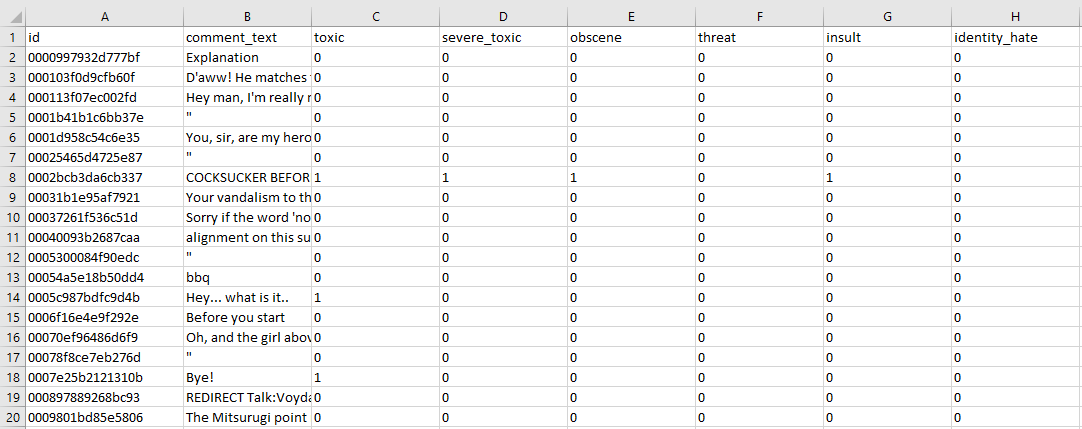
**Introduction**

Our data mining task was to create an algorithm that categorizes Wikipedia comments based on whether they are toxic, and which category of toxicity they fall under. The motivation behind choosing this problem was to create an algorithm for social media platforms so they can hide toxic comments once they are labeled and in turn reduce negativity and cyber-bullying on social media. For us personally, we chose this task because we have both seen the effects of toxic comments on social media. We hope that our algorithm is able to decrease negative comments online and that this leads to improved mental health for all who have been experiencing cyber-bullying and harassment online. One question that we wanted to answer through this project is what are the most recurring words or phrases that are in each category of toxic comments. We also wanted to compare the number of comments in each category to see if a type of toxicity was more prevalent than others. One of our biggest challenges was cleaning the data so that our algorithm was not affected by punctuation and words without meaning on their own. Once this was accomplished, we were able to create an algorithm that predicted the toxic category of the comment with 99.21% accuracy.

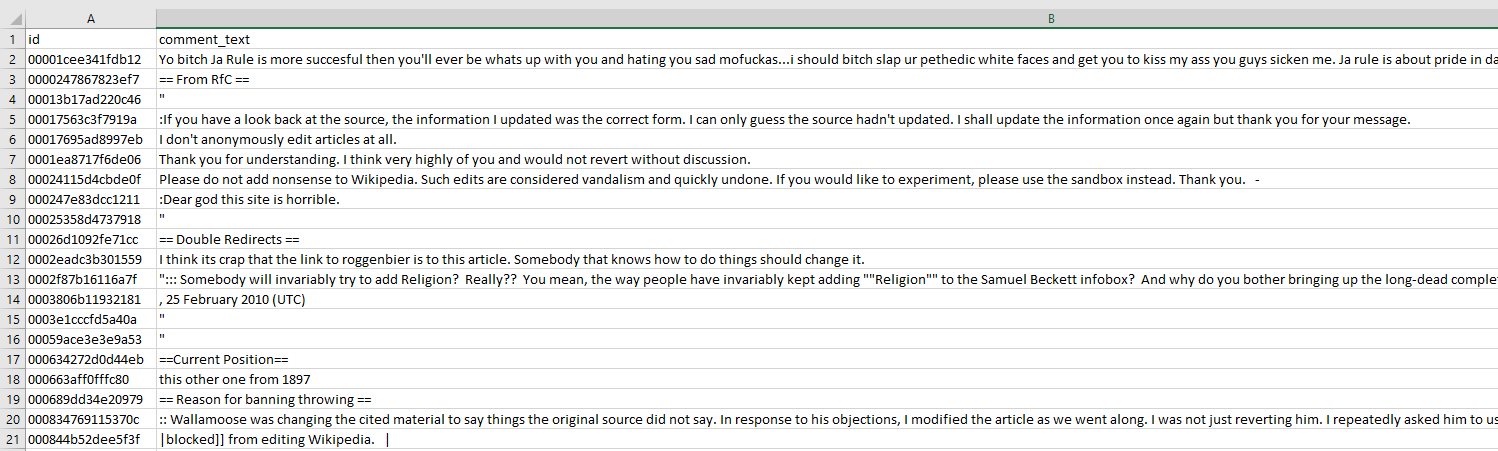
**Data Mining Task**

In order to complete our data mining task, we first had to extract the comments from the training and testing .csv files. The three files divided the training data set, the testing set comments and comment IDs, and the testing set IDs and labels. Our algorithm then took this data and predicted which of the categories the comments in the test data would fit into.

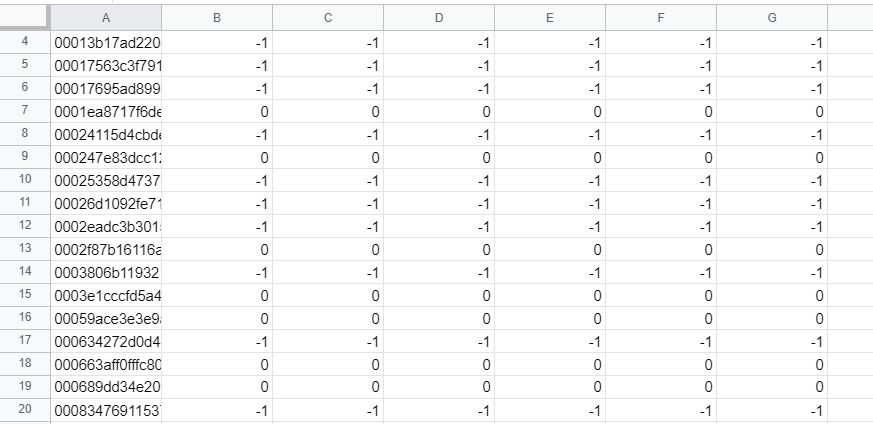
Training dataset:



Testing dataset:



Testing labels:



Our output is predicting which of the comments in the test set will be toxic and which of the categories it will fit into if it is. The accuracy of our algorithm is 99.21% which is displayed after running the program. The program also outputs a word cloud for each label and a bar chart that shows the number of comments in each category from the test set.

The main question that we wanted to answer with this project is what the key words in clean (not toxic), toxic, extremely toxic, obscene, threat, insult, and identity hate comments are. In order to answer this question, we used MatPlotLib to create word clouds for each label. The word clouds display the most frequent 500 words or phrases from the comments that fit that label.

The next question that we wanted to answer through this data mining task was how many comments fit into each label and which of the labels had the most comments. To display these results we created a bar chart using the MatPlotLib and Seaborn libraries. The bar chart displays the number of comments in each category and provides an easy way to compare across categories.

Our biggest challenge in this project was cleaning the data and formatting it into a way that could be parsed by our algorithm. To do this, we tokenize the comments and then remove all stopwords and punctuation using NLTK. Then based off of the edited comments that this process produces, our algorithm predicts which categories the comments fall into.

**Technical Approach**

To correctly read in our input data from Kaggle, we had to find a way to correctly and efficiently read in data from the csv format that was provided to us. We imported and used the Python library Pandas to accomplish this task. We used the built-in Pandas function read\_csv on the training and testing comments csv files to obtain an array that contained the comment itself and the toxicity information at each index.

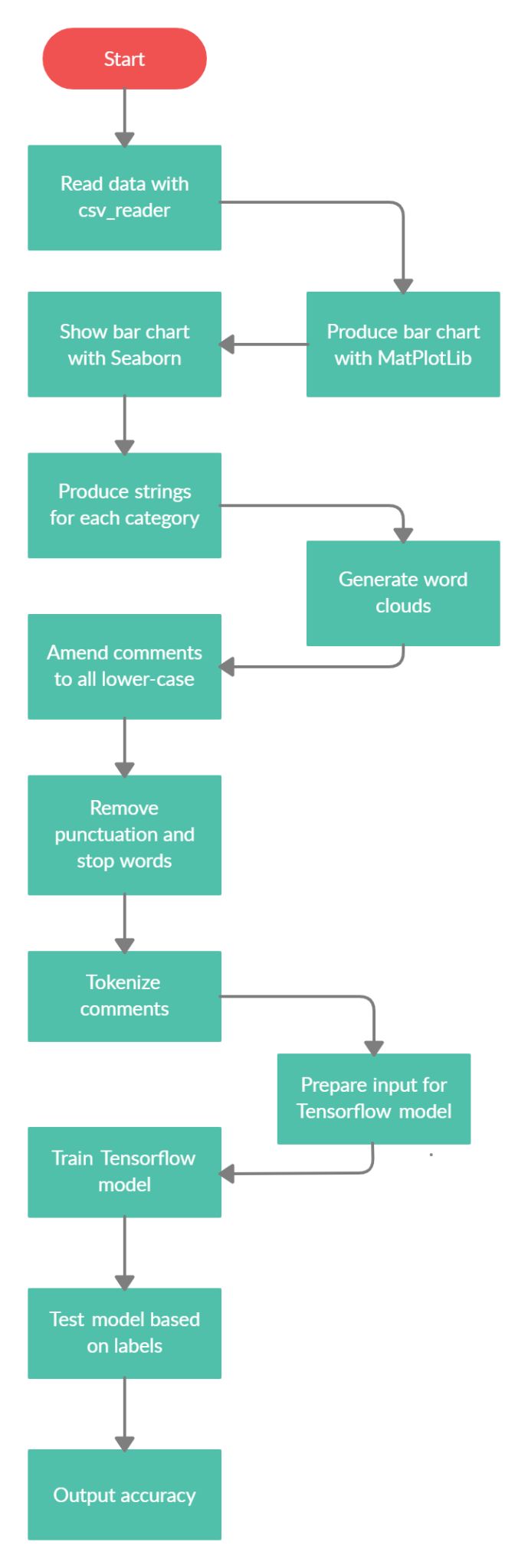
First, we wanted to visualize our data. To understand the proportion of comments that were non-toxic vs. toxic, we used a combination of the Python libraries Seaborn and MatPlotLib. We used Seaborn to produce a bar chart of our data by passing in the comments in our training comments and their respective values for the toxicity categories. We then used MatPlotLib to show this bar chart in an aesthetically pleasing way that contrasted the amount of comments that were non-toxic with the amount of toxic, severe toxic, oscene, threats, insults, and identity hate.

We also created word clouds for each of these categories. We imported the Python library wordcloud to accomplish this task. First, we formed a string of all words present in each non-toxic comment, as well as a string of all words in each of the other categories for each other toxicity category. We did this by using a for loop to join together each word in the comments in the training set whose value for its respective category was 1, not 0. Next, we specified the dimension of the word cloud via the library wordcloud and passed in each string containing the words in each toxicity category to generate the word clouds for each category.

Next, we had to correctly parse and clean our training and testing data in order to train our model. First, we made all characters lower-case, so that we did not have words starting with an upper-case character throwing off our results. Next, we used the Python library RE to remove all punctuation from our comments. After this, we used the Python library NLTK to turn each comment into a list of words in the comment by tokenizing the comment. Finally, we imported a list of stop words from NLTK and removed all stop words from our comments by creating a new list and using list comprehension to only add the words that weren’t stop words back into the comments of our training and testing set.

After this, we imported and used Tensorflow to create and train our machine learning model to classify the comments in our testing data. First, we used NLTK again to properly tokenize and pad our comment data to be used in the Tensorflow library functions. Next, we passed in our training and testing data into our model. We also specified the relevant parameters of our model, such as the loss, optimizer, and metrics, following the examples from Kaggle and the tutorial pages on the Tensorflow website. We use 2 epochs to train our Tensorflow model and obtain the accuracy of our model, which is output via the built-in Tensorflow function fit.

Block Diagram:



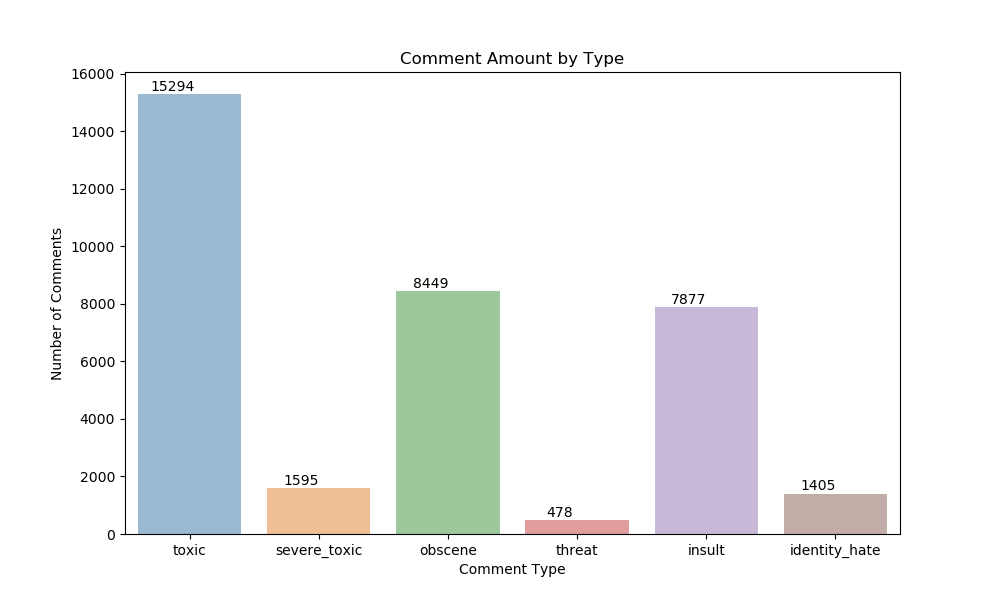
**Evaluation Methodology**

We obtained our data from Kaggle’s Toxic Comment Classification Challenge. Kaggle provided separate .csv files for the training and test datasets. These datasets are provided under the data section of the challenge on Kaggle: <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data>.

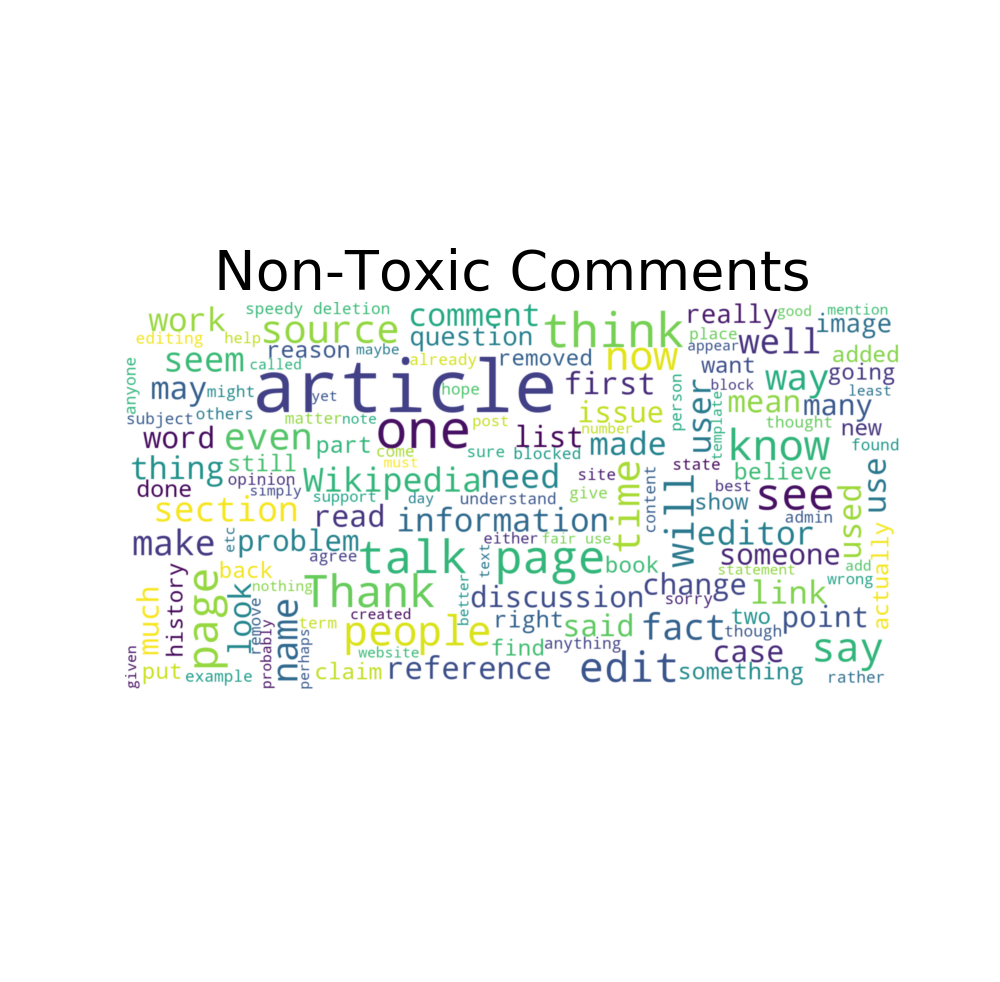
The only challenges we faced with the data sets was cleaning them so that they were prepared to be evaluated by our algorithm. We first had to tokenize the comments and then remove the stopwords and punctuation. We did this using NLTK from the TensorFlow library which was challenging as it was the first time we had used this method.

The metric we used to evaluate the results of our algorithm was accuracy. Since each comment was evaluated as a binary classification of fitting the category or not, we believed that accuracy was the best way to measure the results. The test set results could only be right or wrong for each comment, so more complicated metrics were not necessary to evaluate the results. In our Tensorflow model, accuracy is determined by whether the predicted value of our model run on our testing data for a comment is equivalent to the actual values in our testing labels corresponding to the same comment.

**Results and Discussion**

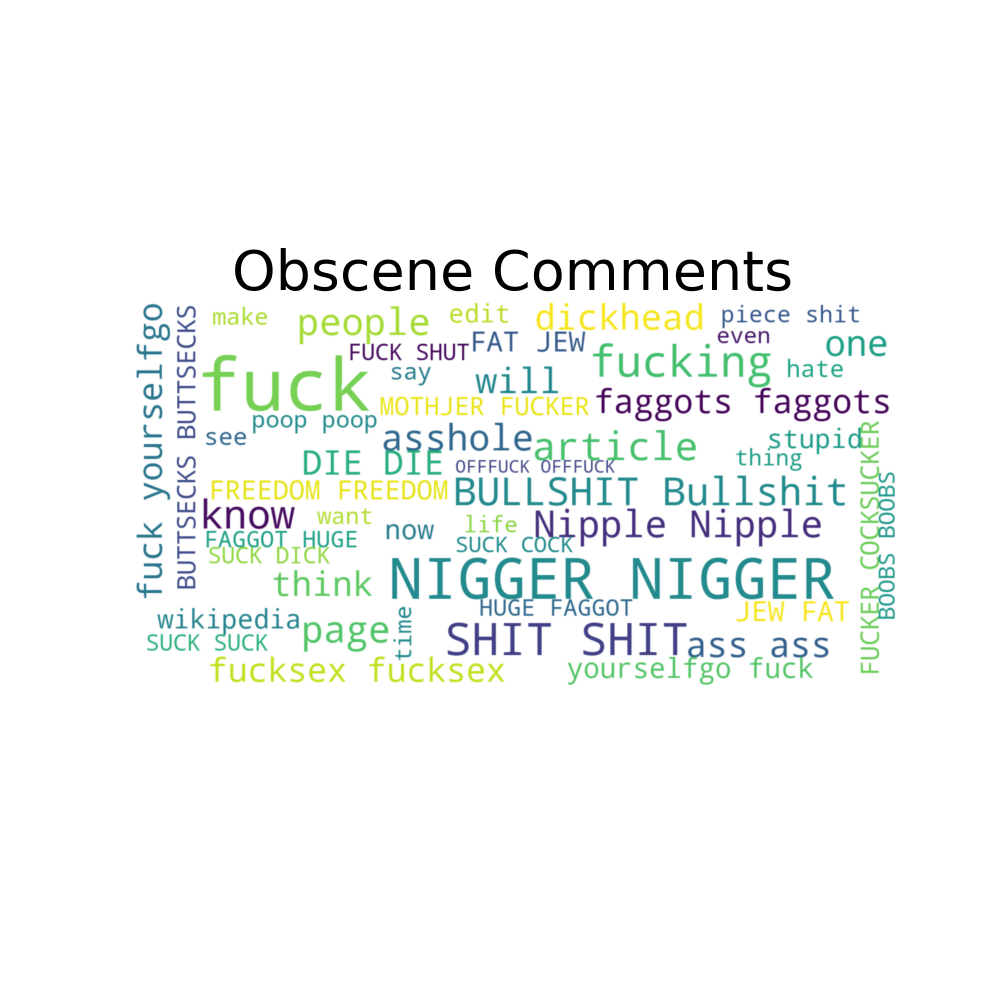
Here is a bar chart indicating the types of toxicity that were most rampant in Wikipedia comments. General toxicity, along with obscene and insulting comments, were by far the most dominant examples of toxicity in our data. We also see a smaller, but still significant amount of severely toxic comments and identity hate. Threats were the smallest category we analyzed.

Here we see that the most common words in our non-toxic pertained to regular discourse and discussion. We also see more Wikipedia specific words such as “article” and “Wikipedia”. Most striking in contrast to the other word clouds is the lack of offensive words and imagery.

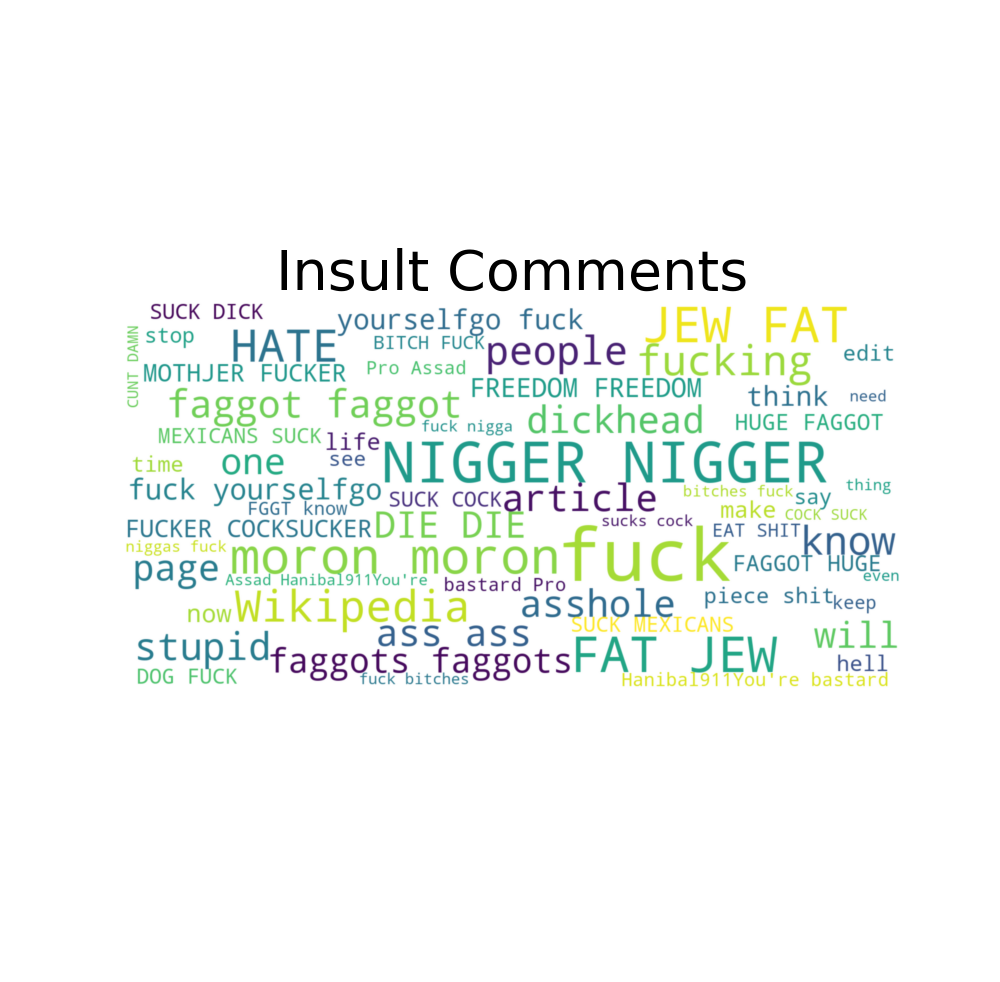


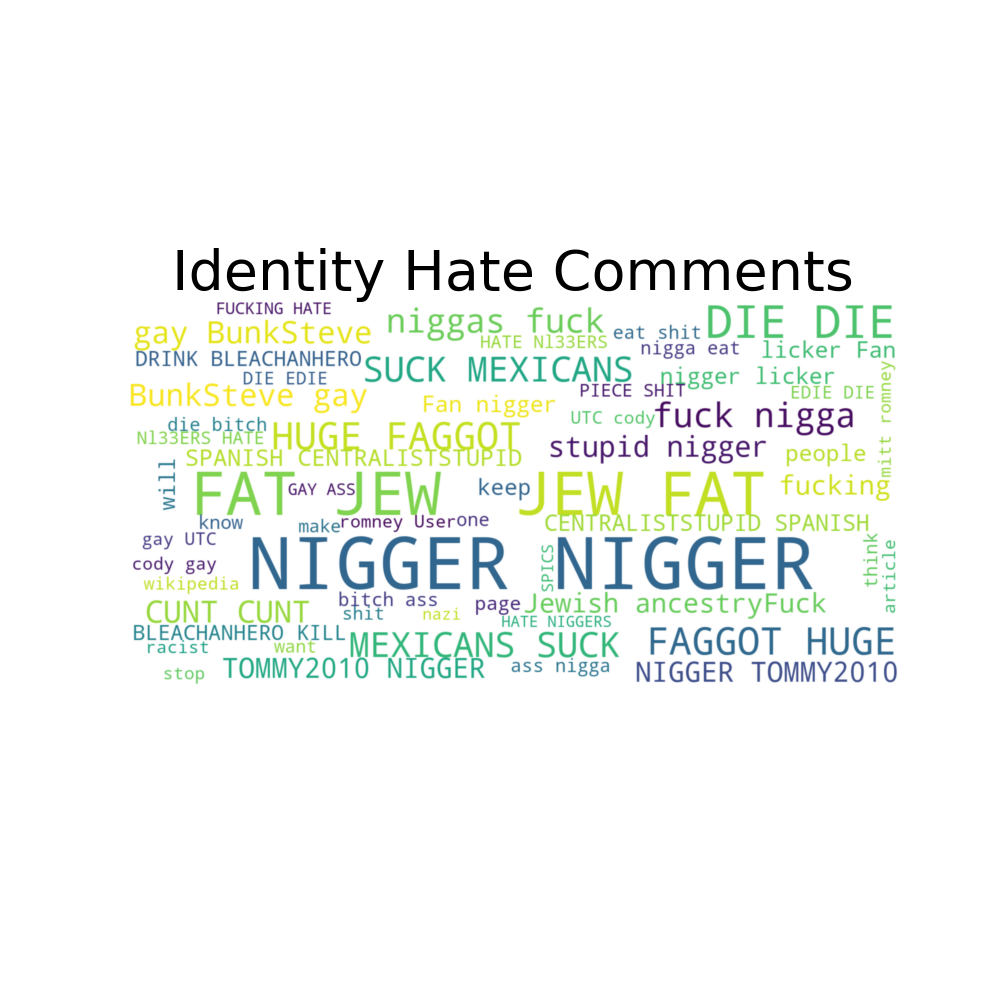
For the generally toxic comments, we encounter the use of generally vulgar language such as “shit” and “fuck”. However, we also see some targeted discriminatory words.

For the severely toxic comments, we see even more discriminatory words, as well an increase in the vulgarity of the language used.

For the obscene comments, we see a similar usage of words, with a small increase in the amount of references to disturbing imagery.

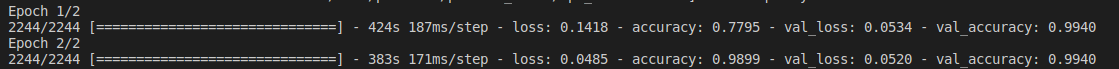
For the insulting comments, we see a continuation of discriminatory words, which is consistent with the category. However, we also see an increase in generally unpleasant adjectives, such as “fat” and “moron”. This is also consistent with the category of insults.



For the words used in identity hate, we see an even greater usage of discriminatory slurs. This is consistent with the category of identity hate.

For comments that are indicated to be threats, we see an increase in the use of verbs. This is evidenced through the appearances of “die” and “kill” in our word cloud. This is consistent with the category of threat that we are measuring.

Accuracy of our model:



We can see that after training our data a second time, we were able to much more accurately label our comments. After the first epoch, we obtained an accuracy of 0.7795. However, after the second epoch, our accuracy jumped to 0.9899, indicating that almost 99% of comments were able to be correctly classified.

**Lessons Learned**

By doing this project we learned how to work with different Python libraries including TensorFlow, NLTK, MatPlotLib, and Seaborn. This was good experience for us as we will likely have to work with these libraries for future projects both in school and on our own. We also learned a lot about string manipulation. We had to learn how to pull the individual words out of the comments and convert them to their own strings. Additionally, we were required to manipulate different data structures in this project which is important knowledge to have as it is a frequent task in coding.

Looking back, we could have done more with analyzing the accuracy of how each label was classified. For example, we might have done very well classifying toxic versus clean comments, but we don’t know the accuracy of how well we classified obscene comments or insult comments. This would have been interesting to analyze, but for the purpose of social media platforms being able to remove negative comments, it is not crucial to know how they are negative.

**Acknowledgements**

In order to complete this project, we used several sources on the internet. Our main source was Kaggle as this is where we found our data and two example projects that provided some direction for the required steps to clean the data. We also referred to the TensorFlow tutorials often to see how to correctly implement its library functions. We occasionally referenced StackOverflow to answer various coding questions about topics such as syntax and the necessary arguments for functions.