Toxic Comment Classification

INTRODUCTION AND DATASET

We based our project on Kaggle's Toxic Comment Classification Challenge. Our goal was to build a model to classify toxic comments into six classes: *toxic, severe_toxic, obscene, threat, insult,* and *identity_hate*. The dataset comes from Wikipedia's talk page edits, in the form of three zip files named "train.csv", "test.csv", and "test_labels.csv".

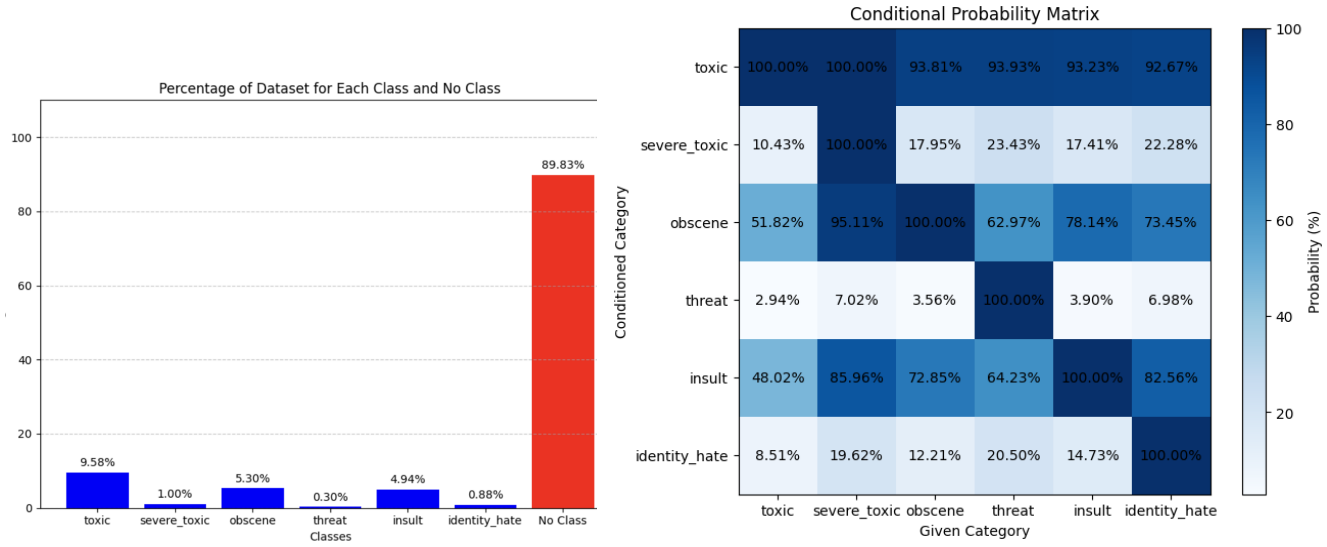

Fig 1. *Data visualizations*

1. PRE-PROCESSING (FEATURE ENGINEERING)

1.1 Re-formatting the test.csv file

To use the test data for validation, we reformatted the two test files (test.csv and test_labels.csv) to match the format of train.csv. We created a function *formatTestData(testcsv_path, test_labelscsv_path)* which removes all validation data that was unused by the competition, indicated by all class values being -1.

1.2 Cleaning Text Data

We simplified our data by removing “stop words” (a, I, of, the, and, etc). This helps our model by making training data less complex thus reducing overfitting. We also applied stemming to the terms. Stemming is frequently used in language processing as it allows for related words to be represented in the same way. We used Natural Language Tool Kit for many of the processes in cleaning the data.

1.3 Manipulation of Training Data

Upon predicting results with our first model, we noticed that our model’s recall was poor due to the overwhelming amount of unlabeled comments and relatively few toxic comments, especially for the categories *severe_toxic, threat,* and *identity_hate*. So in an attempt to improve our model we attempted two strategies.

Our first strategy was to prune some of the unlabeled data. The percentages we used for pruning were 50%, 60%, 70%, 80%, and 90% of the unlabeled data. This changed the proportion of labeled to unlabeled data, as shown below. Before pruning, unlabeled data made up 89.83% of our dataset, as shown in **Fig 1**. Shown below are the proportions of are data at 50% pruning and 90% pruning.

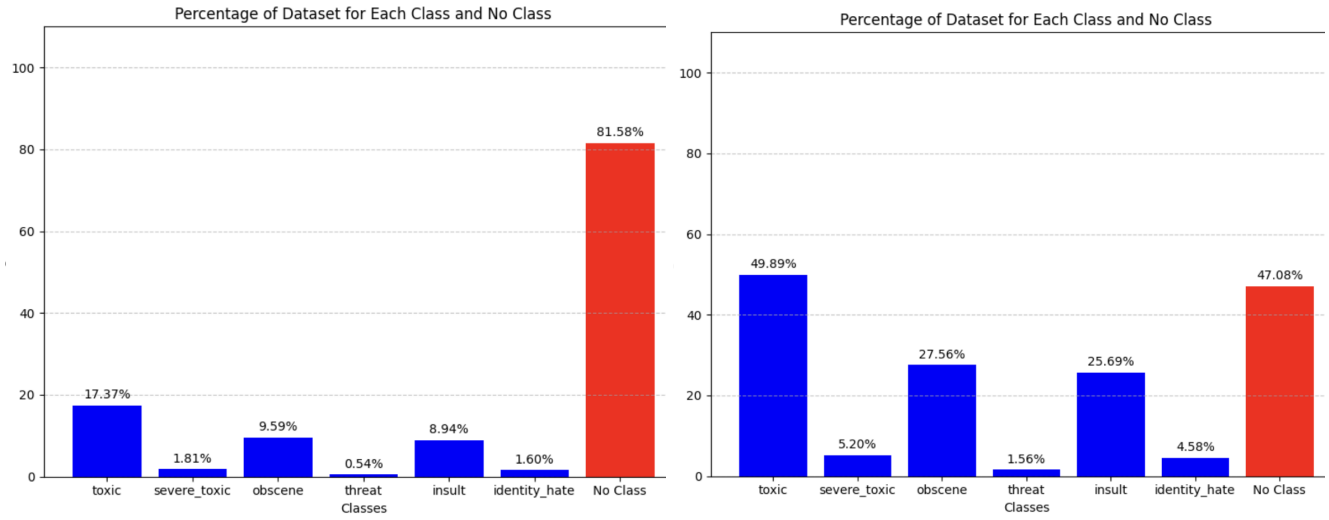

Fig 2. *50% dataset pruning vs 90% pruning*

Our second strategy was to generate new toxic comments based on our existing training data.This was done by taking each toxic comment and shuffling the words in that comment. The new shuffled comment was given the same label as its parent comment. This doubled the size of our toxic comment data.

1.4 Derived features

As mentioned above, we are only provided one feature within the raw dataset. For our dataset, it simplifies down to a feature vector of comment strings, mapping to a vector just ONE class label, rather than all 6 of them (later on we'll touch on how this will be repeated on all 6). Given an NLP problem like this one, we want to convert our data into a proper representation for our learners.

2. MODELS/METHODOLOGIES

LSTM-DenseNet

Long-Short Term Memory networks (LSTMs) are commonly used in language processing for their ability to process long, sequential data. LSTMs are based on the Recurrent Neural Network (RNN) architecture. RNNs are very useful for solving problems that have sequential information, as they can connect earlier parts of the network to later parts. While RNNs are excellent at handling short-term dependencies, RNNs can struggle to learn long-term dependencies. LSTMs aim to solve this long-term dependency problem, allowing for the processing of very long sequences where information from much earlier can impact information later on.

The Densely Connected Convolutional Network (DenseNet) is a form of Convolutional Neural Network

(CNN) which are capable of handling a very large number of layers. CNNs are useful for identifying important features, regardless of their position in the input. A CNN can solve a language processing problem by identifying specific words or strings that may commonly align with a specific categorization.

For our model, we combined an LSTM with a Densenet. Our model consisted of an embedding layer, followed by an LSTM layer with ten output features. This LSTM fed into a Dense layer with six outputs for the six labels. The Dense layer used a dropout value of 0.2. While DenseNets greatest strength is allowing for a very deep network, our computers were unfortunately not powerful enough to handle a very complex network. Ideally, our LSTM would have output many more features and we could have had several Dense layers. Given more capable hardware equipped with a GPU, we would have experimented with more complex architectures.

3. PERFORMANCE VALIDATION

Since so much of our dataset had no labels, just using accuracy as a measurement of performance would be misleading. Even after pruning non-labeled, data, a model could appear to be performing quite well just by always predicting no label, if we measured only using accuracy. Therefore, recall and precision were better indicators of the quality of our models. The results of our models are shown below.

LSTM-Dense

We had five data pruning levels: 50%, 60%, 70%, 80%, and 90%. For each of these levels of pruning, we created had datasets: one with and and one without our generated data. This gave us a total of ten datasets to try training our models on. The results of our ten models are shown below:

	Cleaned Data			Cleaned Data with added Generated Data		
Pruning Level	Accuracy	Precision	Recall	Accuracy	Precision	Recall
50%	74.78%	5.34%	33.71%	56.89%	2.37%	25.72%
60%	64.19%	2.77%	24.66%	50.62%	3.28%	42.07%
70%	59.80%	1.88%	18.72%	39.38%	1.75%	27.07%
80%	51.05%	3.05%	38.52%	33.44%	2.14%	36.87%
90%	34.60%	2.21%	37.47%	24.69%	3.70%	75.14%

Fig 4. *Results from LSTM-Dense models.*

Since we were using multi-class classification, this model was not able to output more than one label at a time. This severely impacted the model's metrics, since many comments had more than one label.

4. FURTHER OPTIMIZATION AND VALIDATION IDEAS

The idea of using a binary classifier as part of our model is worth further exploration. Some labels are

highly correlated with other labels, for example, all *severe_toxic* comments are also *toxic*. Therefore, a comment that is likely to be *severe_toxic* is also very likely to be *toxic*, and a comment that is *toxic* has a higher chance of being *severe_toxic*. Future work could explore how to incorporate such information into the model.

Our LSTM-Dense models might perform better given more layers with larger outputs. The first improvement that should be made to the model would be to make the model able to output more than one label at a time. This would certainly drastically improve the metrics of this model.

Our data manipulation techniques were also relatively primitive. It could be worth exploring better ways of expanding our data for toxic comments, especially the rare label types, such as *threat*.

5. Contributions

Cole Hajek was responsible for writing the code for the logistic regression models. He was also the primary author of the code that cleaned and pre-processed the data. Jacob Berkel was the primary author of the code related to the LSTM-Dense model and the code that generated new toxic comments. He also contributed to the code which cleaned the data.