Final Project Report for CS 175, Fall 2022

Project Name: Team AMA - Classifying Toxicity

CS 175 Fall 2022

List of Team Members:

Ashwin Sampath, asampat2, asampat2@uci.edu
Andy Nguyen, andyn18, andyn18@uci.edu
Marco Lam, mslam2, mslam2@uci.edu

1. Project Summary

Our project is focused on the multi-label classification of toxic comments into different subtypes of toxicity. This project uses logistic regression, recurrent neural networks, and a chain classifier to label the Kaggle dataset, which consists of a large number of Wikipedia comments that are labeled different degrees of toxicity.

2. Data Sets

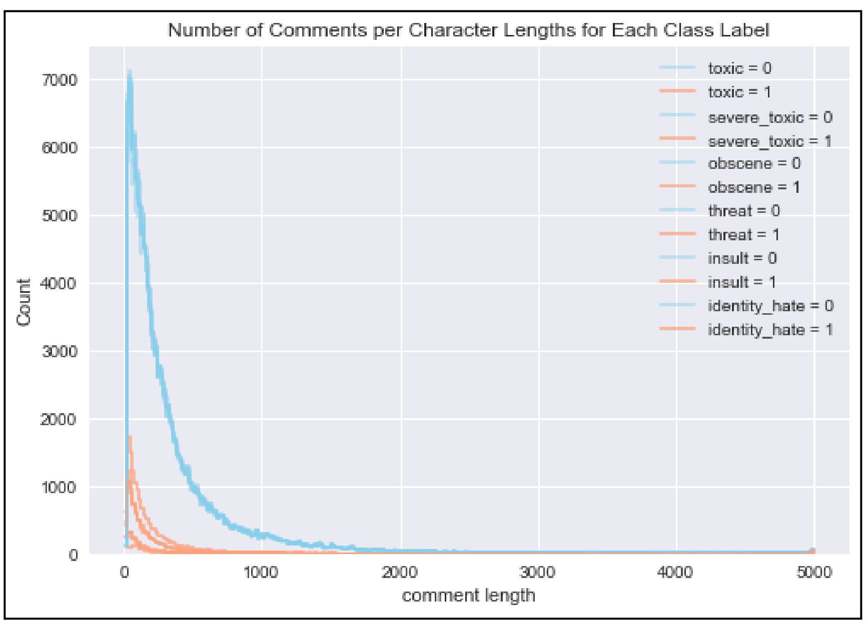
We are using the dataset from a Kaggle competition[4], which is structured as a csv with the following columns: id, comment_text, toxic, severe_toxic, obscene, threat, insult, and identity_hate. The list of comment_text is determined by human readers if each comment falls into any (or some) of the latter 6 labels — 0 for False and 1 for True. Example data include:

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0007e25b2121310b	Bye! \n\nDon't look, come or think of comming ...	1	0	0	0	0	0
006b94add72ed61c	I think that your a Fa**et get a oife and burn...	1	0	1	1	1	1

Note how the second comment is already censored, which can cause noise for our models. Besides interesting aspects of comments that require text pre-processing mentioned in the Technical Approaches section, the number of comments per class also reveals interesting relationships within the dataset itself. For example, there are zero comments that are labeled severe_toxic but not toxic — a Naive Bayes approach can ignore such dependencies. In the chart below, the yellow diagonal shows the total number marked with the respective label, while the non-diagonals shows the intersection of comments labeled one class but not another.

Number of Comments labeled toxic (1) or not (0)						
0 / 1	**toxic**	**severe_toxic**	**obscene**	**threat**	**insult**	**identity_hate**
toxic	15294	0	523	29	533	103
severe_toxic	13699	1595	6932	366	6506	1092
obscene	7368	78	8449	177	1722	373
threat	14845	1483	8148	478	7570	1307
insult	7950	224	2294	171	7877	245
identity_hate	13992	1282	7417	380	6717	1405
Note: yellow cell show the total number of comments with 1 for the column label. i.e. top-left cell denotes 15294 comments labeled with ‘toxic’ = 1. Other statistics: 143346 non-toxic comments (with 0 for all labels), out of 159571 elements total.						

Although the chart may not clearly demonstrate this, the unbalanced ratio of the number of toxic and non-toxic comments is a potential challenge: 143k of the total 159k comments are non-toxic — only roughly 10% of the dataset are toxic. Without balancing the dataset, many training iterations will be filled by non-toxic comments and make little progress at identifying toxic features. Below graph shows another interesting observation: most toxic comments are well under one thousand characters long. This heuristic is unverified and is a potential path for future experiments.



3. Technical Approach

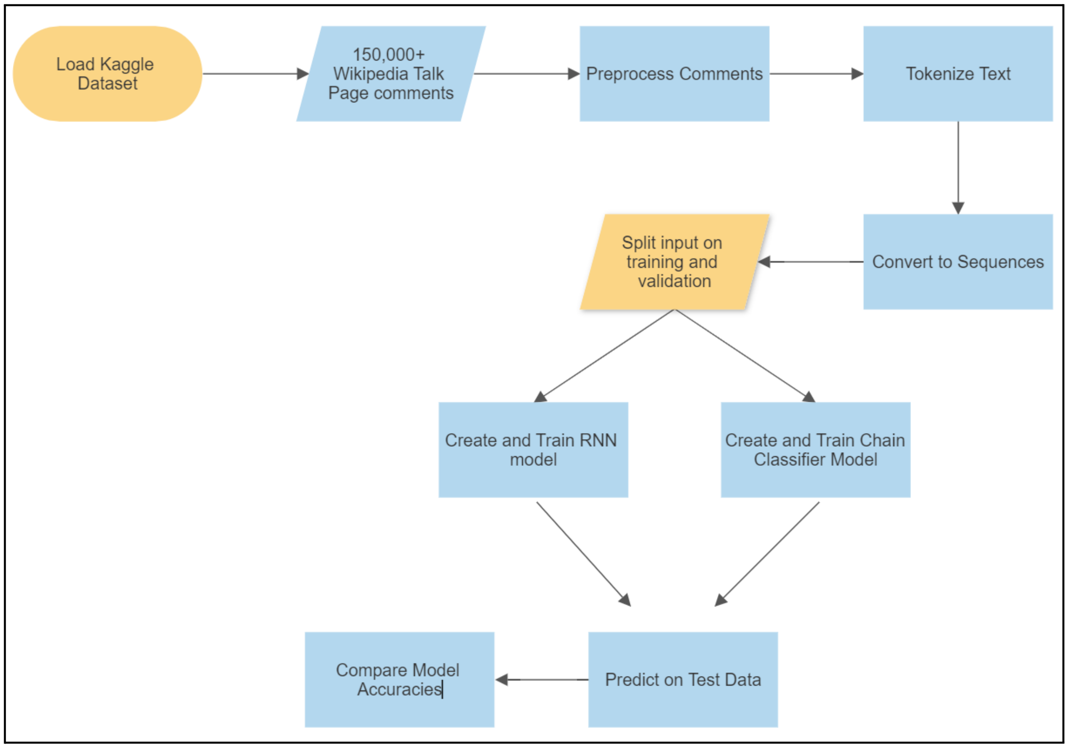
We are implementing Logistic Regression as a baseline model, a Recurrent Neural Network for a more advanced model, and possibly a Chain Classifier as a third model. The baseline model focuses on independent binary classification for each of the 6 output labels. In other words, it is a sort of Naive Bayes approach and does not take into consideration that some labels might be related to each other. Meanwhile, the RNN includes Long-Short Term Memory layers to improve performance, and we plan to compare each models' accuracy scores afterwards.

As for the training process, we split the dataset into training and validation sets. Before training, we preprocess each comment by removing punctuation and special characters, then removing stop words to form a string of words separated by spaces. After the model is trained, we will use it to predict labels for the data in the validation set. To measure accuracy, we will be calculating the area under the ROC curve from the prediction scores.

With such a large dataset, the ratio between 0, nontoxic, and 1, toxic, classifications were unproportional with an abundance of 0s. This led to our classifiers returning higher accuracy scores perceivable after experimentation. In order to prevent this, we balanced the data by selecting a random subset of nontoxic comments equal in size to the total set of toxic comments. That leaves a dataset with the same amount of 0s to 1s, which we called balanced. Classifiers were then trained using both the original dataset and balanced dataset, providing information about the inaccuracy of our classifiers.

For our neural network, we referenced both chapter 7 in *Speech and Language Processing* by Jurafsky and Martin[1] — which discusses the various units of neural networks — and the "Multi-Label Classification with Deep Learning" article[2] to help us implement the model. After reading, we realized that we need a softmax function in our output layer as it is a multi-label classification problem. The text also discussed various loss functions, including cross-entropy loss that we included in our model.

Finally, for the chain classifier, we decided on an underlying Logistic Regression model from Scikit learn that would be chained together. This classifier works well for multilabel classification problems by making predictions with added assistance from previous predictions made earlier in the chain.



4. Experiments and Evaluation

Overall, the naive Logistic Classifier performed much better than expected, achieving close to par on test accuracy compared to the Chain Classifier and much better than RNN. However, it is more prone to false positives than the Chain Classifier, whereas the RNN was slightly better at avoiding false negatives.

- Balancing the dataset by selecting a random subset of nontoxic comments did not improve performance as expected, and instead became less stable, as recorded in figures A1 and A3 in the appendix.
    - Training accuracy lowered by an average of 12%.
    - Training ROC AUC varied from -20.5% (insult) to +11.6% (toxic).
    - The reduced sample size seemed to have limited the model's ability to define the boundary between toxic and non-toxic words. The result is worse performance in the classification tasks.
- To verify the above observations, we further trimmed the dataset to a tenth of the toxic comments — creating a "mini" dataset with the original nontoxic-toxic ratio and effectively scaling down the dataset's size.
    - Interestingly, the test results in A3 demonstrate that the ratio, rather than the smaller sample size, was the larger factor for prediction performance; the models trained on the "mini" dataset performed comparable to models trained on the full dataset.

A future experiment we had in mind came up after presentations, since we came to the understanding that classifying whether comments were toxic or not did not capture the entire picture. Language is just a complex concept that is constantly evolving, and in order to truly build a toxic classifier we have to attempt to cover some of the nuances of language. Sarcasm is part of the language, and trying to differentiate sarcasm from truly toxic comments would allow for more realistic classification. Unfortunately, we were not able to begin experimenting with sarcasm. This future experiment would consist of us classifying another dataset of toxic sarcasm and use it to predict sarcasm in our current toxic comment database.

5. Lesson Learning & Insights

- Unprocessed text needs a lot of work to transform into useful, structured input.
    - Information beyond the text — context, urls, references — are uncaptured by the comment itself.
    - Improper syntax, such as misspelled or misused words, remain a largely unsolved problem for our models.
    - Grammar and punctuation, on the other hand, can be compensated by tokenization and removing non-alphanumeric characters, although special cases may cause incorrect interpretations, especially in terms of spacing.
- As noted in Experiments and Evaluation, balancing the dataset is more complex than simply changing the ratio to 50:50. Doing so in theory helps training efficiency for RNN, but performance seems to suffer as the ratio grows farther than the true boundary.
- A naive Logistic Classifier already achieves reasonable results for this task, better than our trained RNN. As expected, the models did better on classifying the more extreme labels (severe_toxic, threat, identity_hate).
- Because of the complexity of language, many uses of profanity or words that could be considered toxic together could actually be sarcasm, thus proving a problem in the accuracy of our models.
- When comparing the scores across the various classifiers, there’s a common trend that you can see. Some of the more extreme cases of toxicity were easier to predict than the lower degrees of toxicity. This ties back to the complexity of language, and how sarcasm lies within the realm of softer toxic comments, which things such as identifying hate is more often than not is easily identifiable. Much like a real person can distinguish identity hate from toxic, our classifier being able to do so reinforces that our project accomplished parts of our goals.

APPENDICES

A. Additional Graphs and Tables

A1 - Training Scores for Logistic Classifiers

Training Scores for Logistic Classifiers						
Score | Model	**toxic**	**severe_toxic**	**obscene**	**threat**	**insult**	**identity_hate**
Accuracy | Basic Logistic	0.919	0.990	0.959	0.997	0.954	0.991
Accuracy | Balanced Logistic	0.766	0.949	0.850	0.986	0.813	0.756
Accuracy | Mini Logistic	0.924	0.990	0.960	0.997	0.957	0.992
ROC AUC | Basic Logistic	0.761	0.764	0.765	0.663	0.924	0.659
ROC AUC | Balanced Logistic	0.849	0.721	0.807	0.615	0.734	0.637
ROC AUC | Mini Logistic	0.722	0.797	0.738	0.629	0.700	0.603
Log Loss | Basic Logistic	-0.355	-0.208	-0.285	-0.190	-0.283	-0.202
Log Loss | Balanced Logistic	-0.603	-0.457	-0.558	-0.47	-0.564	-0.446
Log Loss | Mini Logistic	-0.366	-0.238	-0.298	-0.211	-0.297	-0.220

A2.1 - Training Scores for Chain Classifier

Overall Training Scores for Chain Classifier			
model	**accuracy**	**ROC AUC**	**Log Loss**
chain_full	0.903	0.714	-0.366
chain_balanced	0.548	0.699	-1.784

A2.2 - Training Precisions for Chain Classifier

Precision Scores per Label for Chain Classifier						
Dataset	**toxic**	**severe_toxic**	**obscene**	**threat**	**insult**	**identity_hate**
Full	0.98	0.71	0.99	0.00	0.76	0.00
Balanced	0.86	0.62	0.96	0.00	0.78	0.00
Mini	1.00	1.00	1.00	0.00	0.87	0.00

A2.3 - Training Recall for Chain Classifier

Recall Scores per Label for Chain Classifier						
Dataset	**toxic**	**severe_toxic**	**obscene**	**threat**	**insult**	**identity_hate**

Full	0.15	0.03	0.23	0.00	0.20	0.00
Balanced	0.47	0.02	0.34	0.00	0.27	0.00
Mini	0.09	0.07	0.15	0.00	0.14	0.00

A3 - Test Prediction Scores

Test Scores for "**Toxic**" Label			
model	**accuracy**	**false_pos**	**false_neg**
logistic_full	0.91691	0.00364	0.07945
log_balanced	0.84065	0.10751	0.05185
logistic_mini	0.9142	0.0015	0.08429
rnn_full	0.79682	0.11526	0.08792
rnn_balanced	0.68663	0.25151	0.06187
rnn_mini	0.83305	0.08237	0.08458
chain_full	0.91655	0.00253	0.08092
chain_b	0.87071	0.07146	0.05783
chain_mini	0.91416	0.00083	0.08501

Test Scores for "**Severe Toxic**" Label			
model	**accuracy**	**false_pos**	**false_neg**
logistic_full	0.99408	0.00067	0.00525
log_balanced	0.99301	0.00184	0.00514
logistic_mini	0.99423	0.00025	0.00552
rnn_full	0.99422	0.00006	0.00572
rnn_balanced	0.99369	0.0008	0.00552
rnn_mini	0.99398	0.00031	0.00571
chain_full	0.99428	0.0003	0.00542
chain_b	0.99426	0.00027	0.00547
chain_mini	0.99422	0.00023	0.00555

Test Scores for "**Obscene**" Label			
model	**accuracy**	**false_pos**	**false_neg**
logistic_full	0.95302	0.00166	0.04533
log_balanced	0.95311	0.00666	0.04023
logistic_mini	0.95092	0.00091	0.04817
rnn_full	0.88352	0.06065	0.05583
rnn_balanced	0.86814	0.08887	0.04298
rnn_mini	0.90511	0.0409	0.05399
chain_full	0.95298	0.00169	0.04533
chain_b	0.95317	0.0066	0.04023
chain_mini	0.95092	0.00091	0.04817

Test Scores for "**Threat**" Label			
model	**accuracy**	**false_pos**	**false_neg**
logistic_full	0.9967	0	0.0033
log_balanced	0.99661	0.00009	0.0033
logistic_mini	0.9967	0	0.0033
rnn_full	0.99669	0.00002	0.0033
rnn_balanced	0.9967	0	0.0033
rnn_mini	0.9967	0	0.0033
chain_full	0.9967	0	0.0033
chain_b	0.9967	0	0.0033
chain_mini	0.9967	0	0.0033

Test Scores for "**Insult**" Label			
model	**accuracy**	**false_pos**	**false_neg**
logistic_full	0.95076	0.00219	0.04705
log_balanced	0.94939	0.00656	0.04405
logistic_mini	0.9508	0.00239	0.04681

Test Scores for "**Identity_Hate**" Label			
model	**accuracy**	**false_pos**	**false_neg**
logistic_full	0.98887	0	0.01113
log_balanced	0.98875	0.00013	0.01113
logistic_mini	0.98887	0	0.01113

rnn_full	0.92233	0.02496	0.05271
rnn_balanced	0.89926	0.05549	0.04525
rnn_mini	0.92377	0.02421	0.05202
chain_full	0.95147	0.00502	0.04351
chain_b	0.94995	0.00974	0.04031
chain_mini	0.95114	0.00295	0.04591

rnn_full	0.98834	0.00055	0.01111
rnn_balanced	0.98065	0.00836	0.01099
rnn_mini	0.98823	0.00064	0.01113
chain_full	0.98887	0	0.01113
chain_b	0.98875	0.00013	0.01113
chain_mini	0.98887	0	0.01113

B. References

[1]Jurafsky, Dan & Martin, James H. *Speech and Language Processing* (3rd ed. draft). https://web.stanford.edu/~jurafsky/slp3/.

[2]Brownlee, Jason. "Multi-Label Classification with Deep Learning". https://machinelearningmastery.com/multi-label-classification-with-deep-learning.

[3]Schroeder, Adam. "CountVectorizer, TfidfVectorizer, Predict Comments". https://www.kaggle.com/code/adamschroeder/countvectorizer-tfidfvectorizer-predict-comments#TfidfVectorizer----Brief-Tutorial.

[4]Jigsaw/Conversation AI. "Toxic Comment Classification Challenge". https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data