# Comment Abuse Classification with Deep Learning

**Theodora Chu**
theodora@stanford.edu

**Kylie Jue**
kyliej@stanford.edu

**Max Wang**
maxwang7@stanford.edu

## Abstract

Social media platforms, online news commenting spaces, and many other public forum of the Internet have become increasingly known for issues of abusive behavior such as cyberbullying and personal attacks. However, determining whether or not a comment or post should be "flagged" is difficult and time-consuming, and many platforms are still searching for more efficient moderation solutions. Automating the process of identifying abuse in comments would not only save websites time but would also increase user safety and improve discussions online. In this paper, we use Wikipedia talk page data in order to train deep learning models to flag comments. Comments are classified as personal attacks or not personal attacks, and ratings were determined by Crowdflower workers. We tested three models: a recurrent neural network (RNN) with a long-short term memory cell (LSTM) and word embeddings, a convolutional neural network (CNN) with word embeddings, and a CNN with character embeddings. Our models improve upon previous results from non-deep learning machine-learning models, and we find that a CNN with character-level embeddings reaches the highest performance.

## 1 Introduction

As discussions increasingly move toward online forums, the issue of trolls and spammers is becoming increasingly prevalent. Manually moderating comments and discussion forums can be tedious, and to deal with the large volume of comments, companies often have to ask employees to take time away from their regular work to sift through comments or are forced to hire contracted or outside moderators. Without careful moderation, social media companies like Reddit and Twitter have been criticized for their enabling of cyberbullying, and users can abuse the platforms for unethical practices like doxing and identifying political dissidents.

According to a 2014 Pew Research Institute survey, 40 percent of internet users have personally experienced online harassment, and 45 percent of those harassed have experienced "severe harassment" (i.e. physical threats, sustained harassment, sexual harassment, etc.) [1]. In extreme cases, cyberbullying can even lead to victims' committing suicide [2]. Although most discussion platforms allow users to report abusive comments, the industry is looking at ways to further actively automate these processes, rather than passively relying on crowdsourced moderation through other commenters.

Since researchers in natural language processing (NLP) and machine learning traditionally haven't applied their research to the area of commenting spaces and abuse, we hope this paper will provide more insight into that space. We want to apply NLP and deep learning to determine whether or not a comment should be classified as abusive. We use linear regression as our general machine-learning baseline and also run a feed forward neural network to provide a deep-learning baseline. We then train two CNNs with different feature embeddings and an LSTM to determine the most

salient features of abusive comments. Lastly, we fine-tune the models before testing them on our datasets.

Through our work, we wanted to determine if deep-learning models could be used to better determine whether or not a comment should be moderated. We also aim to answer the following question: Can comments be automatically moderated based on their "attack" scores?

The rest of the paper will proceed as follows: in section 2, we will describe relevant research that informed our questions and decisions. In section 3, we will explain our approach in more detail. Section 4 will talk about the experiments we ran, and we will conclude our paper in section 5 with recommended next steps for future work.

## 2 Related Work

### 2.1 Sentiment analysis for abusive comments

Comment abuse classification research with machine learning began with Yin, et al.'s paper, in which the researchers use a support vector machine and apply TF-IDF to the features [3].

More recently, research into applying deep learning to related fields such as sentiment analysis has proven quite fruitful. Recurrent neural networks have been known to perform well in sentiment analysis tasks since RNNs use a sequencing model. This worked well for Zhang and Lapata in poetry generation since word generation is dependent on the words that come before them [4]. Similarly, sentiment is often dependent on a sequential understanding of the words in a sentence. However, Bengio, et al. showed that one disadvantage of RNNs is their inability to keep track of long-term dependencies due to the vanishing gradient problem [5]. Wang, et al. used LSTMs to predict the polarity of Tweets and performed comparably to the state-of-the-art algorithms of the time [6]. Huang, Cao, and Dong found that hierarchical LSTMs allow rich context modeling, which enabled them to do much better at sentiment classification. Specifically, they chose to use LSTMs because LSTMs solve the aforementioned vanishing gradient problem. [7]

Other researchers have used convolutional neural networks in sentiment analysis. Nogueira dos Santos and Gatti experimented with CNNs using various feature embeddings, from character- to sentence-level. Because short texts have a limited amount of contextual information, the researchers hypothesized that the word embedding layer would be very important for sentiment analysis. With the two datasets that they used, they found that character-level embeddings performed better than the other embeddings on one dataset and performed equally as well on the other. [8] Mehdad and Tetreault added more insight into using character-level features versus word-level features through their research. Because language in comments and online forums often includes non-standard English (i.e. "a$$hole" or "l o s e r"), learning the interactions between neighboring characters may say more about the abusiveness of a comment as a whole than learning words does. [9] Furthermore, Kim explored different levels of parameter tuning and filters on CNNs applied to various sentence-level classification tasks, including sentiment analysis. He found that even the most basic CNN did quite well, although tuning hyperparameters and using pre-trained word vectors did result in some performance gains. [10]

It is clear that RNNs, specifically LSTMs, and CNNs are state-of-the-art architectures for sentiment analysis. Given the similarities between comment abuse classification and sentiment analysis, we hope to use this research to inform our approach and methodology, which we will talk about in section 3.

### 2.2 Existing comment datasets

Because abuse classification is still a relatively new research topic in NLP and there are legal and privacy issues with making this information public, few datasets have been curated specifically for this problem. However, to understand what makes up a good dataset in this space, we looked into datasets used by others doing similar research. Maus used a YouTube.com comment dataset that had a little over 2500 manually labeled comments. This dataset had approximately half positive/neutral comments and half abusive comments. [11] Mosquera, et al. were able to get access to a dataset of about 700 SMS texts from a large telecommunications company [12]. Kaggle released a dataset

for one of their competitions titled "Detecting Insults in Social Commentary" [13]. And in conjunction with Google's Perspective API release in February 2017, the Wikipedia Detox project released a publicly available dataset containing comments scraped from Wikipedia's talk page diffs. The dataset also included Crowdflower-sourced labels for comments based on aggression, toxicity, and personal attacks.

We encountered several papers where researchers in this space generated their own datasets; however, manual generation tended to limit the size of the data. By far the largest public dataset we encountered, the Wikipedia dataset holds over 150,000 examples. Thus, we decided to use the Wikipedia data moving forward.

### 2.3 "Ex Machina: Personal Attacks Seen at Scale"

Using the personal attack data from Wikipedia's Detox project, Google Jigsaw published a paper on using machine learning to automatically detect abusive comments [14]. Titled "Ex Machina: Personal Attacks Seen at Scale," the paper detailed some of the algorithms behind Google's recently released Perspective API, an API aimed at "hosting better conversations." In their paper, the researchers tested logistic regression and multi-layer perceptrons for their model, word and character n-grams for their features, and one-hot vectors and empirical distributions for their labels. They found that a multi-layer perceptrons (MLP) model with character n-grams and empirical distribution labels had the highest standard 2-class area under the receiver operating characteristic curve (AUC) score and an F1 score of 0.63.

The rest of the paper focused primarily on comparing their best model against a human baseline and on understanding how attacks correlate with commenter identity and other commenting patterns. Their next steps included applying their methodology to other comment characteristics, as well as testing more complicated models such as an LSTM. Overall, Wulczyn, et al.'s work provided us with results against which we could test our models and gave us a better intuition for what features might contribute to a comment's abusiveness.

## 3 Approach

### 3.1 Feature embeddings and labels

To represent each sentence as a set of features, we used pre-trained GLoVe word embeddings. Trained on Wikipedia data, GLoVe representations combine the global matrix factorization and local context window models for representing words in vector space. The advantages of using GLoVe include high-dimensional feature vectors that better capture sentence substructure data and that use global co-occurrence counts, which are not limited by a classifier's window size. [15]

While we chose to use word representations as our standard feature embedding type, we also experimented with character-level embeddings for one of our CNNs since previous studies (Kim; Wulczyn, et al.) have found character embeddings to be more successful in classification problems involving online commenting forums [10, 14]. We combined character embeddings with the CNN model rather than the LSTM since CNNs train quickly enough for character-level embeddings to still run efficiently.

For our labels, we used one-hot vectors since the ideal output of our system would be a recommendation on whether or not to moderate a comment, rather than an empirically distributed score. For more details on how the gold-standard labels for each comment were calculated, see section 4.1 on data processing.

### 3.2 Baselines

We used two baselines in order to get a better understanding of the problem from both a general machine-learning perspective and a deep-learning perspective. We used linear regression with a least squares loss function in order to provide the general machine-learning baseline. We show the least squares loss function here:

$$J(f(x); y)_{\text{least squares}} = \Sigma_i (y_i - f(x_i))^2 \tag{1}$$

We also ran a feed forward neural network classifier to provide a baseline from a deep learning perspective. TensorFlow has a DNN classifier built into its libraries, and its default loss function is logistic loss:

$$J(f(x); y)_{\text{logistic}} = log[1 + exp(-yf(x))] \tag{2}$$

### 3.3 Primary models

For our primary models, we used a recurrent neural network (RNN) with a long short-term memory (LSTM) cell and a convolutional neural network (CNN). We tested both word-level and character-level embeddings with the CNN while we only tested word-level embeddings on the LSTM for efficiency reasons. We chose our models since both have been known to provide state-of-the-art performance for sentiment analysis.

#### 3.3.1 Convolutional Neural Networks

CNNs are known to perform well on data with high locality, when words or features care more about the features surrounding them. For our classification problem in particular, we expect high locality in comments given their short length and their tendency to focus on a specific topic, response, or idea. CNNs also have the added benefit of being more computationally efficient than LSTMs, enabling us to feasibly test character-level embeddings in addition to word embeddings.

For our CNN, we created a word model function that took in the processed sentence vectors (from either our trained character embeddings or the pre-trained word embeddings), the corresponding output labels, and the mode of our model (i.e. training mode, evaluation mode, or inference mode). Our CNN iterated over a given list of kernel sizes and concatenated their outputs from two layers: a convolutional layer that computed 20 features using a kernel-sized filter with ReLU activation and a pooling layer of size 2. After flattening the tensor into 2 dimensions, the CNN then fed into a feed forward neural network composed of two layers. The feed forward neural network used a dropout rate of 0.4 between its layers to prevent overfitting to the dataset.

We defined our CNN loss function as softmax with cross-entropy loss. These equations are defined as follows:

$$softmax(x)_i = \frac{e^{x_i}}{\Sigma_j e^{x_j}} \tag{3}$$

$$CE(f(x); y) = -\Sigma_i y_i log(f(x_i)) \tag{4}$$

#### 3.3.2 Recurrent Neural Network with Long Short-Term Memory cell

LSTMs in particular outperform RNNs with GRU cells on problems where understanding a broader, long-term context is important, and the LSTM cell also prevents the problem of a vanishing gradient. LSTMs are commonly used in sentiment classification because of their ability to use longer-term contexts. In addition, since LSTMs had previously worked well on Tweets [6] and the length of comments in our dataset were comparable to the length of Tweets, we hypothesized that LSTMs would be similarly useful in this problem area. The LSTM cell can be defined mathematically using the following equations:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}_{t-1}) \tag{5}$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \tag{6}$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \tag{7}$$

$$\bar{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \tag{8}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \bar{c}_t \tag{9}$$

$$h_t = o_t \circ \tanh(c_t) \tag{10}$$

We used a bidirectional RNN model with 30 hidden layers for each LSTM cell. The resulting hidden layers were concatenated and passed through a feed forward neural network of two layers.

Table 1: Baseline results

| Dataset | F1 Score | Accuracy | AUC |
|---|---|---|---|
| Linear regression | 0.45 | 0.91 | 0.89 |
| Deep neural network | 0.54 | 0.91 | 0.91 |

Dropout was applied at a rate of 0.2. We used softmax-cross entropy and optimized using the Adam optimizer. We tested a variety of sentence lengths and learning rates, as described in the next section.

# 4 Experiments

## 4.1 Data processing

The Wikipedia Detox project provided several different datasets of tagged Wikipedia comments. We chose to use the hand-tagged datasets on comments as personal attacks in order to more easily compare our results to Wulczyn, et al. [12]. The dataset included 115,846 comments, which were around 88 percent unflagged. We divided the dataset into train, dev, and test subsets using a 3:1:1 split.

Up to 10 crowdworkers had labeled each comment as a personal attack categorized into four different categories, including quoting attacks or recipient attacks. But for the purposes of our study, we focused only on whether or not the comment had been categorized as an attack generally. We used the majority attack rating among crowdworkers in order to create a binary classification of flagged or unflagged for each comment.

## 4.2 Models

We tested three primary models for comment classification:

- An RNN with an LSTM cell and word embeddings
- A CNN with word embeddings
- A CNN with character embeddings

We ran each of our three models over the attack dataset and found that the CNN with character embeddings performed best while both models with word embeddings performed about the same.

### 4.2.1 Baselines

Our baseline models obtained the F1 scores, accuracies, and AUC scores displayed in Table 1. The DNN performed better than regular linear regression, with F1 scores of 0.54 and 0.45 respectively. These results demonstrated the potential for a deep learning algorithm to outperform a regular machine learning model. However, without tuning, both baseline algorithms had lower F1 scores and AUC scores than the highest-performing MLP tested in Wulczyn, et. al. [14]

### 4.2.2 CNNs and LSTM

Out of the two CNNs, the CNN with character embeddings took a larger number of steps to train, requiring 50,000 steps to reach an F1 score of 0.73. The CNN with word embeddings reached a 0.70 F1 score after only 5,000 steps, and performance did not improve with further training. For both models, the optimal learning rate fell around 0.1.

The LSTM performed about the same as the CNN with word embeddings, with an F1 score of 0.69. It required 5000 steps to train; however, it also took a significantly longer amount of time to run, approximately five to ten times longer than CNNs. Its optimal learning rate is 0.0001.

Table 2 shows all three models' F1 scores and accuracies when trained for 5,000 and 50,000 steps. Tables 3 and 4 show tuning for each model based on their learning rates and maximum sample

Table 2: Convolutional neural network results

| Model | F1 Score (5,000 steps) | F1 Score (50,000 steps) | Accuracy (5,000 steps) | Accuracy (50,000 steps) |
|---|---|---|---|---|
| CNN with word embeddings | 0.70 | 0.69 | 0.93 | 0.93 |
| CNN with character embeddings | 0.57 | 0.73 | 0.92 | 0.94 |
| LSTM with word embeddings | 0.70 | 0.68 | 0.94 | 0.93 |
| MLP with character embeddings [14] | 0.63 | 0.63 | N/A | N/A |

Table 3: Learning rate tuning: Accuracy

| Model | 1e-6 | 1e-5 | 1e-4 | 1e-3 | 1e-2 | 1e-1 | 1 |
|---|---|---|---|---|---|---|---|
| CNN with word embeddings | | 0.88 | 0.88 | 0.88 | 0.93 | 0.94 | 0.88 |
| CNN with character embeddings | | 0.88 | 0.88 | 0.89 | 0.93 | 0.93 | |
| LSTM with word embeddings | 0.89 | 0.92 | 0.94 | 0.93 | 0.93 | 0.87 | |

lengths. The learning rate was an important parameter in helping the algorithm learn, and learning rates that were too high or low stopped the algorithm from working. We knew intuitively that some of the learning rates on the lower or higher end of the spectrum would not perform well due to the learning rates we already ran, so we didn't run 1e-6 for CNNs nor did we run 1 with CNN with character embeddings or LSTM with word embeddings. We also investigated varying the maximum sentence lengths, since sentences longer than the maximum sentence length were cut off. While there was a small effect on performance, changing the maximum sentence length was not a significant factor in overall performance. This result can be explained by the fact that comments are generally quite short, and thus the sentence length of the comments are not super variable throughout the dataset.

## 5 Conclusion

After testing both CNNs and an LSTM on our Wikipedia comments, we were able to reach accuracies of up to 0.94 on classifying the test comments as flagged or unflagged. Given this result, it's reasonable to conclude that deep learning can be a useful tool in comment moderation and creating safer online communities.

In particular, we found that character embeddings performed better than our word embeddings for the CNN. It's also worth noting that although CNNs with character embeddings require more training iterations than CNNs or LSTMs with word embeddings, their computational efficiency gives them a benefit over LSTMs, despite similar F1 performance. Therefore, in a practical application to automatically moderate comments, we suggest using a CNN with character embeddings.

### 5.1 Future Work

Based on our background research, we believe that the success of character-level embeddings is due to the nature of online commenting, as users are not held to strict standards of proper English. To confirm the qualitative features behind the data, future work would include looking into the learned weight vectors and learned character embeddings. Because word embeddings generally perform better on natural language problems, using word embeddings such as the GLoVe vectors have become the industry standard. However, since character-level embeddings performed better in this particular classification problem, we would need to look more in-depth at the learned vectors to determine

Table 4: Maximum sample length tuning: Accuracy

| Model | 20 | 50 | 100 | 250 | 500 |
|---|---|---|---|---|---|
| CNN with word embeddings | 0.93 | 0.94 | 0.94 | 0.94 | |
| CNN with character embeddings | | 0.90 | 0.92 | | 0.93 |
| LSTM with word embeddings | 0.92 | 0.93 | 0.93 | 0.93 | |

Table 5: Training Loss as a function of number of iterations

| Model | 1000 | 2000 | 5000 | 10000 | 25000 |
|---|---|---|---|---|---|
| CNN with word embeddings | 0.29 | 0.25 | 0.21 | 0.16 | 0.16 |
| CNN with character embeddings | 0.34 | 0.34 | 0.33 | 0.27 | 0.27 |
| LSTM with word embeddings | 0.21 | 0.22 | 0.18 | 0.16 | |

what specific factors caused the character embeddings to succeed. Given their current success, we would also want to test character embeddings on other types of machine learning models.

Given more time, we would also plan to further tune our models based on the following additional hyperparameters:

- CNN: kernel size widths and numbers of filters
- LSTM: weights of the forget gate

Given the amount of tuning required for the above models, we also believe that it would be worth considering how to decrease the size of the optimal model. One potential classifier could take the output of the character-level CNN and use it as the input into the LSTM. The resulting model would have fewer hyperparameters and allow for easier tuning. [14]

In order to more comprehensively evaluate our models' effectiveness, our next steps would also include comparing our models to human performance. The current attack scores are generated by averaging crowdsourced comment labels, and these labels could also be used to determine human classification accuracy.

Lastly, additional features for classification could be added to our models in order to improve their performance. Politeness and commenter identity features (anonymity, past posts, etc.) are just two potential options. Furthermore, the Wikipedia data could be used to test the effectiveness of multi-class or empirically distributed labels over one-hot labels, and the aggression and toxicity datasets could provide more insight into what other features could be used to best moderate abusive comments.

While there are still many angles that can be done to further our research in this space, the goals for our next steps can be encompassed in one larger idea: attempting to learn and exploit what makes this problem space unique in order to create a more accurate model.

## 6 Contributions

Each part of the project had contributions from multiple team members. Kylie began work on the 'gen_data_files.py' script, which preprocesses and splits the dataset into train, development, and test sets, and it was completed later by Theodora. From the vocabulary that was generated, Max wrote indextoembeddings.py to construct the trimmed embeddings file. Kylie began implementation of the CNN with Max, and it was later debugged by Max and Theodora. Max began implementation of the RNN and later received help from Theodora for debugging. Max and Theodora implemented the baseline models.

Similarly, each member contributed to the final poster and project. Kylie drafted the outline and completed parts of the Introduction, Approach, and Experiments sections of the paper and the corresponding sections of the poster. Theodora conducted a literature review and wrote the Related Work section in addition to writing parts of the Introduction, Approach, and Conclusion. Max wrote the parts of the Experiments section where his project work was most involved and created tables and graphs for the paper and poster presentation. We feel that this project was truly a team effort and learned a lot from creating it!

## 7 References


[1] Duggan, Maeve. "5 Facts about Online Harassment." Pew Research Center. Pew, 30 Oct. 2014. Web. 20 Mar. 2017.

[2] Luxton, David D., Jennifer D. June, and Jonathan M. Fairall. "Social media and suicide: a public health perspective." American journal of public health 102.S2 (2012): S195-S200.

[3] Yin, Dawei, et al. "Detection of harassment on web 2.0." Proceedings of the Content Analysis in the WEB 2 (2009): 1-7.

[4] Zhang, Xingxing, and Mirella Lapata. "Chinese Poetry Generation with Recurrent Neural Networks." EMNLP. 2014.

[5] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.

[6] Wang, Xin, et al. "Predicting Polarities of Tweets by Composing Word Embeddings with Long Short-Term Memory." ACL (1). 2015.

[7] Huang, Minlie, Yujie Cao, and Chao Dong. "Modeling rich contexts for sentiment classification with lstm." arXiv preprint arXiv:1605.01478 (2016).

[8] Dos Santos, Ccero Nogueira, and Maira Gatti. "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." COLING. 2014.

[9] Mehdad, Yashar, and Joel Tetreault. "Do Characters Abuse More Than Words?." Proceedings of the SIGdial 2016 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue. 2016.

[10] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[11] Maus, Adam. "SVM approach to forum and comment moderation." Class Projects for CS (2009).

[12] Mosquera, Alejandro, et al. "On Detecting Messaging Abuse in Short Text Messages using Linguistic and Behavioral patterns." arXiv preprint arXiv:1408.3934 (2014).

[13] Goyal, Priya, and Gaganpreet Singh Kalra. "Peer-to-peer insult detection in online communities." IITK, unpublished (2013).

[14] Wulczyn, Ellery, Nithum Thain, and Lucas Dixon. "Ex Machina: Personal Attacks Seen at Scale." arXiv preprint arXiv:1610.08914 (2016).

[15] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." EMNLP. Vol. 14. 2014.