of a transformation to normalise a tag would be to turn a generic tag @username into $MENTION$. We then apply other transformations such as lower-casing, expanding contractions, removing punctuation, and lemmatizing words with the WordNet Lemmatizer.

Transformers and Feature Selection Available transformers like the TfidfVectorizer from the Scikit-learn library are used to extract features from documents. Some custom transformers are also used to extract features to incorporate in the machine learning pipelines. In particular, a custom Doc2VecTransformer was used to extract a vectorised representation of documents directly inside a Scikit-learn pipeline. This is achieved by relying on the distributed memory approach (PV-DM) available in the Doc2vec model for document embeddings. A custom SentimentTransformer is built to extract the sentiment from documents which is achieved relying on the nltk SentimentIntensityAnalyzer tool. Transformers turned out to be particularly useful during cross-validation to avoid data leaks. Given the extensive number of features, two distinct approaches are chosen to select the most relevant features. The first approach focuses on univariate feature selection with the SelectKBest transformer while the second approach is multivariate in nature and relies on the technique of truncated singular value decomposition (TSVD). Applying TSVD on a TF-DF matrix can be thought of as a form of latent semantic analysis where latent variables carrying semantic meaning are capable of explaining the interrelationship among terms in the TF-IDF matrix.

Relevant terminology The relevant terminology associated with each category was identified with the TF-IDF metric which measures the importance of terms for a document in a corpus.

$$TF\text{-}IDF_{i,j} = TF_{i,j} log(\tfrac{N}{DF_i}) \text{ for word } i \text{ in document } j$$

Relevant terminology is identified by returning for each class the top n terms that have on average the highest TF-IDF score and potentially setting a TF-IDF threshold for values to be considered relevant enough. The association between relevant terms and important features of the best performing classifier is explored to improve the interpretability of the classification outputs. Finally, we explore the behaviour of the classifier when predicting sampled documents containing the most relevant terminology identified.

3 Experimental Results

Dataset The data was retrieved from Davidson et al. (2017) [2] and it can be downloaded from their repository[3]. The dataset contains roughly 25k tweets extracted through the Twitter API containing posts considered by internet users as hateful. These posts were manually labelled by CrowdFlower (CF) workers into three categories:

- 0 for hate speech
- 1 for offensive content
- 2 for tweets not falling into the previous categories

In particular, the dataset contains a total of 24,783 observations and two variables of interest: (1) the tweets stored in string format and (2) the respective labels associated with each tweet. No missing values are present for the variables of interest. The different classes are imbalanced resulting in a more challenging classification problem. Roughly 77% of tweets are labelled as offensive, 6% as hateful, and 17% as neither offensive nor hateful (Fig. 1).

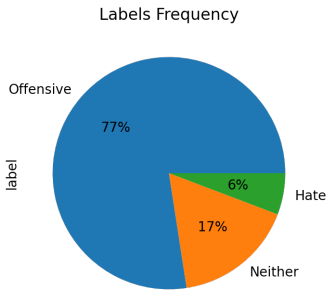


Fig. 1. The pie chart represents an imbalanced distribution of classes

Training-test Split, Classifiers, and Evaluation Metric We then split the corpus of documents by leaving 10% of the observations out for testing and fixing a random state to ensure reproducibility of the results. Different approaches including logistic regression, random forest, XGBoost and support vector machine are tested to identify the best performing model. Some forms of regularisation are also introduced to prevent overfitting. Cross-validation is performed on the training data for hyperparameter testing and to select the optimal number of features. The evaluation metric chosen is the balanced accuracy score in order to deal with the underlying imbalance of some multi-class classification problems. The balanced accuracy is calculated as the macro-average of recall per category. The results will be discussed more in detail in the experimental results section.

Latent Semantic Analysis (LSA) and Logistic Classifier The first approach is to run LSA on the TF-IDF matrix stored in sparse format by using truncated singular value decomposition (TSVD). In contrast with ordinary PCA, TSVD does not centre the variables before it is applied to the data. This makes this dimensionality reduction technique more efficient when it comes to managing large sparse matrices [4]. The lack of centring would not be an issue in this context since the TF-IDF matrix is the result of a twofold normalisation. The first normalisation is row-wise since the term frequency in a document is in percentage terms. This makes documents of different lengths comparable. The inverse document frequency can instead be thought of as a column-wise normalisation. For a total of 22,304 documents in the training data, 19,165 terms were identified. As the marked elbow shape from the scree plot shows (Fig. 2), the highest singular values in absolute terms explain a larger fraction of variability in the data compared to the others which indicate the presence of redundancies.

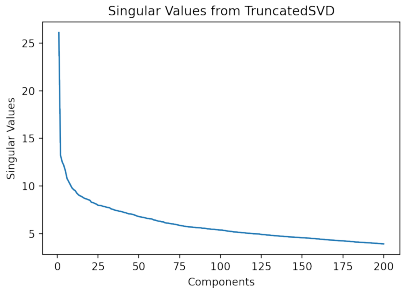


Fig. 2. The scree plot shows singular values and their respective components

This suggests that dimensionality reduction is suitable in this context. A different number of components to feed the logistic classifier were tested using cross-validation. The best performance was obtained with 1000 components and l2 regularisation which returned a cross-validated balanced accuracy of 0.80. The class_weight parameter was set to 'balanced' giving more importance to underrepresented classes in the data to manage the class imbalance problem. The parametrised model was then used on the test set, returning a balanced accuracy score of 0.8086.

Rich Feature Extraction (mixed), Feature Selection, and Logistic Classifier The second approach was to rely on a richer set of features where each document would be associated with its TF-IDF scores, a doc2vec vectorisation, and 3 features denoting sentiment polarity for positive, neutral, and negative sentiment. Given the large set of features, the SelectKBest tool was used for feature selection. After the feature selection, the features were converted to a

dense representation and scaled since the logistic classifier makes use of regularisation. Different cardinalities of the feature set were tested with the best model returning a cross-validated score of 0.8023 and a final score on the test set of 0.8024. Despite being slightly worse, the performance was fairly similar to the logistic classifier with LSA.

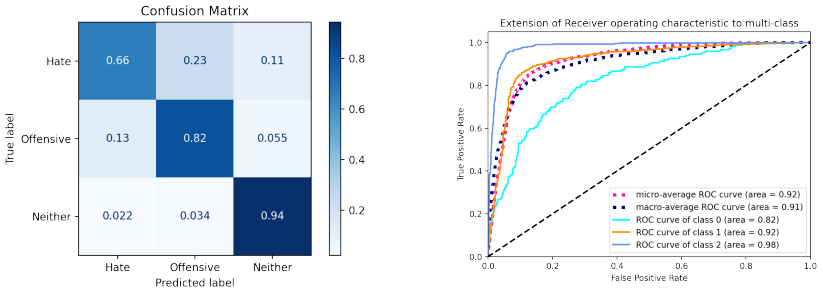
Random Forest and XGBoost The third approach involved the same rich feature extraction process of the second approach. Overall the performance was worse than the previous two models. This is perhaps due to the problems that decision trees generally encounter with imbalanced datasets. The best balanced accuracy score was 0.57 for both the cross-validated score during parameter tuning and the final score on the test set. The performance was better for XGBoost with a 0.68 score for both cross-validation during parameter tuning and the final performance on the test set.

Support Vector Machine The final algorithm tested was a support vector machine that returned the best results with the class_weight parameter set to balanced. In particular, the balanced accuracy score was 0.78 and 0.77 during cross-validation and on the final test set respectively.

Comparison and Results Table 1 compares the results of the different models tested. Both logistic classifiers had a fairly similar performance. The logistic with LSA slightly outperformed the logistic with rich/mixed features on the test set. We will therefore present next the results of the logistic classifier with LSA in more detail. The confusion matrix (Fig. 3 (a)) shows that, as initially hypothesised, the model faces more challenges when it comes to distinguishing between hateful and offensive posts (the respective accuracies are 66% and 82%). The model finds the classification of non-offensive contents more straightforward instead (94% accuracy). These results are confirmed under the more flexible framework of the receiver operating curve where a single threshold is not fixed (Fig. 3 (b)). The quality of prediction in the non-offensive category is the highest (0.98 area under the curve), followed by the offensive category (area 0.92) and the hate category (area 0.82). The model has a stronger tendency to predict as offensive posts that are hateful rather than doing the opposite.

Table 1. Comparison of the results from the models tested.

Model	**Cross-validated score**	**Test score**
Logistic classifier (LSA)	0.80	0.81
Logistic classifier (rich/mixed features)	0.80	0.80
Random Forest	0.57	0.57
XGBoost	0.68	0.68
Support Vector Machine	0.78	0.77



(a) (b)

Fig. 3. (a) confusion matrix and (b) receiver operating curve

Feature Importance and Relevant Terminolgy Since the underlying data is normalised, the coefficients of the logistic regression concerning the hate category can be interpreted as indicators of feature importance. As can be seen from Fig. 4 (a) the importance of features decays exponentially as less relevant explanatory variables are considered. We can now explore the most relevant terminology associated with each category by using the TF-IDF construct (Fig. 4 (b), (c), and (d)). This is achieved by returning the top n terms with the highest average TF-IDF score in each category. A TF-IDF threshold can be set to filter out terms not considered relevant enough such as stop words. We can see that the most relevant terms in the hate category are "fa*", "fagg**", and "nigg**". We can see that there are some intersections with the offensive category with the term "fagg**" appearing. However, the most relevant terms in the offensive category are generally "puss*" and "bitc*". The most relevant term in the non-offensive category is instead "trash".

We can extract a number of relevant terms from the different categories and explore their associations with the most important features in the logistic classifier. Since the features of the logistic classifier are principal components from SVD, they can be represented as linear combinations of the original manifest variables (i.e. terms in the TF-IDF matrix). It, therefore, makes sense to use absolute correlations to inspect the associations between principal components (i.e. features in the classifier) and relevant terms in the TF-IDF matrix. This can be useful to improve the interpretability of the model. As figure Fig. 4 (e) shows, features number 5 and 6 (the number reflects the order of importance for the classifier) are negatively associated with the hate prediction outcome because of the negative coefficient. Among relevant terms, features 5 and 6 have the strongest association with the "nigg**" word. This suggests that these components might represent a semantic construct that moves in the opposite direction with respect to the "nigg**" term (i.e. negative association) or represent a construct that makes use of the term in a non-hateful way. Feature 4 is instead strongly associated with the "fa*" term and probably represents a semantic con-

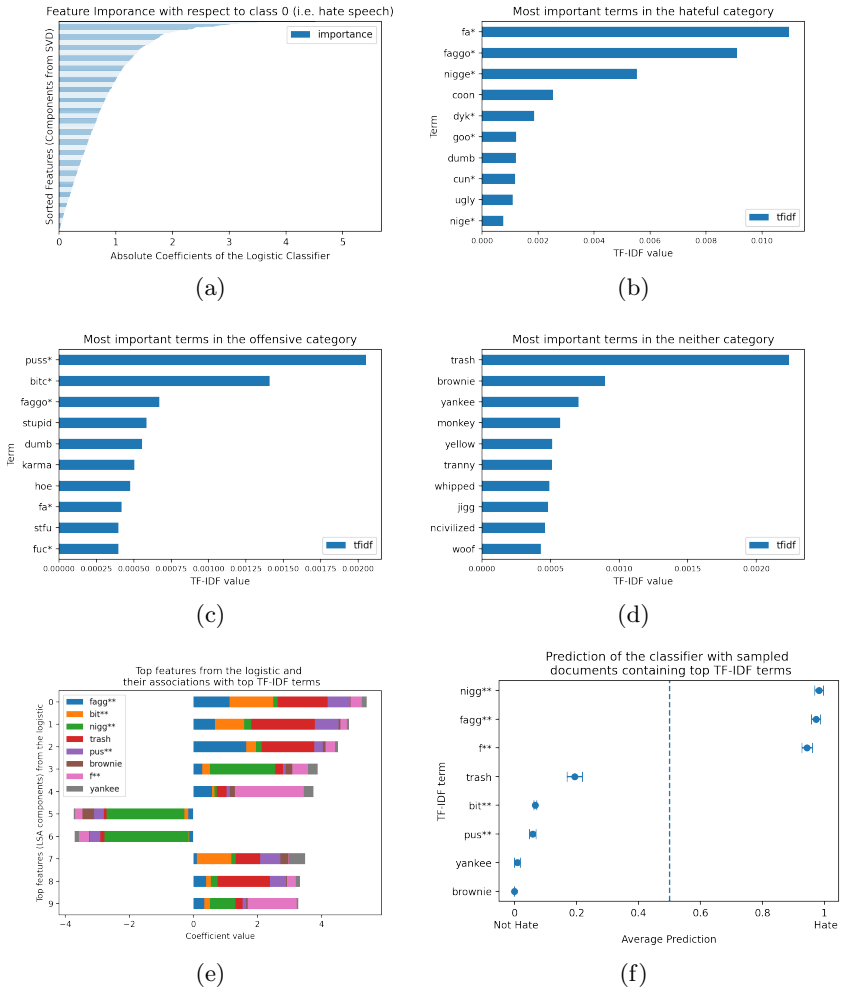


Fig. 4. Relevant terminology and Feature Importance: (a) feature importance, (b) relevant terminology in the hateful category, (c) relevant terminology for offensive posts, (d) relevant terminology for non-offensive posts, (e) association between relevant terminology and important features, and (f) non-parametric bootstrap to explore the behaviour of the classifier when given as input samples containing relevant terms in each category.

struct that makes use of the term in a hateful way. Fig. 4 (f) shows instead the average prediction of the classifier when given as input samples containing distinctive terminologies for each category. The confidence intervals are constructed with non-parametric bootstrap. As expected, Fig. 4 (f) shows that, on average, terms such as "nigg**" or "fagg**" are a strong indicator of a potentially hateful post. It is instead highly likely for samples containing the terms "yankee" and "brownie" to be considered non-hateful.

4 Concluding Remarks and App prototype

In this project, a series of experiments were carried out to identify a classifier for hate speech detection. Different feature extraction procedures were implemented to extract explanatory variable which could be useful in the classification problem. Given the extensive number of features, we tried different feature selection techniques. Different learning algorithms were tested: a logistic classifier, random forest, XGBoost, and a support vector machine. A logistic classifier with features extracted through LSA performed the best among the proposed models. We then explored relevant terminology associated with each category through the TF-IDF construct and identified the associations that these terms have with the most important features and the prediction outcomes of the classifier. Despite some differences, the average accuracy of the model is similar to the one found in Davidson et al. (2017) [2]. In particular, the project proposed here achieved an average accuracy of 81% compared to 82% for Davidson et al. (2017). However, the model proposed here performed better in classifying hateful content with an accuracy of 66% versus 61% in Davidson et al. (2017). There is however still room to improve the performance of the model, particularly for what concerns the hate category (see the poorer performance of the hate ROC curve in Fig. 3 (b) compared with the other categories). This is something that is worth exploring in future studies by experimenting with new models and new approaches for feature extraction and selection. A potential increase in the size of the dataset could also be beneficial to improve the vectorised representation of documents.

We conclude this project by presenting an app prototype for hate speech detection. The user can provide as input some content to inspect for hate speech. The classifier returns the predicted outcome, the probabilities of the respective categories, and a ranking of terms that are potentially driving the decision. The hyperparameters that the user can specify are: (1) a maximum number of terms to include in the ranking (whose order of priority is decided based on the TF-IDF metric) and (2) a maximum number of features to consider (again sorted by importance from the highest to the lowest coefficient in the logistic). Each feature in the logistic (which is also a component from SVD) will be associated with a ranking of terms determined by computing the correlation between each term and that feature. The higher the absolute correlation the higher the importance of that term for that particular feature. Each term will be associated with different importance scores, one for each feature of the model. These scores will be summarised by performing a weighted average: each importance score will be

weighted considering the importance of the respective coefficient in the logistic model (given by the magnitude of the absolute value of that coefficient). The result is a ranking of the top n terms associated with the most relevant features in the model. Fig. 5 shows an example of a query that a user can perform to inspect a post for hateful content and explore relevant terms associated with the predictors of the model.

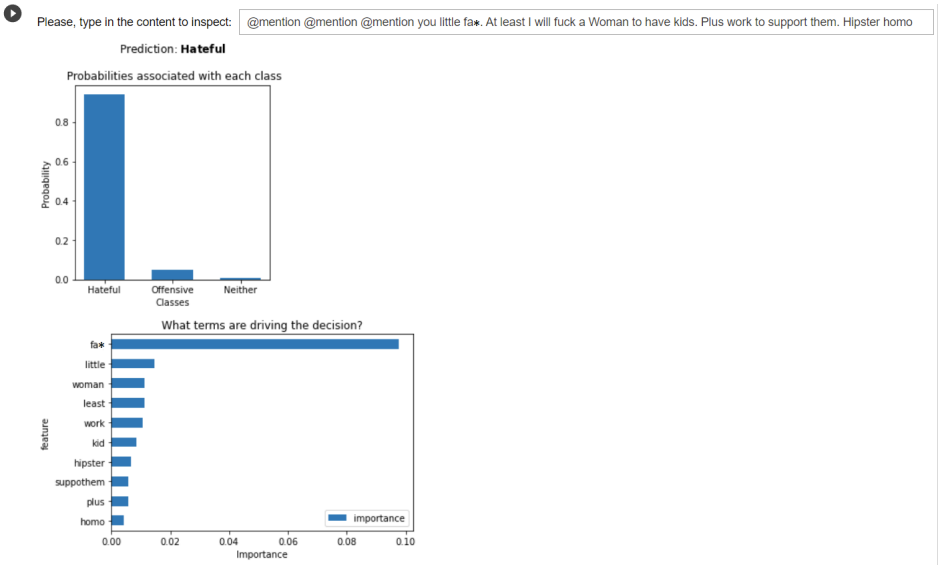


Fig. 5. App Prototype

References

1. Kovács, G., Alonso, P., & Saini, R. (2021). Challenges of Hate Speech Detection in Social Media. SN Computer Science, 2(2), 1-15.
2. Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 11, No. 1).
3. https://github.com/t-davidson/hate-speech-and-offensive-language
4. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
5. Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 759-760).
6. Schmidt, A., & Wiegand, M. (2017, April). A survey on hate speech detection using natural language processing. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media (pp. 1-10).