Cleaning up Comments: Using Deep Learning to Detect Digital Harassment

Luke Carlson
University of Pennsylvania
carjack@cis.upenn.edu

Abstract

Online harassment though hateful or insulting comments has become a major problem in online communities. Solutions like community self-policing via flagging content or empowering moderators to take down problematic text can work but require extra human effort and may not catch hateful content quickly (Koebler, 2017). The ideal solution would be an automated system that can detect these comments as they are posted. In this paper I present a number of machine learning models that aim to accurately identify hateful, offensive, or insulting content on Wikipedia Discussion Pages. As a companion to this paper, all models, evaluations, and error analyses are available in interactive Jupyter notebooks at http://github.com/jLukeC/cleaning-up-comments.

1 Introduction

Abusive behavior online, including cyberbullying, has become a major cultural issue in 2018. Over 41% of U.S. adults have identified themselves of victim of online harassment, with 18% reporting that they have been targeting with physical or sexual harassment online (Duggan, 2017). Web platforms have generally taken a hands off approach to this issue, pushing responsibility onto end users to self police communities (Butcher, 2017). Although this strategy has worked for a number of years, public perception is shifting and now 80% of adults believe that platforms need to step in to address harassment more often (Duggan, 2017). Even Wikipedia, deals with harassment and digital abuse issues among its users (Cohen, 2018).

Jenna Wortham in a New York Times Magazine piece titled *Why Cant Silicon Valley Fix Online Harassment?* (2018) paints a bleak picture of the future of cyberbulling. In order to handle the issue, people need to first agree on what behavior is consistent with digital harassment and then move to build systems to detect harassment. The author suggests "start-ups put lucrative bounties on devising anti-abuse tools".

In the last semester of my masters degree in computer science at the University of Pennsylvania, I have engaged in an independent study with a focus in natural language processing and deep learning. I chose investigate automatic identification of online harassment as I believe that cyberbulling is a real problem and my aim is to build accurate and deployable models while exploring various approaches to an imbalanced multi-label text classification setting.

2 Toxic Comments Challenge

Conversation AI, in coordination with Wikipedia and Kaggle, has attempted to tackle these exact problems with the creation of the Toxic Comments Challenge. First, in order to address the imprecise definition of digital harassment, negative behavior is identified through a combination of 6 different labels creating a more nuanced taxonomy then just a binary harassment/not harassment approach. Second, the startup offered $35,000 in prizes to incentivize developers to create high quality solutions. The competition ran from December 2017 to March 2018 and attracted 4551 teams that submitted models.

2.1 Task Definition

The Toxic Comments Challenge is a multi-label text classification problem. The six labels in this dataset are:

`toxic, severe toxic, obscene, threat, insult, identity hate`

A comment can have any subset of these labels attached to it. For instance, a comment may be considered an `[insult]` if it involves cursing or it may be an `[insult,` `identity hate]` if the cursing includes racism. In one example in the Appendix, a commenter hopes that another user is beheaded, defecated upon, and stabbed in the heart. That comment is labeled as `[toxic, severe toxic,obscene, threat, insult]`.

2.2 Examples

Table A.3 in Appendix A provides a number of examples of abusive comments from the dataset. An important note is that most comments in the dataset are not in fact toxic in any way, these are just representatives of positively identified harassment.

Note: this dataset contains highly offensive material

2.3 Dataset creation

In Wulczyn et al. (2017), the authors lay out their process for assembling the labeled dataset. First they extracted talk page comments from a public dump of Wikipedia pages dated from 2004 to 2015. Then they hired 5000+ human annotators on CrowdFlower to answer questions about the comments including whether to label them as toxic, insulting, etc. Each comment was labeled by 10 or more human annotators and the final label set was created from these results.

2.4 Dataset exploration

Figure 1 shows the label distribution throughout the dataset. There is a clear imbalance between all the labels, with `toxic` appearing a magnitude more often than than `identity hate, threat, severe toxic.`

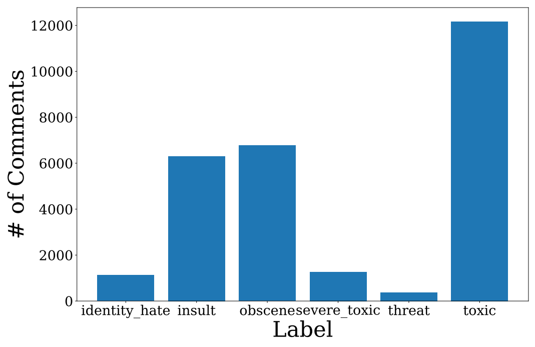


Figure 1: Comment Length Distribution

Figure 2 shows a histogram of comment lengths (in characters) and Table 1 breaks down average comment length for each label. Interestingly, severely toxic comments are ~50 characters longer than the average comment whereas all other harassment types (`insulting, toxic, identity hate, and obscene`) are generally 100 characters shorter than average. That could indicate that users put less thought into offensive comments than normal comments but put in more effort when writing highly offensive comments.

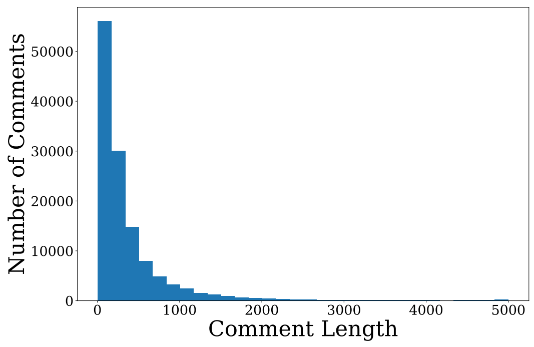


Figure 2: Histogram of Comment Lengths

Label	Average Length
toxic	292.42
severe toxic	443.02
obscene	283.34
threat	303.88
insult	274.58
identity hate	304.36
all comments	394.16

Table 1: Average Comment Length for Each Label

2.5 Definitions

I will be using the following notation in this paper. See Herrera et al. (2016) for more information on multi label classification problems in general.

Let $\mathcal{X}$ be our input space. In this case, each element $\mathbf{x} \in \mathcal{X}$ is a vector of n features representing a text comment: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$

Let $\mathcal{L}$ be the set of k possible labels: $\{l_1, \ldots, l_k\}$. In this case this consists of the $k = 6$ labels mentioned above.

Let $\mathcal{Y}$ be the output space where $\mathbf{y} \in Y$ is a vector $\mathbf{y} = (y_1, y_2, \ldots, y_k)$ where each element y_i is the probability that a given instance $\mathbf{x}$ has a specific label l_i. In other words, $y_i = Pr(l_i|\mathbf{x})$.

Let $\mathcal{F}$ be a multi-label classifier, defined as the mapping $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$. For a given text instance $\mathbf{x}$, the output of classifier $\mathcal{F}$ is denoted as $\mathcal{F}(\mathbf{x}) = \hat{\mathbf{y}}$.

3 Models

In this paper I will be discussing five different modeling approaches consisting of a simple baseline, two classical machine learning approaches, and two deep learning models.

3.1 Multi label Naive Bayes

A Naive Bayes model is a classic starting point for text classification. Naive Bayes classifiers treat the input text as a bag-of-words and assumes all features are conditionally independent. The multi-label variant I built for this task involves creating a binary Naive Bayes model f_i for each target label l_i. Then each input comment $\mathbf{x}$ is evaluated independently on all $|\mathcal{L}|$ models. The output for model f_i is then y_i:

$$y_i : Pr(l_i|\mathbf{x}) = Pr(l_i) \prod_{x \in \mathbf{x}} Pr(x|l_i)$$

The end-to-end model $\mathcal{F}$ then outputs the label probabilities $\hat{\mathbf{y}} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))$

For my implementation, I used the the `scikit learn` python library to create a pipeline that starts with a `Count Vectorizer` then a `Multinomial Naive Bayes` wrapped in a `One vs Rest Classifier`.

3.2 Naive Bayes + Logistic Regression

Wang and Manning (2012) lays out an informative process for building solid machine learning models for topic classification: their innovation is to train a linear classifier using Naive Bayes log counts as features. For this model I worked off of Jeremy Howard's take on the NB+LR model.

3.3 Deep Learning

3.3.1 Neuron and Neural Networks

In essence, a neuron is a function that takes in a matrix of inputs x and a matrix of weights W that indicates the relative importance of each input field then outputs a value between [0,1].

$$h(x; W) = s(W \cdot x + b)$$

Where s is an activation function that handles squashing any input to a value in [0,1] and has other valuable properties for model training. A neural network is then a number of connected layers of neurons and the exact layout defines the model architecture.

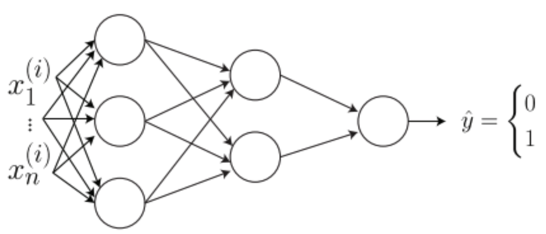


Figure 3: A neural network architecture

A neural network model consists of $H_{in}, H_1, \dots, H_{out}$ layers of neurons. The first layer H_0 takes in the input then the data is reshaped in any way through the hidden layers $H_1 \dots$ until reaching the output layer H_{out} whose shape is defined by the task at hand. In our case, we want to output predictions for each label so there would be 6 outputs. So the prediction for a given instance could be described as:

$$\hat{\mathbf{y}} = H_{out}(H_2(H_1(H_{in}(\mathbf{x}); W_1); W_2); W_{out})$$

3.3.2 Recurrent Neural Networks

In our task, we can think of x as the representation of one of the words in our sentence $\mathbf{x}$. You can imagine that in natural language processing tasks the order of words in a sentence is crucial otherwise a model might not understand the true meaning of a sentence like `You are not very smart`. The problem with the simple neuron is that it has no temporal understanding of the inputs: $h(x_1)$ and $h(x_2)$ are completely independent. Thus a simple neural network would only be able to approach this problem as a bag-of-words-model.

The solution to this problem is a Recurrent Neural Network. RNNs maintain an internal state which enables them to process sequential data. Each neuron in the network predicates its current output on knowledge it has on previous outputs. That means adding a new weight matrix U and including prior input x_{i-1} in the computation.

$$h(x_i; W, U) = s(W \cdot x_i + U \cdot x_{i-1} + b)$$

3.3.3 Long Term Dependencies and the Vanishing Gradient

A recurrent neural network aims to encode a variable length sequence of word tokens into a fixed length context vector used for final evaluation. When the input is very long though, it becomes difficult for the resultant fixed length vector to be a good representation of the input.

For instance, we know that the average text sequence length in this task is 394 words (Table 1). An RNN may be able to create a good representation for most sentences of that length but when it comes to longer sentences, say 600+ words, the resultant context vector may not have kept track of the most important parts of the sentence and thus the label prediction is poor.

This is known as the vanishing gradients problem: as the context matrix is multiplied through each neuron, the derivatives of these computations become smaller and smaller. Neural networks are trained through the back propagation algorithm where the gradient of the loss function is computed with respect to each neuron and its weights are updated accordingly. When the partial derivative of the context with respect to the cost function becomes minuscule it can lead to entire layers not being updated.

3.3.4 Long Term Short Term Networks

To handle this, Long Term Short Term neurons were introduced in Hochreiter and Schmidhuber (1997). These networks maintain a context parallel to neuron computation: each neuron decides what context to add or remove, but unlike RNNs this second context is not threaded through each neuron. Thus the final fixed length vector used for prediction becomes a better representation of the input, leading to better predictions.

For this paper, I used the `keras` library to create a neural network starting with an `Embedding Layer (32,32)`, `LSTM (32,32)` and finally a `Dense (32,6)` layer with a sigmoid activation on the six outputs. For training, I used adam optimizer with binary cross entropy loss and trained for 10 epochs.

3.3.5 Bidirectional Models: Dependencies in Both Directions

To go one step further with the long term dependency problem, at times the output of a sequence model should depend on both prior output *and* future output. In this case, that could mean that when a model is consuming a comment, whether it should be labeled as insulting could depend on context from the start of the sentence as well as context from the end of the sentence.

In Bidirectional neural networks, two models are stacked on top of each other. One reads the input left to right and the other reads the input right to left. Each output is then computed using both models, thus maintaining context from both directions (Schuster and Paliwal, 1997).

Ding et al. (2018) showed that Bidirectional LSTMs can be highly effective when applied to sentence classification. To create my BiLSTM, I wrapped my earlier model in a `Bidirectional` layer and trained it for 8 epochs.

3.3.6 Weakness of LSTM

The downside of LSTMs is the number of additional weight parameters added to the system. A recurrent neuron consists of the following learned parameters that sum to $nm + n^2 + b$ learned parameters for each neuron:

$$W : n \times m \text{ shaped matrix}$$
$$U : n \times n \text{ shaped matrix}$$
$$b : n \text{ length vector}$$

A LSTM on the other hand, consists of four different gates, each with their own W and U matrices. In total that leads to $4(nm + n^2)$ learned parameters. A BiLSTM is even larger as it consists of two LSTMs so minimum $8(nm + n^2)$ parameters.

4 Evaluation

For the Toxic Comments Kaggle competition, models were evaluated on a public test set (25%) and a private test set (75%). There were over 4,500 team submissions for this challenge. My best model put me in the top 24% percent of submissions — where a $+.001\%$ increase in my model's accuracy would have put me in the top 2.5% of all competitors.

4.1 Methodology

The Kaggle competition used Mean Columnwise Area Under Receiver Operating Characteristic Curve as its scoring metric — essentially this is the average ROC AUC score across all labels. One downside of using a pure average is that there is a clear label imbalance in this dataset, see Figure 1. In this paper, I used a weighted mean ROC AUC as the evaluation metric on my test set in order to account for the imbalanced label distribution.

Receiver Operating Characteristic Area Under the Curve Scores							
Model	toxic	severe toxic	obscene	threat	insult	identity hate	weighted avg
Random Baseline	0.496	0.522	0.497	0.481	0.507	0.451	0.498
Naive Bayes	0.806	0.728	0.802	0.528	0.779	0.571	0.782
Naive Bayes with LR	0.947	0.936	0.954	0.933	0.944	0.918	0.946
Basic LSTM	0.944	**0.982**	0.970	**0.949**	0.967	0.946	0.957
Bidirectional LSTM	**0.962**	**0.986**	**0.978**	**0.949**	**0.975**	**0.954**	**0.969**

Table 2: Every model's ROC AUC scores for each label

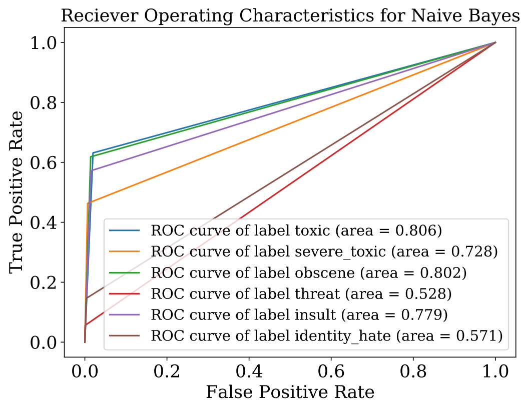


Figure 4: ROC graph for Naive Bayes Model

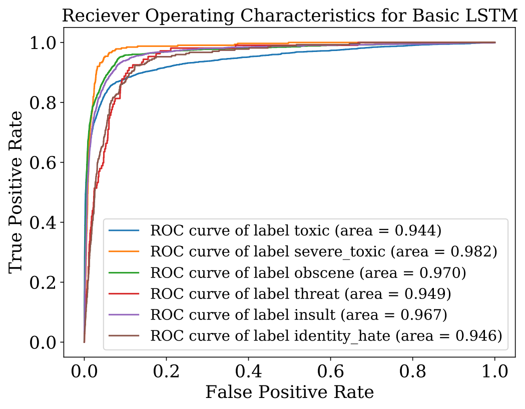


Figure 5: ROC graph for LSTM Model

4.1.1 Dataset

The dataset for this project is a collection of 160,000 labeled comments from Wikipedias discussion pages. For evaluation, I split the data 80/20 between a training set and a test set.

4.1.2 Receiver operating characteristic area under the curve

For this task, the evaluation metric the I used to judge all the models is based on the receiver operating characteristic curve. The ROC curve graphs a models true positive rate against its false positive rate at prediction threshold ranging from [0,1]. The area under an ROC curve indicates the probability that a model can discriminate between a true positive and a false positive.

As this is a multi-label problem with true and false positives for each label, the final score of a model is its weighted average ROC AUC. I used a weighted average as the labels do not have an equal distribution in the dataset. For my implementation, that meant using the `roc_auc_score` in `scikitlearn` with the `average = weighted` argument.

4.2 Results

Table 2 breaks down every model's ROC AUC score for each label along with a final weighted score. Appendix A.1 contains ROC graphs for each model as well. For illustrative purposes, Figure 4 shows the receiver operating characteristic curve for the Naive Bayes model and Figure 5 shows the same for the LSTM model.

We can see that by the 0.2 threshold the LSTM model is highly effective at distinguishing between a true positive and a false positive. In the next section, I'll analyze the few mistakes the model continues to make at this threshold.

4.3 False Positives at 0.2 Threshold for BiLSTM

At the 0.2 threshold, an example false positive for the most robust model, the bidirectional LSTM, is:

```
is anyone there?  HELLO?
      MOM??????
```

The model ranks it as `toxic` whereas in truth it has no labels. In this case, the biLSTM was clearly influenced by the all capital letters and believes that to be a sign of toxic behavior. Another false positive of the BiLSTM is:

'Might want to point out experts 99.9% of biologists with PhDs many of who have the same god, accept evolution. Jist because he is stupid or lies doesnt change the truth! 203.192.91.4'.

The model labels this as [toxic, obscene, insult] which is pretty far off from the true set of labels: none. This is a pretty important error to catch because if the model were deployed in the world, it might label non offensive comments with a religious connotation as toxic.

At the same time, the message includes words like stupid and lies which probably tripped up the model. This shows how important it is to analyze the output of machine learning or deep learning models because even if the model reports a high accuracy, it may be learning biased or insensitive predictors.

In other instances of false positives, I'd actually side with the model over the human annotators. For instance, the model identifies the following comment as toxic:

'Im 12 and i can understand it perfectly. You should learn english instead of making us dumb it down for you.'

The dataset provides no label for this comment. I would actually side with the model over the human annotators and say that the model was correct in identifying this as insulting.

4.4 False Negatives at 0.2 Threshold for BiLSTM

As for false negatives, an example is the following comment:

Fuck you Buddy!!! Go suck a cock!

The model labeled this as [toxic, obscene, insult] while the human annotators labeled it as [toxic, severe toxic, obscene, insult]. The difference in the true label set is just + severe toxic so I would consider this to be a pretty small miss. A large miss is below:

Fine. I will destroy you.

The model gave this comment no label while annotators labeled it as [toxic, threat]. The model may have mislabeled this because the language is not harsh — though it definitely is a threat.

All in all it has been quite valuable to analyze mislabeled data: it can show confusion points, biases, questionable human annotations, and more. Generally, if a comment was mislabeled it was one label off from the correct annotation (ex. 2 instead of 3 labels).

4.4.1 Choosing the best deep learning model

For both of my deep learning models, I trained for a number of epochs and kept the model with the highest cross validation accuracy. Figure 6 shows how the LSTM's accuracy on the training and validation set performed over each epoch. Figure 7 shows the same for the loss on each dataset.

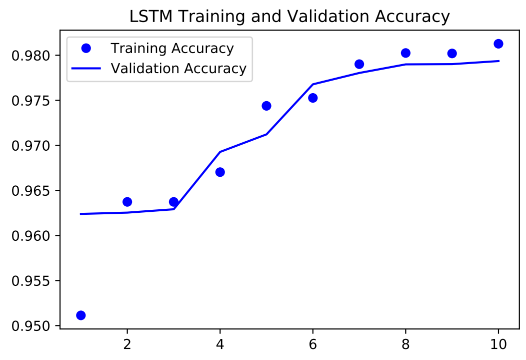


Figure 6: LSTM: Accuracy vs Epochs

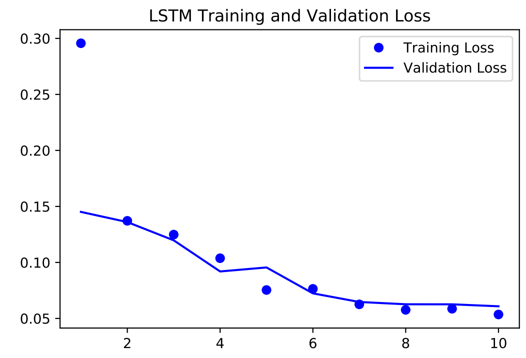


Figure 7: LSTM: Loss vs Epochs

Label	Top Word Features
toxic	ridiculous, adam, frequent, sockpuppet, inapproriate, nonsense, swift, bluesband, transferred, 24hrs, sarcastic, abstain, awkward, implement, expire, nambla, bespoke, rubbish, saxmunds, silly
severe toxic	is, newcomer, retardeds, sexless, create, mostly, duded, condescending, bandwidth, garbage, hanging, uneducated, words, tasty, indrian, infinite, wolfkeeper, stick, insurance, tempted
obscene	worldwide, participation, von, sex, xxx, american, wouldn, infinitely, tawker, muthafuckaa, tuesday, danger, moron, stories, reversion, value, puss, fegget, 2014, horseshit
threat	killing, earthquake, sitush, hang, bowel, hot, shoulder, 6ft, cut, trying, introducing, children, happen, dies, dammed, ll, jews, wh, 210, bw
insult	am, retardedyour, gg, reversion, theories, input, protestant, edjohnston, sloppy, title82, rubbish, warmongering, revandalising, quotemining, fucky, browsed, participation, worldwide, infinitely, rescind
identity hate	am, super, sexual, wikipeida, country, barack, filled, israeli, communist, shanghai, vegans, dutch, sized, expunged, brahmin, whitewash, transgender, morgan, niggaz, polar

Table 3: Top word features for each label

We can see that accuracy on both training and validations sets went up each time, indicating that the model was becoming both more accurate and more generalizable. After the 6th iteration though, the accuracy and loss began to plateau with gains in the training set outpacing gains in the test set — a sign that the model was heading towards over fitting so I chose to stop at 8 epochs.

The same accuracy and loss graphs for the BiLSTM can be found in Appendix A.2.

4.4.2 Strongest Features in NB+LR

Table 3 shows a list of the Naive Bayes + Logistic Regression's top 20 weighted word features for each class. The list definitely shows some noise but there are clear patterns that indicate how the model differentiates each label. For instance, the `identity hate` model highly weighted words related to races, ethnicities, and cultures: **barack, israeli, communist, shanghai, vegans, dutch, brahmin, whitewash, transgender,** and **niggaz** were all in the top twenty.

Other labels had similar patters: threats involved many violent concepts (**killing, hang, dies, cut**), obscenity centered around sex and curse words (**sex, xxx, muthafuckaa, moron, puss, fegget**), insults were a blend of wikipedia specific terms (**reversion, revandalising, quotemining**) and curses (**retardedyour, rubbish, fucky**) and toxic word features were generally less harsh words than severely toxic words which is a good sign that the two are separable. Across the labels, some word misspellings receiving very high weightings. That indicates that toxicity and poor spelling may be connected and in the future misspellings could be preprocessed or incorporated into another feature.

4.5 Discussion

In general, I found that from a Naive Bayes classifier to a bidirectional LSTM neural network the more complex models obtained higher accuracy scores on this task. I wasn't certain that this would be true, but in this case it seems like the multilabel classification problem was difficult enough that more expressive models performed better.

One of the more interesting model comparisons is that the NB+LR model stacks up quite well against even the BiLSTM: their accuracy difference is just -2%. That is consistent with the findings from Wang and Manning (2012) which argued that a well designed NB+LR model can perform at the level of deep learning models.

On the other side, there wasn't a major difference between the LSTM and BiLSTM. Stacking two LSTMs to create a Bidirectional LSTM definitely increased training time but did not lead to a major accuracy gain ($+1\%$) over using a single LSTM.

With more advanced feature engineering or stronger word embeddings it is likely that almost all models above would perform near 100% or at the level of human disagreement. As the false positive and false negative analysis section showed, human annotations are not always in agreement. At that point, one could argue that this problem is solved and using more complex deep learning models would be unnecessary. In this project, I chose to build my own embeddings off the training data as that outperformed word2vec (Mikolov et al., 2013) and GLoVe (Pennington et al., 2014) while providing slightly worse accuracy than FastText (Joulin et al., 2016). Using larger vector representations or training on top of FastText could be a valuable future experiment.

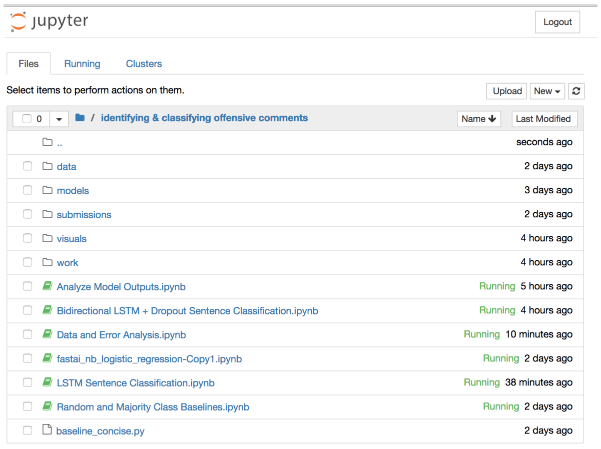
If we consider the Kaggle rankings, the best models of all performed with roughly a 98.8% average ROC AUC — and these submissions likely consists of an ensemble of deep learning models. When compared to my vanilla BiLSTM, they provide a +2% boost.

5 Conclusion

This turned out to be a challenging, yet rewarding, task. I obtained results with four different models from both classic and deep machine learning plus the accuracy scores were high enough that these models could plausibly be deployed into digital communities. I'm excited to see where automated tools for detecting online harassment will go and I hope they can help make the internet a safer place for everyone.

5.1 Open Source Jupyter Notebooks

As mentioned previously, all of the code involved in this project is available at `http://github.com/jLukeC/cleaning-up-comments`. In the repository you will find Jupyter notebooks that step through every model along with the evaluation script and data & error analyses. Everything is run in `python3` and the libraries necessary to run the notebooks include `jupyter`, `pyplot`, `keras`, and `pandas`. Feel free to point out any improvements or add a pull request with your own additions.

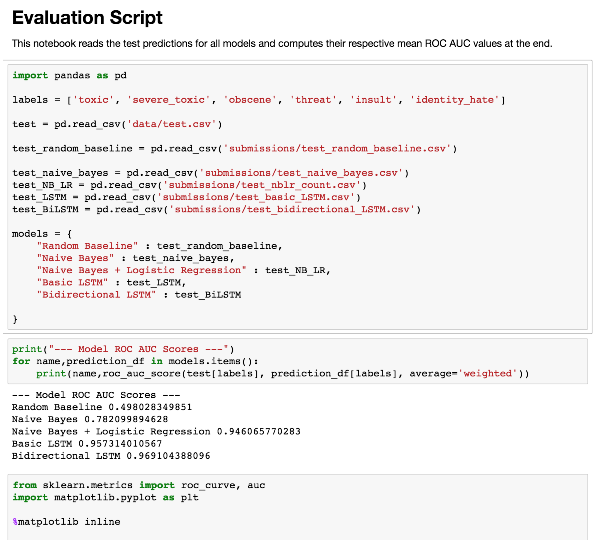


Select items to perform actions on them.

/ identifying & classifying offensive comments

..		seconds ago
data		2 days ago
models		3 days ago
submissions		2 days ago
visuals		4 hours ago
work		4 hours ago
Analyze Model Outputs.ipynb	Running	5 hours ago
Bidirectional LSTM + Dropout Sentence Classification.ipynb	Running	4 hours ago
Data and Error Analysis.ipynb	Running	10 minutes ago
fastai_nb_logistic_regression-Copy1.ipynb	Running	2 days ago
LSTM Sentence Classification.ipynb	Running	38 minutes ago
Random and Majority Class Baselines.ipynb	Running	2 days ago
baseline_concise.py		2 days ago

Figure 8: Jupyter Notebook Overview



Evaluation Script

This notebook reads the test predictions for all models and computes their respective mean ROC AUC values at the end.

```python
import pandas as pd

labels = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']

test = pd.read_csv('data/test.csv')

test_random_baseline = pd.read_csv('submissions/test_random_baseline.csv')

test_naive_bayes = pd.read_csv('submissions/test_naive_bayes.csv')
test_NB_LR = pd.read_csv('submissions/test_nblr_count.csv')
test_LSTM = pd.read_csv('submissions/test_basic_LSTM.csv')
test_BiLSTM = pd.read_csv('submissions/test_bidirectional_LSTM.csv')

models = {
    "Random Baseline" : test_random_baseline,
    "Naive Bayes" : test_naive_bayes,
    "Naive Bayes + Logistic Regression" : test_NB_LR,
    "Basic LSTM" : test_LSTM,
    "Bidirectional LSTM" : test_BiLSTM

}
```

```python
print("--- Model ROC AUC Scores ---")
for name,prediction_df in models.items():
    print(name,roc_auc_score(test[labels], prediction_df[labels], average='weighted'))
```

```
--- Model ROC AUC Scores ---
Random Baseline 0.498028349851
Naive Bayes 0.782099894628
Naive Bayes + Logistic Regression 0.946065770283
Basic LSTM 0.957314010567
Bidirectional LSTM 0.969104388096
```

```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

%matplotlib inline
```

Figure 9: Notebook Example: Evaluating Model Submissions

References

Mike Butcher. 2017. *Unless online giants stop the abuse of free speech, democracy and innovation is threatened*. Tech Crunch.

Noah Cohen. 2018. *Conspiracy videos? Fake news? Enter Wikipedia, the good cop of the Internet*. Washington Post.

Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. 2018. Densely connected bidirectional LSTM with applications to sentence classification. *CoRR*, abs/1802.00889.

Maeve Duggan. 2017. *Online Harassment*. Pew Research Center.

Francisco Herrera, Francisco Charte, Antonio J. Rivera, and Mara J. del Jesus. 2016. *Multilabel Classification: Problem Analysis, Metrics and Techniques*, 1st edition. Springer Publishing Company, Incorporated.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. 9:1735–80.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification.

Jason Koebler. 2017. *The Reddit Moderator Getting a PhD in Online Moderation*. Vice.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the ACL*, pages 90–94.

Jenna Wortham. 2018. *Why Cant Silicon Valley Fix Online Harassment* New York Times.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1391–1399, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

A Supplemental Material

A.1 ROC Charts

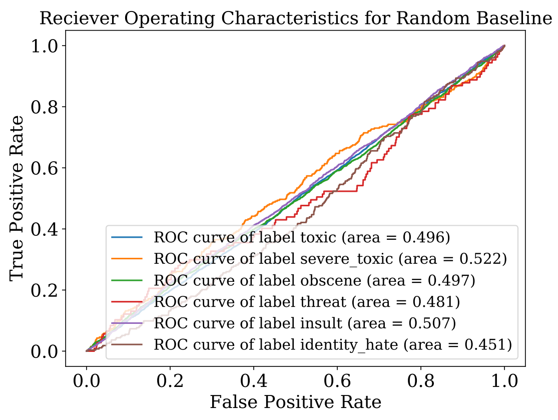


Figure 10: ROC for Random Baseline

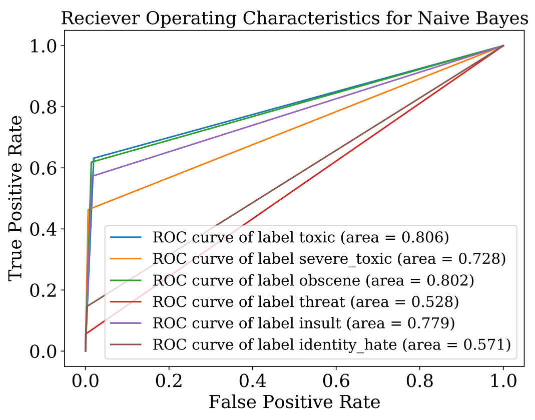


Figure 11: ROC for Naive Bayes

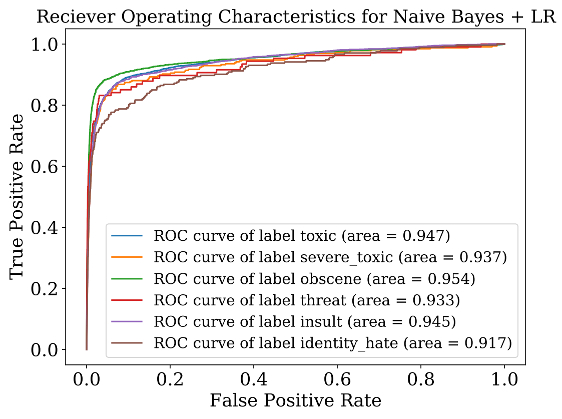


Figure 12: ROC for Naive Bayes + Logistic Regression

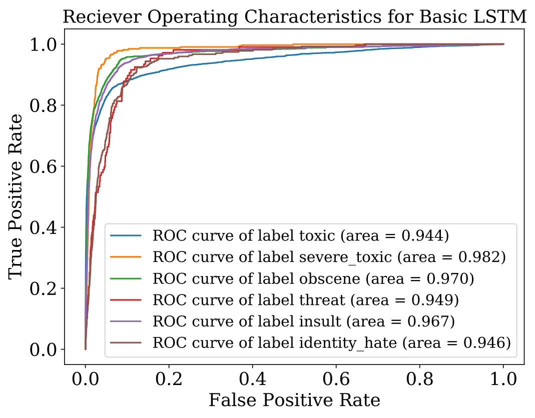


Figure 13: ROC for LSTM

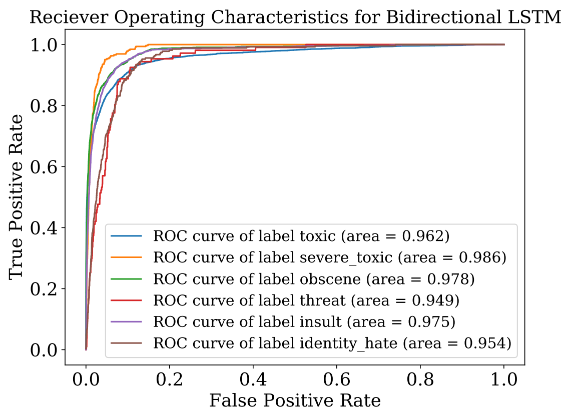


Figure 14: ROC for BiLSTM

A.2 Deep Learning Accuracy and Loss Charts

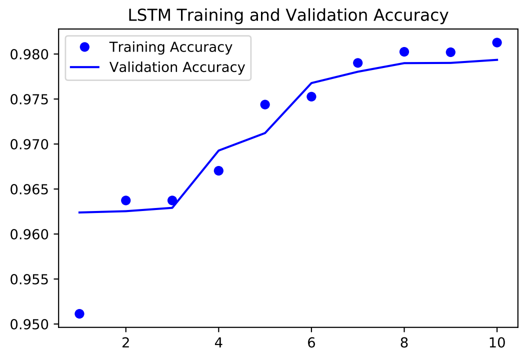


Figure 15: LSTM: Accuracy vs Epochs

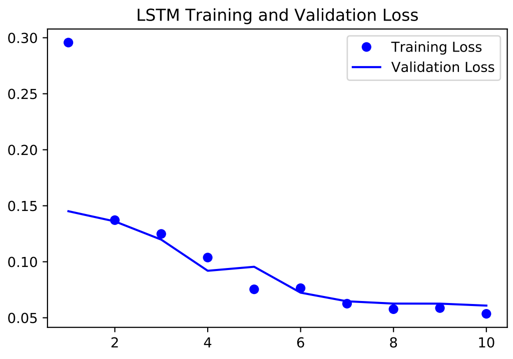


Figure 16: LSTM: Loss vs Epochs

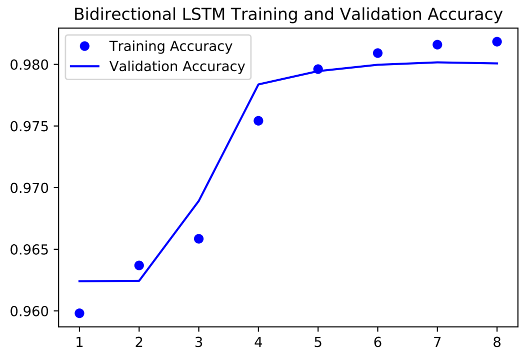


Figure 17: BiLSTM: Accuracy vs Epochs

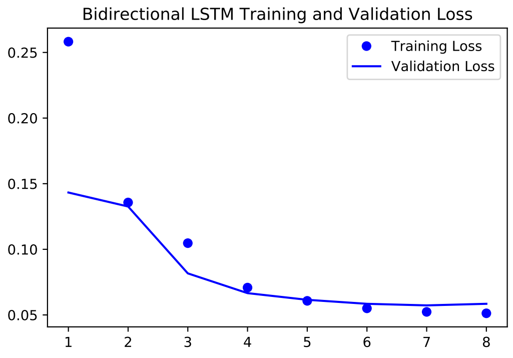


Figure 18: BiLSTM: Loss vs Epochs

A.3 Example Comments

Comment Text	Labels
Alright, well do us all a favor. FUCK OFF! No one wants to read these articles and see that citation bullshit all over the page. That stuff has been there for years and no one gave a rats ass until now. Seriously dude, fuck off.	toxic, insult
...It is extremely silly for Wikipedia to label a male transvestite as a ""she."" It is impossible to change one's sex, you idiotic communist pieces of shit. Science trumps your bizarre fundamentalist socialist religion. Stop enabling people who suffer from gender identity disorder. It is harmful to their health...	toxic, obscene, insult, identity hate
FUCK THE POOR BRITAIN IS SHITTTTTTTTTTTTTTTTTTTTTTTTT	toxic, severe toxic, obscene, insult, identity hate
Edit this again and I will hack your computer, destroy your C drive and then trace your IP address to find where you live...	toxic, threat
I'm going to hunt you down and murder you violently, you disgusting paki twat	toxic, obscene, threat, insult, identity hate
JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!! JIM WALES MUST DIE!!!!!!!!!!!!	toxic, severe toxic, threat
I hope World of Warcraft dies a long and painful death World of Warcraft sucks the biggest donkey balls on this planet. I cannot express how much I hate this piece of wank other than hoping it is raped, stabbed, drowned, raped again when it's dead, drown it in its own blood from being stabbed. Afterwards, it should be buried in a 150 feet deep pit full of maggots and when they have finished with it, the whole pit should be set alight with 20,000 matches.	toxic, obscene, threat, insult
HOPE YOUR HEAD GETS CUT OFF AND SOMEONE WIPS THERE ASS WITH IT AND THEN STABS YOU IN YOUR HEART	toxic, severe toxic, obscene, threat, insult
you're hot i will rape you ;)	toxic, severe toxic, obscene, threat, insult