Paying attention to toxic comments online

Manav Kohli*
Intuit
manav_kohli@intuit.com

Emily Kuehler
Stanford University
ekuehler@stanford.edu

John Palowitch
Google
palowitch@google.com

Abstract

Deep learning methods have recently begun to be used to detect abusive comments made in online forums. Detecting, and classifying, online abusive language is a non-trivial NLP challenge because online comments are made in a wide variety of contexts, and contain words from many different formal and informal lexicons. Furthermore, spelling and grammar mistakes (many of them intentional) abound. In this paper, we examine and implement baseline and existing approaches to the task of classifying online abuse, and also introduce and analyze variants of the existing models. Our goal is to provide a scientifically rigorous perspective on the strengths and weaknesses of the range of approaches. As such, we apply each method to two different data sets, and provide in-depth visualizations of model performance and explanatory wins and losses. Additionally, we provide a fully reproducible open-source code repository.

1 Introduction

As online news, social, and gaming platforms continue to grow their user bases, user-generated content moderation has become an increasingly important task. This is especially true in our era of heightened political polarization and elevated awareness of the effect of hate speech on marginalized communities. In a recent Pew survey, 39% of experts and leaders in fields related to online discourse said that the future will be more shaped by harassment and troll-like behaviors, versus 19% who said it would be less shaped [10]. Most companies that rely on online content now deploy content-moderation systems that are at least partially automatic, and research groups have begun to direct programs toward this issue (e.g. the Jigsaw team from Google).

Detecting and controlling online verbal abuse in an automated fashion is an inherently natural language processing task. Recently, deep neural networks have been explored as potential engines for abusive and toxic comment detection. Current research papers on this topic has taken one of three flavors: (i) introduction or analysis of text-based toxicity detection models, (ii) development of toxicity detection approaches which rely heavily on user metadata or non-text sources, or (iii) discussion and comparisons of methods for preprocessing online comment data for NLP task related to abuse detection. We discuss the particular publications in these categories in the next section. Our work

*Authors are listed alphabetically

falls into the first category; we discuss the state-of-the-art models for toxic comment detection based solely on textual information. This paper's contributions to this area are three-fold:

- We provide a rigorous and reproducible application of a wide array of baseline and existing models on two data sets
- We introduce novel character, word, and sentence-level RNN and assess the added-value of attention mechanisms in some cases
- As our data sets involve toxic comments that are *sub-classified* into types, allowing for multiple types, we discuss the modeling nuances this entails

The remainder of this paper is organized as follows. In Section 1.1 we discuss existing work. In Section 1.3 we discuss the nuances of using multiple-type training examples and the resultant modeling strategies we employ. In Section 2 we discuss and show results from baseline models on the data sets. In Section 3 we discuss and show results from word-based recurrent neural networks. In Section 4 we discuss and show results from character-based recurrent neural networks. In Section 5 we discuss the training of custom word embeddings, and use them with the best versions of our approaches discussed in prior sections. In Section 6 we give a summary of our results.

1.1 Existing work

A recent well-cited non-neural approach to toxic comment detection introduced an NLP regression classifier to the n-gram, linguistic, syntactic, and semantic features of comments posted on Yahoo! Finance and News [7]. Separately, the authors of [12] obtained human ratings on a large corpus of Wikipedia comments, then used ML approaches with all combinations of word/character n-grams, logistic/multilayer-perceptron (MLP), and discrete/continuous rater labels. A survey of other approaches to online abuse detection was given in [11], containing a summary of common features used for non-neural and early neural approaches.

The authors of [6] compared deep learning approaches to the existing non-neural methods, including that of [7]. They saw gains over existing methods from word and character recurrent neural networks (RNN), as well as from a support vector machine (SVM) classifier supported by naive bayes (NB) scores on character counts. The authors of [2] applied deep learning comment classification methods to a large Twitter data set. They used convolutional neural networks (CNN), long-short-term memory networks (LSTM), and both SVMs and gradient-boosted decision trees (GBDT) with word and character embeddings trained during the deep learning model fitting. The authors of [8] trained RNNs and RNNs with attention mechanisms on (i) a large corpus of Greek news site comments and (ii) Wikipedia talk page comments, finding that these deep learning approaches outperformed the word/char logistic/MLP models used in [12]. Finally, a class project from Stanford's 2017 section of CS224n used the same Wikipedia data set analyzed in [12] with an LSTM network and both a character and word based CNN [3].

1.2 Data sets

The first data set we use is constructed from two corpi from the Wikipedia Detox data set: a collection of Wikipedia.org comments with boolean decisions from raters on whether or not the comments were *aggressive* [13], and those same comments with boolean decisions from (different) raters on whether or not the comments represented *personal attacks* [14]. As in [12], we use the majority vote from multiple raters to decide the final comment label. Recall that this data set was also analysed in [8] and [3], making this a good baseline dataset on which to test our approaches. We combine the labels of both classes into one data set, so that some comments have multiple labels. Each class comprises about 10% of total comments, and the classes have about 0.89 Pearson correlation. The second data set we use has not to our knowledge been analyzed in any publication. The data is taken from a Kaggle competition, hosted by Google's Jigsaw team, for classifying Wikipedia comments into any combination of six types of "toxicity": toxic, severe toxic, obscene, threat, insult, and identity hate. These labels were again obtained by human ratings.

The Detox data set had over 100k labeled examples, and the Jigsaw data set had over 160k labeled examples. We shuffled the examples and split each data set into a 3:1:1 train, dev, and test partition. The partitions were fixed across the various models with a random seed. The dev set was used to

choose the best set of model parameters across training epochs performed on the train set. Final evaluation was conducted on the test set.

1.3 Sigmoid multi-membership model

The data sets we analyze come with the challenge of potentially multiple categories per training and testing example. One easy solution to this problem is to fit one model for each class in a one-vs-all fashion. This approach has the obvious benefit of being able to optimize cost on each class directly. However, a disadvantage to this approach is training time: six models need to be fit, instead of (possibly) just one. Furthermore, a comprehensive deep learning model could learn important correlations between classes, boosting overall accuracy.

We can account for the presence of multiple classes per example by putting the final logit layer of a classification model through an entry-wise *sigmoid* instead of a softmax. Correspondingly, we use a *binary* cross-entropy score rather than the standard softmax cross-entropy. For example, let h be the final logit layer of a model. In an application to the Jigsaw data, h would be 1×6 row vector. Our final prediction vector in a sigmoid model would be $\hat{y} = \sigma(h)$ where σ is the entry-wise sigmoid function. The corresponding loss function is

$$C(y, \hat{y}) := \sum_{k=1}^{K} -y_k \log(\hat{y}_k) + (1 - y_k) \log(\hat{y}_k)$$

Note that this loss function reduces to standard cross-entropy loss when there is just one correct class. The sigmoid model allows multiple values of $\hat{y}$ to be near 1, and therefore can learn when to assign multiple classes high probabilitiy. This will be useful when, as just one example, a set of comments in the training data are labelled both "toxic" and "obscene".

1.4 Evaluation metric

The evaluation metric we chose is the class-wise average of the area under the receiver operating characteristic curve (AUC), which is the measure of classification accuracy integrated over classification thresholds. Explicitly, define the two dimensional function

$$f(t) = (r, p)$$

where $t \in [0, 1]$ and r, p the recall and precision (averaged over multiple classes). The AUC metric is defined as the area under the curve defined by the (r, p) pairs produced by moving t continuously from 0 to 1. Mathematically speaking, this metric is equivalent to the probability that a randomly chosen positive example has a higher predicted probability (for the positive class) than a randomly chosen negative example (for the same positive class).

2 Baseline models

As baseline models we trained both a naive bayes (NB) model and logistic regression (LR) model on each class. Both models were trained on sparse sentence vectors with the 10,000 most-common 1,2-$n$gram token features. For comparison, we also trained a sigmoid version of LR:

	Classifier	identity_hate	insult	obscene	severe_toxic	threat	toxic	averages
0	sigmoid	0.9576	0.9623	0.9741	0.9809	0.9791	0.9445	0.966417
4	logistic	0.9650	0.9715	0.9812	0.9824	0.9797	0.9608	0.973433

	Classifier	aggression	attack	averages
0	sigmoid	0.9445	0.9809	0.96270
4	logistic	0.9424	0.9495	0.94595

Table 1: Baseline results for Jigsaw data (top) and Detox data (bottom)

Interestingly, the sigmoid model performs slightly worse across classes on the Jigsaw data, but better across classes on the Detox data. To save both space and training time, we treat the sigmoid model as default.

3 Word-level RNNs

We built and trained a variety of word-level recurrent neural networks, each sharing the following characteristics. Token-level sentence vectors were computed from the top 30,000 tokens. Sentences were padded or (non-randomly) truncated to length 150. Tokens in each sentence were given pre-trained embeddings from GloVe vectors as provided by [9]. The Adam optimizer [4] was used with an initial learning rate of 0.001, decreasing by a factor of 0.95 every 100 steps. With a batch size of 32, there are about 2,000 steps on the Detox training data and 3,000 steps on the Jigsaw training data. All networks mentioned in this section were built using standard library functions from `TensorFlow` [1]. Our baseline RNN employs both mean and max pooling; explicitly:

- $e_t =$ is word t's embedding
- $h_t =$ is the t-th output of the RNN cell; $H =$ is the full matrix of RNN outputs
- h_{mean}, h_{max} are the mean and max of the RNN outputs along the time dimension
- $\theta = U[h_{mean}; h_{max}] + b$ is the logit vector. Note U has row-dimension equal to # classes.

We focused on two network "flavors", bi-directional (BD) and attention (ATTN), fitting gated recurrent unit (GRU) and long-short-term memory (LSTM) versions of each. For the BD network, h_t is the concatenated vector containing the output vectors of word t from both a forward and backward run through the sentence. For the ATTN network, we added a 150×150 weight matrix W_a to the network, and computed two attention vectors:

$$a_{mean} = \text{softmax}(h'_{mean} W_a H), \quad a_{max} = \text{softmax}(h'_{max} W_a H)$$

Then the final dense layer is instead given $[h_{mean}; h_{max}; Ha_{mean}; Ha_{max}]$. Note that when the network is *also* bi-directional, each of those vectors are twice their usual length. Our best results came from RNNs with both BD and ATTN mechanisms, seen in Tables 2-3, though the gains were extremely incremental. We print results from RNNs with only BD or ATTN in the Supplemental. We provide further commentary in the Discussion (Section 6).

	cell	identity_hate	insult	obscene	severe_toxic	threat	toxic	averages
8	gru	0.9613	0.9835	0.9879	0.9860	0.9630	0.9748	0.976083
9	lstm	0.9569	0.9834	0.9875	0.9865	0.9617	0.9732	0.974867

	cell	aggression	attack	averages
9	gru	0.9621	0.9683	0.9652
10	lstm	0.9621	0.9687	0.9654

Table 2: Un-flavored RNN results (top:Jigsaw, bottom:Detox)

	cell	identity_hate	insult	obscene	severe_toxic	threat	toxic	averages
14	gru	0.9745	0.9853	0.9887	0.9882	0.9763	0.9776	0.981767
15	lstm	0.9736	0.9850	0.9889	0.9882	0.9755	0.9775	0.981450

	cell	aggression	attack	averages
15	gru	0.9642	0.9699	0.96705
16	lstm	0.9640	0.9697	0.96685

Table 3: RNN results with bi-directional and attention mechanisms (top:Jigsaw, bottom:Detox)

For each data set, we examined the top-100 comments from three categories: truly-toxic comments with the lowest losses (t-wins), truly-toxic comments with the highest losses (t-misses), and truly-safe comments with the highest losses (s-misses). These helped us look at the features of truly-toxic

comments which were best-learned (t-wins), the features of truly-toxic comments which caused mistakes (t-misses), and the truly-safe comments which probably could have been toxic had the raters voted slightly differently (s-misses). For each of these top 100 comments, we also printed the 5 tokens from the sentence with the highest a_{mean} attention scores. The full comment lists, and some discussion, can be found in the supplemental docs.

4 Character-level RNNs

One of the challenging aspects of toxic comment classification is that comments are unedited and slang-filled. The most appealing feature of a character-level model in this case is that this issue of unusual vocabulary words is sidestepped. Due to the small number of English characters, we also eliminate the need for a large embedding matrix. Two approaches were taken in constructing the inputs to the character-level model. Our first approach was to simply one-hot encode each character, while the second approach used dense character vectors constructed from pretrained GloVe word embeddings. Our word vector to character vector process consisted of the following steps: for all occurrences of a given character in the GloVe vocabulary, we take the sum of the word vectors for which each character has occurred and divide by number of occurrences of given character. As a toy example, if our vocabulary were 'the, dog, eats' and we wished to find the character embedding for 'e' we would take the mean of the word embeddings of 'the' and 'eats.' This method was able to leverage the large dataset that the GloVe vectors were trained on and capture similarities between characters. The method was evaluated using Principal Component Analysis, which allowed us to visualize the characters in two dimensions. Numerals 0-9 were grouped together, as were the majority of capital-lower case pairs.

Architecturally, the character-level models were very similar to the word-level RNNs, substituting our character embeddings for the pretrained GloVe embeddings (see Section 3). However, for the character-level models we used only max-pooling and increased the number of neurons in our hidden layer to 256. As with our word-level model, we compare a gated recurrent unit (GRU) and long-short-term-memort (LSTM). The results are presented in the tables below and we elaborate in our discussion section.

	cell	identity_hate	insult	obscene	severe_toxic	threat	toxic	averages
1	gru50	0.8887	0.9082	0.9099	0.9499	0.9188	0.8823	0.909454
2	lstm50	0.8761	0.8886	0.8921	0.9381	0.8966	0.8681	0.893289

	cell	identity_hate	insult	obscene	severe_toxic	threat	toxic	averages
1	gru150	0.9299	0.9558	0.9570	0.9754	0.9447	0.9273	0.948344
2	gru200	0.9384	0.9564	0.9604	0.9792	0.9481	0.9325	0.952525

5 Experiments with custom embeddings

Our first approach to training the baseline models relied on frequency based tokenizers. While this enabled us to quickly iterate on the architectures during development, it did not provide easily accessible insights into how the sentences were being modeled and what types of words assumed similar meanings in the comments. This shortcoming, coupled with our interest in building vectors more tailored to our corpus, motivated us to training our own GloVe model.

Our training corpus contained 210,554 tokens and we used either the top 10,000 or 100,000 most frequently occurring ones, compared to 6 billion and 400,000 in [9]. This inherently skewed how our vectors were trained since common words often occurred with low frequencies in our corpus. For example, the token "woman" did not break into the top 1,500 most frequently occurring ones. Because of the nature of context based models, this clearly limited how well the true meaning of "woman" was interpreted at testing time.

The custom embeddings were assessed using T-distributed Stochastic Neighbor Embedding (t-SNE) [5] to visualize them in two dimensions. Figure 1 displays a small collection of words that include common tokens, analogy pairs, and typically positive or negative words.

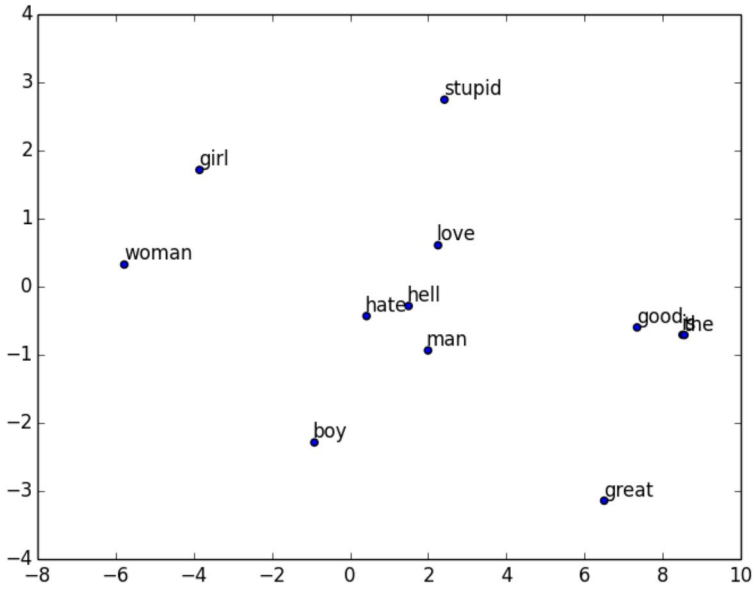

Figure 1: t-SNE Visualization of Custom Embeddings

As a validation, commonly used words shared the same vector space (”is”, ”the”). While it developed some intuition for the analogy task, it also faced difficulty as the distance between ”woman” and ”girl” was nearly inverse to that between ”man” and ”boy”. This may highlight how the words are used much differently in the context of these comments, such as how ”man” is used to address someone else in a conversation.

The experiments we ran varied a combination of the following hyperparameters:

- Embedding size, 50 (default) or 100
- Learning rate, 0.05 (default) or 0.1
- Number of most frequently occurring words, either 10,000 (default) or 100,000
- Window size (symmetric), 5 (default) or 10

We assessed the accuracy of each experiment by using the embeddings for classifying comments with a logistic regression baseline. This enabled us to quickly compare the generated embeddings against each other during development as well as in isolation of any performance changes due to the RNN models.

As [9] found changing the dimensionality of the embedding resulted in one of the most significant increases in accuracy, the first experiments ran compared the performance of embeddings while maintaining a constant number of features. Table 4 includes results from a subset of the experiments run with an embedding size of 50.

Table 4: AUC by Window Size, Learning Rate, and Vocabulary Length

	toxic	severe_toxic	obscene	threat	insult	identity_hate	average
10/0.05/10K	0.6722	0.7773	0.7192	0.7184	0.7063	0.7451	0.7231
5/0.05/10K	0.6709	0.7472	0.7017	0.7568	0.7048	0.7467	0.7214
10/0.1/10K	0.6557	0.7858	0.7069	0.7207	0.6937	0.7191	0.7137
10/0.05/100K	0.6780	0.7868	0.7130	0.7586	0.7097	0.6979	0.7240

While increasing the number of words under consideration resulted in a marginally higher average AUC, it was also at the cost of a much longer training time. Increasing the learning rate saw the worst performance and may have been caused by overfitting to the training set, which was only exacerbated by limiting the vocabulary to the 10,000 most frequently occurring tokens. Reducing the window size by 5 offered the most attractive approach since it significantly reduced the training time and maintained a similar level of accuracy.

The best performing embeddings achieved an AUC score of 77.01% with the logistic regression baseline. Barring reducing the window size to 5 and increasing the embedding size to 100, the rest of the hyperparameters remained at the default values.

Retaining accuracy while reducing the window size also suggests that toxic tokens affecting each of these categories occur near each other. For example, accusatory statements such as "this is blatant discrimination and gang behavior" or personal attacks like "number 57 is an absolute disgrace" suggest that decreasing the window size could retain comparable predictive information while offering a significant efficiency boost.

We then tried initializing the custom embeddings with pretrained GloVe vectors. This approach resulted in a huge increase in performance and the average AUC jumped to 91.69%. Unsurprisingly, incorporating the meaning of words from much broader contexts significantly helped our model by incorporating information from more experience. See the supplementary material for a t-SNE visualization of the same words as were included in Figure 1. Table 5 includes the results for using those embeddings with the word-level RNNs results.

	cell	identity_hate	insult	obscene	severe_toxic	threat	toxic	averages
2	gru	0.9669	0.9829	0.9872	0.9873	0.9683	0.9732	0.9776
4	lstm	0.9674	0.9831	0.9872	0.9869	0.9698	0.9726	0.9778

Table 5: RNN Results on Toxic Dataset with Custom Embeddings

6 Discussion

In this paper we built a wide range of recurrent neural network models for the task of classifying abusive comments authored by users of Wikipedia. This is a challenging NLP task because the number of positive examples is low, comments were allowed to be in multiple classes, and the text of the comments contain highly non-standard vocabulary. Our first finding was that, surprisingly, non-neural baseline models based on TFIDF sentence vectors perform quite well. One hypothesis for this is that expletives are highly indicative of abuse, and often easily tokenized (this hypothesis was supported by analysis of attention scores and comments with low losses from the RNN models). Standard GRU and LSTM RNNs based on pre-trained word embeddings offered marginal improvement over the non-neural baseline models, and bi-directional and attention mechanisms again add marginal improvement over the baseline RNNs. A future research area for word-level models is to examine which particular types of words had high attention scores for particular comment classes. This could yield deeper insights into the character of online abuse.

Character-level models offer a potential way to addressing the out of vocabulary problem common in online comments, due to the high frequency of slang and lack of edits. However, breaking the comment into its character level token increases the total number of tokens by a magnitude of approximately 5 as compared to word-level tokens, thus requiring significantly more training time to reach the same amount of data. While there is a training time increase, we see a decrease in memory required as the character embedding matrix is much smaller than the corresponding word matrix. Even through the use of character embeddings inferred from GloVe word embeddings, performance of the character-level model was not on par with that of the word-level models. Given the similarity of the architecture we employed between the two models, this supports the notion that expletives and slurs are highly indicative of abuse, a baked in feature unavailable to the character-level model. A future avenue of research to pursue would be an architecture that attempts to infer word-level features from character-level input.

Using custom word embeddings saw a marginally low decrease in the word level GRU and LSTM accuracies. This may suggest that training on the corpus skewed the meanings too much towards the training set. Since only the top 10,000 most often occurring words were considered when building the cooccurrence matrix, some variants of toxic words may have been given less focus despite having significant impacts on the overall sentiment. Given more computational resources, varying the size of the most frequent words considered and reducing the threshold for how many times a pair must occur could shed more light onto this gap. Another area to pursue further could be a deeper analysis of the differences between the vectors initialized randomly and those not, specifically looking at how they converge during training. Through comparison, a stronger intuition could be derived as to how the words were represented in the context of the corpus.

Through experimentation and developing state of the art models we were able to achieve a high accuracy with predicting toxic comments. With finer tuning and a deeper look into efficiency gains, we hope that our models could be deployed in production environments to catch inflammatory comments before they reach vulnerable users. Moreover, with online communities and internet-connected devices increasingly accessible to younger audiences, we hope that research like this will contribute to creating a safer environment for all.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee, 2017.

[3] T. Chu, K. Jue, and M. Wang. Comment abuse classification with deep learning. 2017.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[6] Y. Mehdad and J. Tetreault. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, 2016.

[7] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee, 2016.

[8] J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, 2017.

[9] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation, 2014.

[10] L. Rainee, J. Anderson, and J. Albright. The future of free speech, trolls, anonymity and fake news online, 2017.

[11] A. Schmidt and M. Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, 2017.

[12] E. Wulczyn, N. Thain, and L. Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee, 2017.

[13] E. Wulczyn, N. Thain, and L. Dixon. Wikipedia talk labels: Aggression. 2 2017.

[14] E. Wulczyn, N. Thain, and L. Dixon. Wikipedia talk labels: Personal attacks. 2 2017.