Flame Wars: Automatic Insult Detection

Sasha Sax

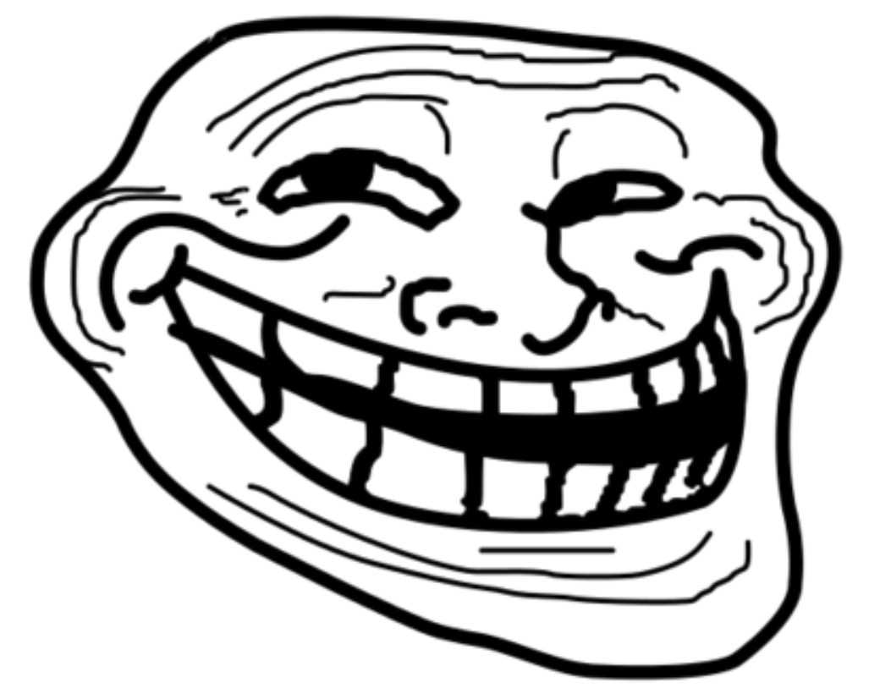
Stanford University, Department of Computer Science

Contact Information:

Department of Computer Science

Stanford University

Email: `asax@stanford.edu`



Abstract

Abusive comments are increasingly drowning out constructive ones, and websites are reacting by shutting down comment sections. Human moderation is slow and expensive, so an algorithmic solution would be preferable. In this project I explore the problem, existing literature, and several models to automatically flag abusive comments.

Introduction

Online forums seem to be increasingly toxic, and this has had a chilling effect on internet forums. In the last year alone, Bloomberg, The Verge, The Daily Beast, and Viceś Motherboard all shut down their comment sections due to concerns about the tone and quality of comments. Abusive comments seem to be becoming an existential threat to online discourse.

Current solutions usually employ human moderators. The moderators then either must approve each comment before it appears, or review a comment when it has been flagged by a user. Both of these techniques have significant drawbacks: approving every comment is slow and discourages discussion, but waiting until a comment has been flagged means that, in some sense, the damage has been done. Both of these techniques are also financially expensive as they require a real human to do the job.

An ideal solution would be a completely autonomous system to prevent abusive comments from ever being posted, but due to the imperfections of current systems it is more likely that flagged messages will be sent to human moderators. Prior approaches for abusive comment detection usually include basic machine learning approaches such as SVM [7] [3], Naive Bayes [2] [6], random forests [2], or logistic regression [4] over a bag-of-ngrams [3]. Newer approaches have tried incorporating word vectors and Paragraph Vector [5] - but it seems that no one has yet tried a recurrent neural approach.

Problem Description

The term abusive comment actually describes a wide variety of comments. Hate speech, insults, profanity, threats, and various ethnic, racial, or homophobic slurs can all be considered abusive, and these categories are not mutually exclusive.

Consequently, it is nontrivial to label comments. Most prior literature has restricted attention to specific categories above in order to make the task more well-defined. I will focus only on insult detection because there is a publicly available dataset for this task, and I couldnt find any others. Insult detection can also be difficult because two sentences may only be insults when combined - so there are cross-sentence dependencies (*youre an academic. All academics are gits.*).

Data

I used Impermiums Detecting Insults in Social Commentary from the 2012 Kaggle competition [1]. The dataset has fundamental problems (discussed later), but it seems to be the only one available until the Yahoo one comes out. For this reason, Nobata et. al [5] cite their 2.5 million comment dataset as the major contribution of their paper. I would have liked to use that dataset for this project, but it has been not yet been made available.

Dataset	% Insults	# Comments
Train	28.24%	4921
Dev	49.46%	750
Test	47.23%	1197

Table 1: Breakdown of datasets

There are several problems with the data. The simplest is that the training set has a much lower percentage of insults than does the validation or test set, so the test isnt representative of the training data. In addition, I randomly sampled 50 comments from the test set, and 50 comments from the train set, and I disagreed with 4% of the train labels and 16% of the test labels. I found that the test set had many borderline cases. In addition, it is difficult to judge whether a comment directly insults someone in the conversation - the criterion Impermium used.

Approach

Online comments are noisy, and sometimes ungrammatical. (E.g. *And you r sucking my dope dyck to hard. Take your fake ass Chief Keef face up the block.*) These traits sometimes, but dont always, indicate insults. It is also simple to avoid word-level blacklists (e.g. *asshole* vs. *a$$hole*). Words also sometimes have repeated characters (*fuuuuuuuuck*) and its often possible to shrink these to the original lemma (*fuck*) at the cost of losing information.

Because these relatively simple character-level changes create large word-level difficulties, I approached the problem with a char-LSTM to classify the comment. I used l2, dropout, character dropout, and noninsult downsampling to regularize the model. I would like to try multiple layers and bidirectionality, but due to the dearth of comments these models did not perform well. I decided, given the limited time for my project, that it was not worth it to tune these more interesting models.

I also compared this model to a bigrams+unigrams logistic regression (with stemming, tokenizing) and a vector-sum bag-of-words approach. My logistic regression would have placed 10/50 in the Kaggle competition, and so I believe it provides a strong benchmark.

Example of Char-LSTM

This is an example of the char-LSTMs ability to deal with noise and out-of-vocabulary problems, while a bigram/unigram model cannot.

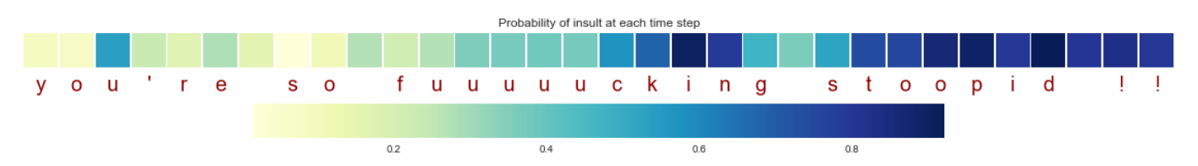


Figure 1: Probability of insult at each timestep

The bigram model classifies this as a non-insult (Pr(insult) = 0.237)) while the char-LSTM correctly labels it as an insult (Pr(insult) = 0.999)).

Results

I used both F1 and AUC to evaluate the the different models. F1 is calculated as follows:

$$F1 = \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

AUC is closely related to the Mann-Whitney U score, and can be computed via numerical integration of the fitted ROC curve.

Evaluation consisted of computing these two metrics on the predictions of the models on the test set, which is a set of 1200 comments, roughly half insults. The results are as follows:

Model	F1 Score	AUC
Uni	0.663	0.794
Uni + Bigrams	0.640	0.794
Stemming + Unigrams	0.671	0.796
Stemming + TFIDF + Unigrams	0.603	0.767
Stemming + Uni + Bigrams	0.665	**0.816**
Stem + 1,2grams SVM	0.580	0.736
GloVe Vector-Avg 100d	0.507	0.694
GloVe Vector-Avg 200d	0.568	0.733
GloVe Vector-Avg 300d	0.615	0.743
charLSTM 350d	**0.718**	0.792
bd charLSTM 350d	0.702	0.756
2-layer charLSTM 350d	0.667	0.510

Table 2: Preliminary results

Analysis

Its interesting to note where the model succeeds and fails, especially in relation to the word-level models. In short, the character model generalizes better to out-of-vocabulary and unseen data, but there are better architectures than a simple LSTM.

Error Analysis

Negation *(Youre not a stupid git)* It does not do well on classically hard examples like negation. This is similar to other (easy) tasks such as sentiment analysis. A tree structure or dynamic memory network might be able to capture this.

Non-insulting uses of words *("bitch" is just another word)* The model may not have enough data or layers to capture contexts like this.

Dependencies on earlier parts of the comment *(a female dog is sometimes called a "bitch")* This is a tough concept for a unidirectional RNN to capture. A bidirectional RNN may be able to capture this, but the dataset is quite small for a bidirectional RNN

Success Analysis

Long distance dependencies *(you're dumb, but these characters should distract you from that fact!)* Somewhat surprisingly, the model succeeds even when the insult comes at the beginning of the sentence. The filler words throw off the baseline models, but the char-RNN gets these right. This even works in the case where the filler words are a non-insult (the model does not consider , but these characters should distract you from that fact! an insult).

Conclusions

The char-LSTM does well on some examples that prior models struggled with due to their bag-of-words assumption, inability to capture higher-than-word-level concepts, and inability to handle OOV words. The char-LSTM handles some of these cases, and outperforms the baseline models. The char-LSTM is also end-to-end, and has no a-priori linguistic knowledge. That it outperforms the baseline is a testament to the power of recurrent architectures in NLP tasks.

In the insult detection task, the char-LSTM still suffers from its typical drawbacks. It does not handle negation well, nor does it understand mitigating contexts and dependencies on earlier parts of the comment. As discussed above, more complex models may be able to handle these cases, and I suspect that they would perform better than the single directional LSTM. Unfortunately, these models benefit from large amounts of data, and this one is quite small.

In conclusion, neural architectures are a promising approach for insult detection, and I suspect also for abusive comment detection. It appears straightforward to create a high specificity filter that can flag comments for moderation. In this regard, character level architectures are far preferable to word-level ones because they are more difficult to circumvent via character replacement.

In the next week Yahoo is releasing a 2.5 million comment dataset of 25% abusive comments. It would be interesting to see how both the current architectures (from this paper), and more complex ones fare on the data. I believe the gap between the baseline and the char-LSTM will increase as the amount of data increases. I would be interested to see which architectures do well, and also which domains (comments in Yahoo! Finance vs Yahoo! Sports) are more amenable to automatic detection methods. It would also be interesting to see how well word-level models do, especially when combined with backoff.

References

[1] Detecting insults in social commentary, data, 2012.

[2] Priya Goyal and Gaganpreet Singh Kalra. Peer-to-peer insult detection in online communities. *IITK, unpublished*, 2013.

[3] Tatsuya Ishisaka and Kazuhide Yamamoto. Detecting nasty comments from bbs posts. In *PACLIC*, pages 645–652, 2010.

[4] Andreas Mueller. Recap of my first kaggle competition: Detecting insults in social commentary, 09.

[5] Chikashi Nobata, Joel R. Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 145–153, 2016.

[6] Amir H. Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. *Advances in Artificial Intelligence: 23rd Canadian Conference on Artificial Intelligence, Canadian AI 2010, Ottawa, Canada, May 31 – June 2, 2010. Proceedings*, chapter Offensive Language Detection Using Multi-level Classification, pages 16–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics, 2012.