Detecting Hate Speech

Mick Leungpathomaram

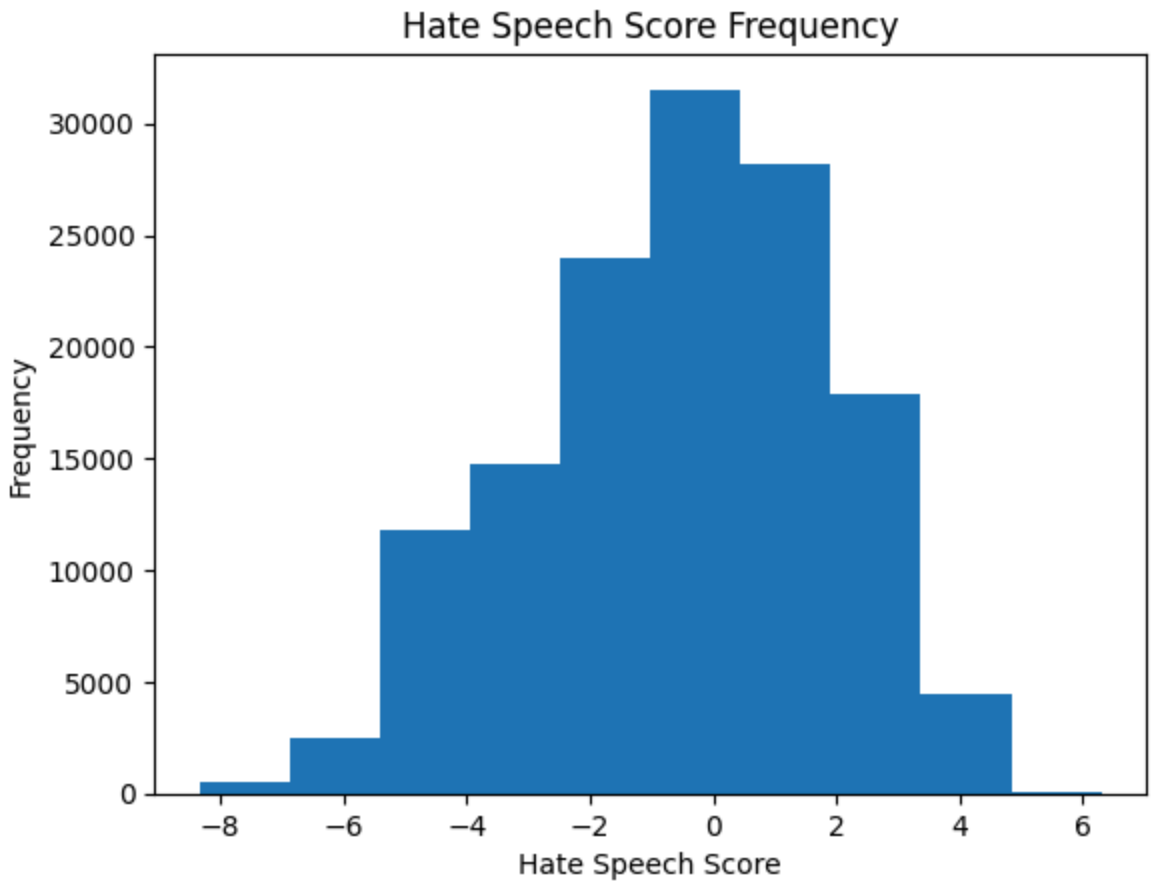
Introduction

Social media platforms are used by millions daily, with the number only growing as time goes on. However, hate speech towards groups or individuals is an unavoidable problem. With the sheer amount of activity on popular platforms, manual moderation is inadequate to detect and remove such remarks. Automated filters, while helpful in flagging potentially hateful comments, lack the nuance required to detect whether a comment truly is hate speech, and still require manual review. Machine learning offers promising potential for hate speech detection by capturing complex linguistic patterns and contextual nuances. This report outlines the development and evaluation of various deep learning models designed to accurately identify hate speech in text. Our approach includes comprehensive preprocessing, feature engineering, and the application of a variety of machine learning techniques. We focus on enhancing detection accuracy, minimizing false positives and negatives, and ensuring model robustness across different contexts. By improving automated hate speech detection, this work has the potential to create safer and more inclusive online environments.

Data

This project used the UC Berkeley hate speech dataset.[1] This dataset contains roughly 39,565 social media comments taken from Youtube, Reddit, and Twitter. Each comment is reviewed by an annotator, who gives the comment a hate speech score based on the characteristics of the comment, such as calls for violence, dehumanization, or bias. In cases where multiple annotators reviewed the same comment, each review is its entry in the dataset. As such, the database contains 135,556 entries in total. A score greater than 0.5 is considered to be hate speech, less than -1 is supportive speech, and any score between -1 and 0.5 is considered neutral or ambiguous. The histogram below shows the score distribution for the entire dataset. As we can see, the hate speech scores follow a roughly normal distribution, centered around -0.567.

[1]Chris Kennedy, "Measuring Hate Speech," Measuring Hate Speech, October 11, 2019, https://hatespeech.berkeley.edu/.



Mean	-0.567
Median	-0.334
Standard Deviation	2.380

The dataset also includes label features describing the type of hate speech (i.e. insult, genocide, etc.), and the target group, but the only relevant parts of the dataset for this report are the comments and the hate speech score. More information about how the hate speech scores were determined can be found in the paper *Constructing interval variables via faceted Rasch measurement and multitask deep learning: a hate speech application.*[2] The dataset came with URLs and continuous whitespace removed, but otherwise had no pre-processing.

[2] Chris J. Kennedy, Geoff Bacon, Alexander Sahn, and Claudia von Vacano, "Constructing Interval Variables via Faceted Rasch Measurement and Multitask Deep Learning: A Hate Speech Application," arXiv preprint arXiv:2009.10277 (2020).

The data was split into training and testing sets, using a 0.5 split. To evaluate model performance, we looked at the RMSE(Root Mean Square Error) in predicting the train and test sets.

Methodology:

Pre-Processing:

During pre-processing, certain characters were changed or removed to reduce noise. Mentions and tags were replaced with "@", "#", and "url", respectively. Non-printable characters, such as emojis or non-unicode values were replaced with "~". Stopwords were also removed, using the Natural Language Toolkit's stopwords list. During initial tests, punctuation was also removed, but this step was removed from pre-processing as it was found to decrease model performance.

Feature Engineering:

The data is predicted using a variety of features, as seen below. Note that although a hate speech score between -0.5 and 1 is considered 'neutral', trial and error found that relaxing the threshold as seen in **Support_common** and **Hate_common** resulted in a small yet consistent improvement in network performance.

- **Support_common:** The number of words in the comment that commonly show up in comments with a hate score less than 0.
- **Hate_common:** The number of words in the comment that commonly show up in comments with a hate score greater than or equal to 0.
- **Neutral_common:** The number of words in the comment that commonly show up in comments with a hate score between -0.5 and 1.
- **Support_unique:** The number of words in the comment that are much more common to find in support comments.
- **Hate_unique:** The number of words in the comment that are much more common to find in hate comments.

- **Has_url:** The number of links the comment contains
- **Num_mentions:** The number of "@" symbols the comment contains
- **Num_tags:** The number of "#" symbols the comment contains
- **Non_alphnum_prop:** The proportion of non-alphanumeric characters in the comment
- **Length:** Length of the comment.

The 'hate_unique' feature was found by creating dictionaries containing the word frequencies for 'hate' and 'support' groups, and subtracting the latter from the former. The 'support_unique' feature was found in a similar manner.

Models:

Several different machine learning techniques were used in the creation of the models. All models were trained using cross validation, and gridsearch. Linear Regression, Random Forests and Support Vector Machines were used during initial attempts, but were discarded due to consistently poor performance relative to other models. K-Nearest Neighbors, XGBoost and Decision Trees all had comparatively similar performance. The best models of each type were used in Voting and Stacking Ensembles. While Voting Ensembles had a poor performance, the Stacking Ensemble was consistently the best performing model. In all cases, using a Decision Tree for the final model resulted in the best performance. A maximum depth of 5 was used, as although performance continued to improve(up to a point), the gains were minimal and not worth the increase in runtime.

Although outside the original scope of the project, Recurrent Neural Networks are well-suited to predicting sequential data, and a RNN model likely would have strong results. Initial attempts to create such a model were hindered by the memory requirements for one-hot encodings. This was resolved by reducing the train and test sets to 10% of the total database each. Unfortunately, the model created fails to learn anything due to an unknown issue with the code.

Results

All RMSE scores have been rounded to the thousandths place. While some models, such as XGBoost and K-Nearest Neighbors used negative RMSE, the negative sign has been removed for consistency. The code used can be found here.

All Features:

	Decision Tree	XGB	KNN	Stacking Regressor
Train RMSE	1.547	1.400	1.52	1.148
Test RMSE	1.546	1.358	1.510	1.349

Without any of the "common" features.

	Decision Tree	XGB	KNN	Stacking Regressor
Train RMSE	1.607	1.451	1.559	1.451
Test RMSE	1.604	1.414	1.542	1.414

Without any of the "unique" features.

	Decision Tree	XGB	KNN	Stacking Regressor
Train RMSE	1.548	1.400	1.522	1.148
Test RMSE	1.546	1.358	1.512	1.362

With all features, but without stopword removal

	Decision Tree	XGB	KNN	Stacking Regressor
Train RMSE	1.650	1.458	1.621	1.176
Test RMSE	2.057	2.001	1.937	1.966

With all features, but without punctuation removal

	Decision Tree	XGB	KNN	Stacking Regressor

Train RMSE	1.664	1.467	1.616	1.184
Test RMSE	1.662	1.430	1.614	1.438

Results from the *Constructing interval variables via faceted Rasch measurement and multitask deep learning: a hate speech application* paper for comparison:

Prediction Algorithm	**Outcome Representation**	**Trained Parameters**	**CV RMSE**	**CV MAE**	**CV Corr.**	**Training Comments**
Outcome mean baseline	Continuous	1	2.061	1.964	0.000	42,000
Google Jigsaw Identity Attack	Binary + calibration[a]	Unknown	1.868	1.914	0.423	∼100,000
Google Jigsaw Toxicity	Binary + calibration[a]	Unknown	1.557	1.904	0.655	∼100,000
In-domain embeddings + Bi-LSTM	Continuous	9,121	1.342	1.033	0.763	37,817
Universal Sentence Encoder FE	Ordinal + IRT	33,539	1.286	0.990	0.780	37,817
Universal Sentence Encoder FE	Continuous	32,897	1.245	0.969	0.792	37,817
BERT-Large WWM FT	Continuous	335,142,913	1.142	0.893	0.824	28,571
RoBERTa-Large FT	Ordinal + IRT	355,360,769	1.126	0.858	0.841	28,571
RoBERTa-Large FT	Categorical + IRT	355,360,769	1.117	**0.849**	**0.843**	28,571
RoBERTa-Large FT	Continuous	355,360,769	**1.078**	0.861	0.839	28,571

Discussion

In the context of this problem, the RMSE denotes the average difference between the predicted hate speech score and the actual hate speech score of a comment. The test data had a mean of -0.567 and a standard deviation of 2.381. As such, the baseline RMSE for the test set is 2.381, as a model that simply guesses the mean hate speech score for every comment will be off by 2.381 on average.

Our models showed considerable improvement over this baseline. The Stacking Regressor, in particular, demonstrated strong performance across all feature sets. When using all features, it achieved a test RMSE of 1.349, outperforming non-ensemble models like Decision Trees, XGBoost and K-Nearest Neighbors by a substantial margin.

In some instances, the test RMSE was lower than the train RMSE. For example, the XGBoost model with all features had a train RMSE of 1.400 and a test RMSE of 1.358. This

could be due to several factors, including the potential for the training data to contain more noise or variability, or the test set being more representative of the underlying distribution the model is trying to learn.

On the other hand, removing "unique" features yielded slightly better results compared to removing "common" features, indicating that these unique features might contribute more valuable information for the detection task.The importance of preprocessing steps like stopword and punctuation removal was also highlighted by the results. Without stopword removal, the test RMSE increased markedly to 1.966 for the Stacking Regressor. Similarly, not removing punctuation slightly increased the test RMSE to 1.438, indicating these preprocessing steps enhance model accuracy.

Our best model was the Stacking Regressor using all features and pre-processing functions, with a test RMSE of 1.349. For comparison, the best-performing model from the UC Berkeley paper was the continuous RoBERTa-Large FT model, with an RMSE of 1.078. While the Stacking Regression model did not surpass this, it still outperformed the benchmark Google Jigsaw models by a wide margin, and did almost as well as the In-domain embeddings + Bi-LSTM model. Overall, the results indicate that our machine learning models, especially the Stacking Regressor, can effectively detect hate speech with a high degree of accuracy. These findings underscore the potential of advanced machine learning techniques to contribute to creating safer online environments by accurately identifying and mitigating hate speech.

References

Kennedy, Chris. “Measuring Hate Speech.” Measuring Hate Speech, October 11, 2019. https://hatespeech.berkeley.edu/.

Kennedy, Chris J., Geoff Bacon, Alexander Sahn, and Claudia von Vacano. "Constructing Interval Variables via Faceted Rasch Measurement and Multitask Deep Learning: A Hate Speech Application." *arXiv preprint* arXiv:2009.10277 (2020).