Hateful Speech Classification with Machine Learning approach

Yuxin Zhang
Cornell Tech
Cornell University
yz2726@cornell.edu

Songyu Du
Cornell Tech
Cornell University
sd933@cornell.edu

Junhan Xu
Cornell Tech
Cornell University
jx359@cornell.edu

Abstract

In this project, we tried to train a model that takes the preprocessed twitter text data as input and classify whether the text belongs to a hateful speech or not. Overall, we experimented text-encoding method at both word level and character level, and trained five machine learning models including Bernoulli Naive Bayes, SVM, Xgboost, BERT, and LSTM. By implementing experimental analysis and comparing the performance metrics obtained by each model, we achieved our best performance through a combination of word-level text-encoding and BERT model, with an F-1 score of 0.9284.

1 Introduction

With the development of social networks, diverse social platforms have become indispensable parts of people's life. We may share our moments on Instagram, record diaries through Snapchat, and type comments on Twitter. The words posted on social media are not all amicable words. Some of them may contain content that is racist, offensive, sexist, and so on. The goal of the project is to try to scan through these social media posts and detect whether they contain "hate speech" or not. This is an application result as if the model is well-treated, we can apply it to diverse social media platforms to monitor users' behaviors and thus build a friendly online environment.

2 Background

For such a long time, the topic about hate speech detection has been an interesting area which attracts lots of researchers to contribute diverse research papers with all kinds of detection methods. For example, Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber, in their paper *"Automated Hate Speech Detection and the Problem of Offensive Language"*, improved the accuracy of automatic hate-speech detection by further classifying the difference between hateful speech and offensive language[2]. We refer to the experience and datasets from these papers to build up our own model to detect the hateful words.

To understand this report, we recommend you better have the background knowledge in the following areas. The understanding of the basic intuition behind supervised learning classification algorithm, including what inputs it takes and what prediction it tries to make, is helpful to have an overall comprehension of this paper. You also better have some basic knowledge about text-encoding methods, one at the word level and the other at the character level. It's fun if you don't get the way we clean the data as it's just a preprocessing process. Moreover, the formulation and differences between the following machine learning models, Naive Bayes, SVM, XGboost, BERT, and LSTM are the main parts of this project. For the performance metrics, the understanding of F-1 score and accuracy is enough for this project.

Applied Machine Learning Project Report. Fall 2020.

3 Method

3.1 Data Source and Preprocessing

We use three datasets from three existing hateful language classification papers, which all use tweets as text data for classification and have been labeled by experts[2][9][8]. For our binary classification purpose, we merged data points labeled as "Racism" and "Sexism" into one "Hateful" category. Then, We examined the class distribution of our full dataset, which is important for the modeling methodology as a non-symmetrically distributed dataset, if randomly sampled in the training set, would lead to a biased model[10]. Further, to prepare data for text encoding, we preprocessed it by converting text to lowercase, lemmatizing verbs, replacing contraction, removing URLs, non-ASCIIs, punctuations, and stopwords.

3.2 Text Encoding method

We applied text-encoding methods at two different levels – the word level, and the character level[5]. On the word level, we applied the N-gram encoding model, which converts a collection of text documents to a matrix of token counts. We experimented the data encoding model with N = 1, N = 2, and N = 3. And we fixed feature dimension at 6000 to avoid multidimensionality issue in model training.

Text-encoding on a character level might not sound promising since unlike words, characters don't have semantic meanings. However, it has proved to be successful in the NLP field of text translation, predictions, and classification in recent years. And this is because character level models pick up on "deep" structures of language and leverage them. And we can achieve the character level encoding by tokenizing texts on the character level, transforming them to a sequence of integers, and rearranging them to the same length[1].

3.3 Machine learning Algorithms

We experimented with five Machine learning algorithms – Naive Bayes, Linear SVM, Xgboost, BERT, and LSTM, with the first four trained on word-level encoded dataset and the last one trained on character-level encoded dataset.

Naive Bayes. First, we implement the Naive Bayes classifier (Bernoulli NB) on our data, which is the easiest and fastest classifier among those we try in this project. Naive Bayes model has lots of advantages, like easy to implement and fast processing. However, at the same time, it contains some drawbacks. For example, it relies on the independent assumption and also cannot hold data precision for small values. Nevertheless, we can still use this wide-used classifier to get a preliminary feeling of the data and model performance.

SVM. Secondly, we choose to implement the SVM classifier because it's one of the binary classification algorithms. The training time of SVM could be long with a certain number of features, type of kernel function and regularization parameter, However, the performance of the SVM model is generally considered good, and SVM also allows easy tuning with different regularization hyperparameters.

Xgboost. Thirdly, we choose Xgboost classifier to experiment with for many reasons. Firstly, we would like to leverage the high precision of a boosting method. Secondly, Xgboost enables flexible hyperparameter tuning and regularization. Thirdly, it's a model with explanatory power[6].

BERT. Compared to other word-level encoders that treat each word one by one, BERT process words in relation to all the other words in a sentence, thus considering the full context of each word. And compared to other machine learning algorithms, BERT is a deep learning algorithm that has a 12-layer, 768-hidden, 12-heads, neural network architecture, and it achieves a "state-of-art" performance on NLP problems involving text interpretation[7][4].

LSTM. Lastly, LSTM is chosen to be applied along with the character-level text-encoding to capture sequential dependencies features between characters. As part of an RNN algorithm, LSTM utilized "memory" of past input into predicting the present result. Besides, LSTMs are developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

4 Experimental Analysis

4.1 Dataset overview

The preprocessed dataset has a total of 47652 data points. We divided it into three categories – the training set, the validating set, and the testing set, with each occupying 68%, 17%, 15% of the whole dataset.

The class distribution of the full dataset shows 56% data points(Text) classified as "Hateful", showing a close-to even distribution of our binary classification. This means that we can use accuracy as a measurement metric for the model performance without suffering from bias. [see Figure 1]

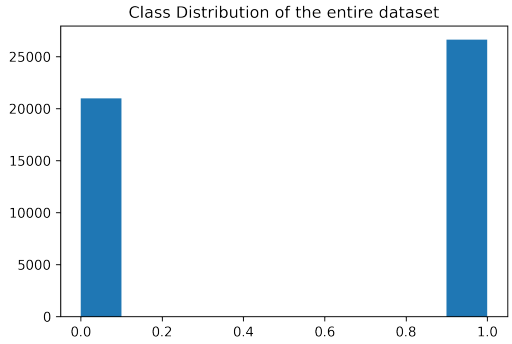


Figure 1: Dataset distribution

4.2 Model Tuning and performance

Naive Bayes. As the Naive Bayes model acts as the base model which gives a preliminary feeling of the performance, we don't need to do any model tuning through adjusting the parameters. We just trained our model on the 1-3 grams training dataset and got the performance metrics by predicting the 1-3 grams validating dataset. By using the F-1 score as the primary evaluation metric, the 1-gram dataset provides the best model performance with an F-1 score of 0.8938.

Linear SVM. We found training SVM with Gaussian kernel took an extremely long time with the size of our feature matrices, so we decided to only train a linear SVM model on the preprocessed training datasets and tuned the regularization hyperparameter (C) on values of [0.01, 0.03, 0.09, 0.1, 0.15, 0.5, 1.0]. Best performance is achieved, if measured by F-1 score, at the linear SVM model with C = 0.09 on 2-gram word-level encoded dataset. The best F-1 score is 0.9162.

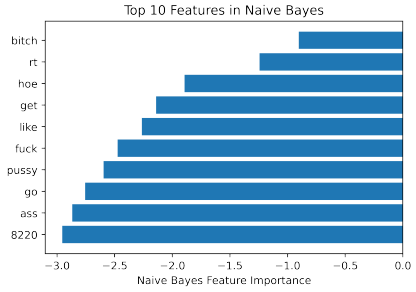
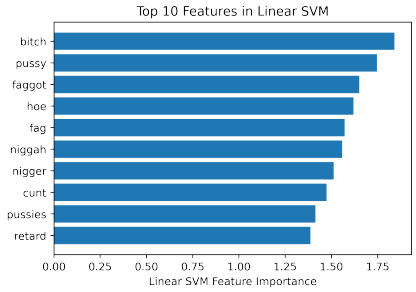
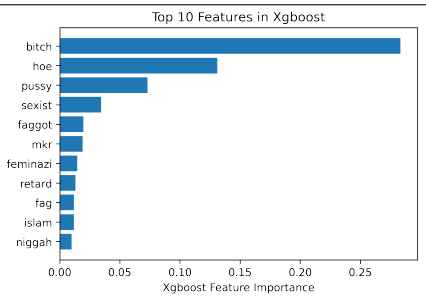
Xgboost. For Xgboost model, we focus on tuning parameters on n_estimators ([500, 800, 1000, 1500, 2000, 3000, 5000, 8000]) and min_samples_split ([2,5,8]) on the 1-gram, 2-gram, 3-gram word-level encoded datasets, using the default tree structure. Using validation set F-1 score as the primary measurement, the best performance of the model is achieved on the 1-gram word-level encoded dataset, with n_estimators = 5000 and min_samples_split = 5. And the corresponding highest F-1 score on validation set is 0.9110.

BERT. Using the pre-trained BERT Model from TensorFlow Hub, we encoded both the training and validation dataset. Also, we applied the NN model with a "sigmoid" activation function, a 0.4 portion of dropout, and Adam optimizer with learning rate of 2e-5, and a 10 epoch training process. It eventually attained a F-1 score on validation set at 0.9284.

LSTM. Using the character-level encoded dataset, which is of dimension 586, we focus on tuning parameters hidden nodes number ([30,40,50,60,70,80,90, 100]) and hidden layer number ([1,2,3,4,5]). Using "tanh" as the activation function, 20% drop out at each layer, and binary cross entropy as loss. The best validation set performance is achieved at 60 hidden nodes and 1 hidden layer, with F-1 score of 0.8925.

The results of these experiments were presented in Table 1.

Table 1: Model Performance and Feature Analysis table

Data encoding method	ML model	Feature Dimension	Best F-1 Score	Top features
Word-level	Naive Bayes	6000	0.8938	
	Linear SVM		0.9162	
	Xgboost		0.9110	
	BERT	128	0.9284	/
Character-level	LSTM	586	0.8925	/

5 Discussion and Prior work

- **Word-level text-encoding generally outperforms the character-level text-encoding**
  We experimented with four machine learning algorithms based on a dataset processed through word-level encoding. And each one of the four has better performance on validation set than the model trained on a dataset encoded on character-level, when measured by F-1 score. This can be explained by the fact that the identification of hate speech depends highly on the semantic meaning of certain word, instead of only on word and character sequence combinations. This is also justified in previous research on hateful language classification. For example, Ji Ho Park and Pascale Fung, in their paper *"One-step and Two-step Classification for Abusive Language Detection on Twitter"*, experimented character-level, word-level, and hybrid-level text-encoding, and trained them separately on CNN[5]. And the character-level has the worst performance on validation precision, Recall, and F-1 score.
- **Ordinary machine learning algorithms can achieve compelling performance as the pre-trained Neuron Network model**
  In the word-level encoded dataset, we applied three ordinary machine learning algorithms – Naive Bayes, Linear SVM, and Xgboost. We also applied a pre-trained NN algorithm called BERT. Different from our expectation, the BERT only outperforms the other algorithms by

little, achieving a validation F-1 score of 0.9284. And the worst performance is attained by Naive Bayes at 0.8938. This means that the benchmark model is good enough to classify the dataset, indicating that our datasets are well-processed and encoded.

- **Different machine learning models extract similar features from the dataset**

  Applying three ordinary machine learning algorithms, we analyzed the top features extracted by each model with explanatory power. For Naive Bayes, the model assigns negative weight to each feature, and those with more negative weights are features that are more important in deciding a "Non-hateful" tweet. On the contrary, the Linear SVM and Xgboost model assign positive weight to each feature, giving higher weights to features that are more important in deciding a "hateful" tweet. To align top-feature comparison in deciding a "hateful" tweet, we ranked the feature importance in Naive Bayes model in descending order (from less negative to more negative). By visualizing them in a horizontal bar graph, we can see that the top features (as words) from three models are very similar. This indicates that those models detected and extracted similar feature structures and assigned similar weights (in relative terms) to them.

- **Binary classification archives higher accuracy and F-1 score compared to multi-class classification**

  Our trained models demonstrate the highest F-1 score of 0.9284 from BERT, and the second highest of 0.9162 from Linear SVM. Referring to some previous literature, we find their result somewhat achieving lower F-1 score. And the largest difference in their research is that they focus on detecting classes of more than two. For example, *"Using Convolutional Neural Networks to Classify Hate-Speech"*, researchers assigned each tweet to one of the four classes (racism, sexism, both, and non-hate-speech), and it eventually achieved a validation F-1 score of 0.783[3]. The text classification itself has annotation difficulty because they are emotionally and textually dependent, thus subjective to individuals. As a result, annotating more classes inevitably brings more human error to the original data, thus compromising the classification performance.

6 Conclusion

In this project, different hate speech classification models trained using pre-labeled tweets were implemented successfully and analyzed comprehensively. Both word-level (bags of 1-gram, 2-gram, and 3-gram) and character-level text encoding methods were implemented in generating feature matrices from the preprocessed dataset. In achieving binary classification of tweets on hate speech, four machine learning algorithms, Naive Bayes, Linear SVM, Xgboost, and BERT, were implemented based on word-level encoded dataset, and LSTM were implemented based on character-level encoded dataset.

As shown in the results, the performance of all these models with different text-encoding methods was great, with the F-1 scores at a maximum of 0.9284 and a minimum of 0.8925. Best performance was achieved using BERT on word-level encoded data with a 0.4 portion of dropout, Adam optimizer with learning rate of 2e-5, and a 10 epoch training process. Worst performance was achieved using character-level encoded dataset.

Overall, we concluded four interesting observations from our experiments and previous works. First, word-level text-encoding outperforms the character-level text-encoding methods. Second, ordinary supervised machine learning algorithms can achieve compelling performance as the pre-trained Neuron Network model. Third, different machine learning models extract similar features from the dataset. Last, binary classification archives higher accuracy and F-1 score compared to multi-class classification.

Assuming that hate speech in the English language continues to transform at the pace at which it is currently doing so, the detection and classification of hate speech would require constant updates on pre-labeled datasets. Since our project only experimented with datasets that were labeled a few years ago, the question of whether experimenting with data gathered recently can affect the performance of hate speech classification remains an open one.

References

[1] Azalden Alakrot. "Towards Accurate Detection of Offensive Language in Online Communication in Arabic". In: *Procedia Computer Science*. 2018, pp. 315–320. URL: 10.1016/j.procs.2018.10.491.

[2] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *Proceedings of the 11th International AAAI Conference on Weblogs and Social Media*. ICWSM '17. Montreal, Canada, 2017. URL: https://data.world/thomasrdavidson/hate-speech-and-offensive-language.

[3] Björn Gambäck and Utpal Kumar Sikdar. "Using Convolutional Neural Networks to Classify Hate-Speech". In: *Proceedings of the First Workshop on Abusive Language Online*. Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 85–90. DOI: 10.18653/v1/W17-3013. URL: https://www.aclweb.org/anthology/W17-3013.

[4] Yanrong Ji. "DNABERT: Pre-Trained Bidirectional Encoder Representations from Transformers Model for DNA-Language in Genome". In: 2020. URL: 10.1101/2020.09.17.301879.

[5] Ji Ho Park and Pascale Fun. "One-Step and Two-Step Classification for Abusive Language Detection on Twitter". In: *Proceedings of the First Workshop on Abusive Language Online*. 2017. URL: 10.18653/v1/w17-3006.

[6] Amir H Razavi. "Offensive Language Detection Using Multi-Level Classification". In: *Advances in Artificial Intelligence Lecture Notes in Computer Science*. 2010, pp. 16–27. URL: 10.1007/978-3-642-13059-5_5.

[7] Bertie Vidgen and Leon Derczynski. "Directions in Abusive Language Training Data: Garbage In, Garbage Out". In: 2020.

[8] Zeerak Waseem. "Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter". In: *Proceedings of the First Workshop on NLP and Computational Social Science*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 138–142. URL: http://aclweb.org/anthology/W16-5618.

[9] Zeerak Waseem and Dirk Hovy. "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter". In: *Proceedings of the NAACL Student Research Workshop*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 88–93. URL: http://www.aclweb.org/anthology/N16-2013.

[10] Rich Zemel et al. "Learning Fair Representations". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 325–333. URL: http://proceedings.mlr.press/v28/zemel13.html.