Toxic Comments Analysis and Removal

Definition

Platforms that aggregate user content are the foundation of knowledge sharing on the Internet. Blogs, forums, discussion boards, and, of course, Wikipedia. But the catch is that not all people on the Internet are interested in participating nicely, and some see it as an avenue to vent their rage, insecurity, and prejudices.

Wikipedia runs on user generated content and is dependent on user discussion to curate and approve content. The problem with this is that people will frequently write things they shouldn't, and to maintain a positive community this toxic content and the users posting it need to be removed quickly. But they don't have the resources to hire full-time moderators to review every comment.

The dataset has six labels that represent subcategories of toxicity, but the project is going to focus on a seventh label that represents the general toxicity of the comments.

Problem Statement

The goal is to create a classifier model that can predict if input text is inappropriate (toxic).

1. Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments.
2. Create a baseline score with a simple logistic regression classifier.
3. Explore the effectiveness of multiple machine learning approaches and select the best for this problem.
4. Select the best model and tune the parameters to maximize performance.
5. Build a the final model with the best performing algorithm and parameters and test it on a holdout subset of the data.

Metrics

Unfortunately for the problem, but fortunately for the Wikipedia community, toxic comments are rare. Just over 10% of this dataset is labeled as toxic, but some of the subcategories are extremely rare making up less than 1% of the data.

Because of this imbalance, accuracy is a practically useless metric for evaluating classifiers for this problem.

The Kaggle challenge based on this dataset uses ROC/AUC, or the area under a receiver operating characteristic curve, to evaluate submissions. This is a very generous metric for the challenge, as axes for the curve represent recall (a.k.a. sensitivity), the ratio of positive predictions to all samples with that label, and specificity, the ratio of negative predictions to all negative samples. This metric would work well if the positive and negative labels were relatively even, but in our case, where one label represents less than a third of a percent of the data, it's too easy to get a high score even with hardly any true-positive predictions.

Instead, I propose using an F1 Score, which severely penalizes models that just predict everything as either positive or negative with an imbalanced dataset.

Recall, as mentioned earlier, is the ratio of true positive predictions to positive samples. Precision, on the other hand, is the ratio of true positive predictions to the sum of all positive predictions, true and false.

Each gives valuable insight into a model's performance, but they fail to show the whole picture and have weaknesses where bad models get high scores. Predicting all positive values will bring recall up to 100%, while missing true positives will be penalized. Precision will harshly penalize false positives, but a model that predicts mostly negative can achieve a high precision score whether or not the predictions are accurate.

The F1 score is a harmonic average between precision and recall. This combines the strengths of precision and recall while balancing out their weaknesses, creating a score that can fairly evaluate models regardless of dataset imbalance.

My justification for focusing on any_label as the target is that distinctions between the specific labels are relatively ambiguous, and that there is greater value focusing on general toxicity of a comment to more reliably flag it for review. This will reduce the workload of moderators who will ultimately be making the final call, and the specific category more relates to the consequences for the commenter rather than whether or not the comment should be deleted.

Algorithms and Techniques

As a natural language processing problem, is a classification task that involves high dimensionality data. I will vectorize the data and test multiple classification algorithms.

I will vectorize the text data using the term frequency – inverse document frequency (tf-idf) statistic. This technique takes into account not only the frequency of words or character n-grams in the text, it also takes into account the relevancy of those tokens across the dataset as a whole. The inverse document frequency reduces the weight of common tokens while boosting the weight of more unique tokens. I will establish a benchmark for performance with the top 10,000 words, and the number of tokens and the mix of words and character n-grams will be a parameter to tune for higher performance later on.

I will also create a number of engineered features containing various attributes of the comment text, such as average word length, capitalization, and number of exclamation points. I will run the benchmark test without these features and experiment with them to optimize the solution.

With the benchmark vectorization and features, I will experiment with multiple algorithms with default parameters to determine the most effective approach to the problem. The models I will use are:

- Logistic Regresssion (Benchmark)
- Multinomial Naive Bayes
- Support Vector Machine
- Support Vector Machine with Naive Bayes Features
- Light GBM

Recurrent neural networks work well on this problem and top Kaggle leaderboards, but I think deep learning approaches are too resource intensive for an algorithm that has to run instantly every single time a comment is posted on one of the most popular websites on the Internet. A major requirement if this were a real-life business problem is efficiency. Additionally, adapting the model to secondary features would require stacking, which is a

messy solution that increases the complexity of both training and predicting. I've used model stacking in Kaggle competitions before, and it comes at the expense of efficiency.

One more consideration is transparency, and this is the biggest aspect of the decision not to use neural networks for this application. I want to have the ability to easily audit the model to ensure that it isn't picking up bias around race, gender, sexual orientation, culture, or unforeseen categories from the curators of the data. SVM, Naive Bayes, and LightGBM will make it much easier for a third party to analyze the impact of specific features on the model and make appropriate adjustments to combat bias.

I predict that there will be a toss-up between Logistic Regression and Support Vector Machines. Naive Bayes may have strong performance due to the dramatic difference between the the frequency distributions of the vocabulary between comments.

Support vector machine models are unique in that they find the boundaries between classes by looking at the distances between the separating line and the nearest point for each class, and are not effected by outliers. In this project I will be using the linear kernel. The kernel trick with support vector machines involves projecting data with complex or nonlinear boundaries into a higher dimension where they are linearly separable and drawing a hyperplane between them using a linear model algorithm. Imagine a 2D problem where you have a circular boundary between two classes and need to draw your decision boundary with linear regression. With the kernel trick you'd be projecting that flat dataset into the dimensions and ideally the circle would look more like a hill, allowing you to cut through it with a 2D plane and establish a linear boundary. The linear kernel is the most efficient of the kernels and is very resistant to overfitting, which I think may be an issue with the dataset.

LightGBM is a tree-based ensemble model that is trained with gradient boosting. The unique attribute versus other boosted tree algorithms is that it grows leaf-wise rather than level-wise, meaning that it prioritizes width over depth. Boosted tree can be confused with forest models, but there is an important distinction. Where forest models like Scikit-Learn's RandomForest use an ensemble of fully developed decision trees, boosted tree algorithms use an ensemble of weak learners that may be trained faster and can possibly generalize better on a dataset like this one where there are a very large number of features but only a select few might have an influence on any given comment.

LightGBM is not a high performer on Natural Language Processing tasks, but I think it's worth trying a treebased model here due to the very large disparity between toxic and clean comments with the engineered features and top 30 words. There may be some great, reliable splits here. The reason it doesn't perform well on NLP tasks is that it requires dense input data, and the nature of vectorizing text creates absolutely massive matrices filled mostly with zeros. It also has a relatively small number of features that it can focus on, while the vectorized text may contain over 20,000 features.

Support Vector Machine with Naive Bayes Features or Multinomial Naive Bayes may perform even better. The paper *Baselines and Bigrams: Simple, Good Sentiment and Topic Classification*[3] experiments with this algorithm on a variety of types of datasets and found that Support Vector Machines performed exceptionally well at sentiment analysis on datasets with lengthy texts, such as full-length movie reviews. The same paper suggest that Multinomial Naive Bayes works better on snippets of text. Assuming an average sentence length of 75 to 100 characters, the Wikipedia dataset used in this project averages 2-4 sentences, which is more than a snippet but not exactly lengthy. Naive Bayes with SVM features was found to interpolate between Naive Bayes and Support Vector Machines, combining strengths of each.

After establishing a baseline score for each model, I'll choose the best one or two, depending on how close they are, and tune the vectorization and model parameters to optimize performance.

Model Evaluation and Validation

The model has been trained, tested, and optimized using training and test subsets of the data. I will use an unseen holdout subset of the data to evaluate the model.

The F1 Score on the holdout data is 0.8072.

Because it's performance is similar to the results obtained in the previous stage, I can confidently say that the model will generalize well to unseen data. Because it is a real world dataset with a huge variety of comments discussing a diverse range of topics and covering situations from informative posts to flame wars, this is probably one of the better scenarios for training a model on text.

Justification

The final model offers a significant performance boost over the benchmark linear regression model, about 11%. So far we've been talking in the abstract about F1 Scores, but now let's dig into the real world performance and what those numbers actually mean.

This model has 96% accuracy. Now on the surface, that sounds great. But since this is a highly imbalanced dataset, that doesn't mean a lot. In fact, if I had just created a model that predicted "0" for every single item, it would get an accuracy of 90%.

The real metric of how well the model performed at predicting a toxic comment is recall. This model achieved a recall score of 0.74, which means that it correctly 74% of the actual toxic comments as toxic. That may seem low, but there's a catch. If we predict every result as "1," we'll get 100% recall.

As discussed before, the F1 Score provides a target that helps a model find the nuance in an imbalanced dataset between catching the positive results without focusing on them to a point where the usefulness of the model suffers. A confusion matrix can illustrate the concept of balancing true positives and true negatives, as well as accuracy, recall, and precision.

Overall, I do believe that this model is robust enough for this application and it offers a large advantage over both the standard approach of human flagging for review (though I wouldn't eliminate that as a feature) and an out-of-the-box model.