**Content Moderation Using Jigsaw’s Toxic Comment Classification Challenge**

Ethan Carty (emc21034), Adam Chao (adc21010), Noor Sharaf (nos21008), Connor Shaw (cos21006), Will Shostak (wis19003)

**GitHub:** <https://github.com/WillSho1/contentModeration>

**Introduction**

The detection of toxic comments is crucial in maintaining a positive, ethical, and safe social media experience; however, ever-changing slang and class imbalance make the task complex. To understand this problem and its ethical implications, we decided to develop a model that learned how to classify comments or other text content as harmful. The model was trained and tested using an expired competition dataset on Kaggle, Jigsaw’s Toxic Comment Classification Challenge. The entire dataset includes nearly 224,000 hand-labeled comments from Wikipedia talk pages. Each comment is labeled with the following overlapping labels: toxic, severe toxic, obscene, threat, insult, and identity hate. Through this project, we hoped to better understand the ethical implications of censorship on the internet, and the best ways to approach this problem.

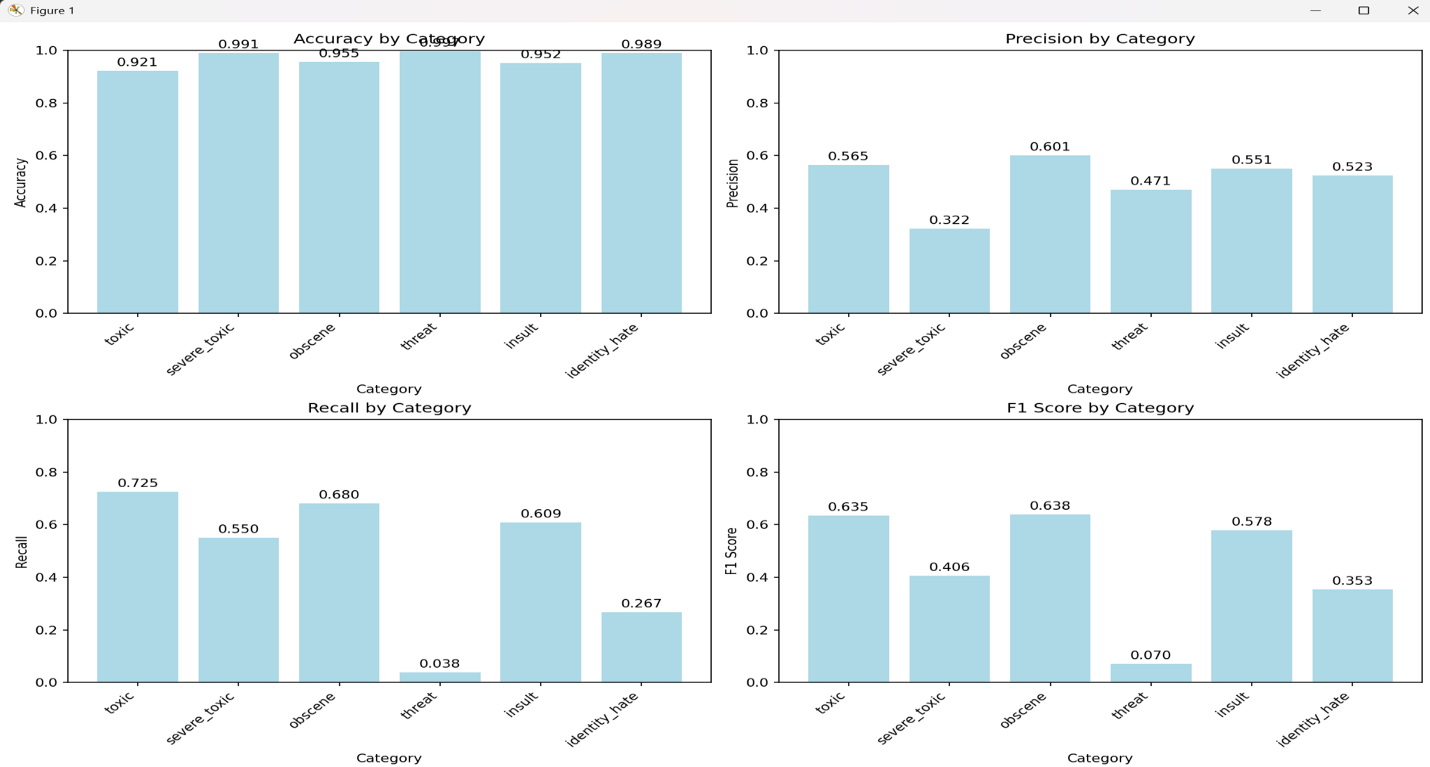
**Methodology and Implementation**

To train a Naïve Bayes model on the dataset, the data must first be cleaned and vectorized. Using pandas, the csv is loaded into the script as arrays, and the test labels are merged with the test comments; they were originally separated for the purpose of the competition. Some comments that were not scored in the competition were labeled with –1, so these were removed from our data. To clean the text, we applied our cleaning function to all comments, which removes special characters and converts to lowercase. For vectorization, we elected to use sklearn’s TfidVectorizer, which has a built-in function to remove stop words. We removed further irrelevant words using minimum and maximum document frequency, removing words that appeared in more than 70% of the comments and words that appeared in fewer than 5 comments to prevent over-complexity and overfitting. In the vectorizer, we also applied sublinear term frequency scaling and l2 normalization. We also chose to include unigrams and bigrams, which increases the size of our vocabulary, but also introduces word associations that may be important in identifying toxic comments. The vectorizer also allowed us to limit the maximum size of the vocabulary. All of these parameters were tuned in order to improve the accuracy of our model.

To train our model, we chose Naive Bayes rather than logistic regression due to our members familiarity with Naive Bayes and its performance in text classification problems. We initially started with a Gaussian Bayes model, which even after fine tuning, did not produce adequate results. We were struggling to get accuracy scores above 60%, with precision, recall, and F1 all being considerably low. We then decided to take a different approach and went with a Multinomial Naive Bayes model. This model assumes the data follows a multinomial distribution which was much closer to what our data showed. After implementing the model, we received much better results.

**Results and Discussion**

Our results for the Multinomial Bayes started as mediocre. We had high accuracy scores >90% for most categories, but some measurements from our confusion matrix such as recall, precision, and F1 score were lackluster. For example, our evaluation metrics for severe toxic showed an accuracy of 99% but a precision of only 50%, recall of 6%, and F1 score of 11%. Whilst our accuracy was successful, the other scores being so low implied that our model could, more or less, always predict not severely toxic. One can see this with the confusion matrix as well, 64,000 comments were not severely toxic and our model only achieved 56 correct severely toxic comments with 311 false negatives and 53 false positives. This shows a heavy bias towards predicting not severely toxic and didn’t make for that great of a model. A lot of this bias stemmed from the fact that of the 150,0000 comments in the dataset, only a small portion had labels of severe toxic or threat. Furthermore, only 211 comments were truly threats and the other 63,000 were not. Our model didn’t predict a single comment was a threat and so with the very limited training data this made sense. We then decided that we needed to go back and perform some feature engineering to get better recall, precision, and F1. We went back to our text processing and reduced the amount of features it had from 20,000 to 5,000 for one. This is because of the sheer number of comments (150,000). Furthermore, we changed the data processing such that we increase max\_df and decrease min\_df variables to avoid losing too much informative words, and take into account our more niche classifications, like identity hate. Our model was now able to correctly identify some threat comments as compared to the previous iteration, and saw higher scores across the board for the other categories. The final results we were able to achieve are below.

**Conclusion**

The project demonstrates that the simple TF-IDF + Multinomial Naïve Bayes pipeline can identify much of the toxic content. However, raw accuracy fails to tell the whole picture. The filter missed some of the severe-toxic and threat comments—not common but still concerning—until we rebalanced classes and changed decision boundary levels. These adjustments illustrate two important issues for practical use: the need to fix features for the training imbalance and sustain the model as the language changes. When both conditions are met, the baseline becomes a more robust and reliable moderation system.

**Ethical Considerations**

Our model flagging false positives can have multiple repercussions for the user experience. For one, flagging false positives can lead to undeserved bans and sanctions on users. A user might then be left confused by a ban or suspension after a normal comment was flagged as toxic. Additionally, the flagging of false positives can leave users confused as to what is okay and what is not okay to say. Should they read the terms and conditions of a site, have a good understanding of them, and then get flagged for commenting something falls well within the terms of use. This can be quite frustrating especially if it happens more than once to a user, or to a user where their income may come from this social media account. It’s negative overall for a user’s experience and can lead to people abandoning the website/platform altogether if left unchecked. Lastly, the flagging of false positives can take away resources that are actually needed for false negatives, where those users need to be held accountable. This model was focused more on false positives than false negatives, but it was still an issue, and there being this many false positives frustrates users who aren't doing anything wrong.

If toxic comments from certain groups are more frequently flagged there’s a few things we could do to mitigate this bias within our model. When looking at the data we could diversify the dataset more and specifically include non toxic examples from the frequently flagged groups. This could help to reduce machine related bias toward that group and improve F1. We could also implement balanced sampling help these groups or even think of reweighing the data to help reduce the times that group is flagged. There’s also the option of implementing adversarial debiasing where we would train two models and have a predictor and adversary. This forces the predictor model to make fairer decisions as it tries not to be influenced by the adversary. Another option is implementing some fairness constraints and introducing separate classifiers that allow for the presence of reclaimed terms and in-group language patterns. Lastly, we could make sure that there is an appeal process for sanctions, human review of borderline cases, and be open to feedback so that we can make changes where needed.

Platforms should prioritize inclusivity over accuracy when it comes to text classification. A large issue is political statements. If a dataset is biased towards one part of the political spectrum, then a social media app has a higher chance of being biased towards one side. Accuracy is defined as how many comments the model correctly identifies as a certain label, and inclusivity is the difference in performance between individual groups. An accurate model is always something one should strive for, but if the model is incorrectly identifying an entire group, but is more accurate elsewhere, then that simply is not enough. Sacrificing some accuracy for better results across the board is something that would improve the overall user experience. An example being that if a model was very accurate for 5 of the 6 toxic labels that we have in our case. If that 6th label is a threat and the model can’t really identify those as well, then those who get threatening comments are going to feel left out and as if they aren’t cared about. Sacrificing some accuracy in the other 5 labels to then get the model to perform better on threat comments would be a net benefit to the user base, as now those threat comments won’t slip by as they were before. Chasing accuracy, whilst important, leaves one’s model open to some potential severe bias toward a certain group, whereas prioritizing inclusivity first will give a person an all-around better model.

Overall, due to the simplicity of our model, we are happy with our results. We are also delighted that we learned a lot about internet ethics, machine learning, and the best engineering practices in the process. Despite automatic internet censorship being around for decades, we are excited to have dipped our toes into the water and have experienced what it is like to design machine learning-powered policies.