Toxic Comment detection with bi-directional LSTM

Fei Liu*
Machine Learning Engineer
Pinterest
fliu5@stanford.edu

Xiaoyan Wu
Department of Computer Science
Stanford University
xw1@stanford.edu

Abstract

This article elaborates several approaches built on bi-directional LSTM to tackle Toxic Comment Classification problem. Strategies tried include pre-trained word embeddings, data augmentation, attentions, char-CNN model, neural ensemble of models with different configs, parameter regularization, and more ReLU layers on top of recurrent states. We found that bi-LSTM with 2 ReLU layers is the best performing model, which achieves an average AUC of 0.9782 over the labels. This model significantly out-performs the baseline NBSVM model which achieves an average AUC of 0.9533 over the labels.

1 Kaggle Challenge

The kaggle competition[1] for toxic comment classification is an ongoing competition about detecting different types of toxic comments by classifying them into one or more labels from 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', and 'identity_hate'.

The challenge is particularly interesting because of the heated discussion that toxic contents online have influenced the overall health of the society. It is also interesting in the sense that the service providers are finally leveraging deep learning to supervise their service in a scalable way.

The baseline is an NBSVM model, which achieves an average AUC score of 0.9533. Other methods attempted include attention, character-level CNN layer beneath LSTM, neural ensemble, and data augmentation. The best performing model is Bi-directional LSTM with pre-trained embeddings, combined with two ReLU layers built on top of the recurrent states of LSTM. A small L2 regularizer on the model parameters is also applied to prevent overfitting. It achieves best result with data generated from data augmentation for "identity_hate" labeled data. Our best score is 0.9782, which is a big improvement from the baseline above.

1.1 Dataset and example

The competition data is consist of approximately 160,000 entries of training data and the comments are of various length raw text. We found approximately 50,000 unique words used. The data is the standard dataset in the competition. The goal is to assign each comment a multi-category label that can be one or more of the following categories 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', and 'identity_hate'.

For example, the setence that contains the lines "i CAN STILL POST WITH THIS COMPUTER...I SAID BLOCK ME, COME THE FUCK DOWN HERE AND ARREST ME...SAN DIEGO CALIFORNIA, CHULA VISTA, FUCKING GET YOUR INFORMATION RIGHT FAGGOT SHITHEAD!!" repeated hundreds of times was labled as 'toxic', 'obscene', and 'insult'.

*My mother, Qianjun, always says that Fei, you go and you do what you want to do. Life motto starts among her lines. Thanks!

[1]Kaggle online competition: https://www.kaggle.com

Table 1: Data Statistics

Label Name	Count of Lines
toxic	15294
severe_toxic	1595
obscene	8449
threat	478
insult	7877
identify_hate	1405

1.2 Data statistics

Data stats are presented in Table 1. The "threat" and "identity_hate" labels are weak spots because their numbers of comments in the raw data are the fewest. We train our model on 90% of the original training data, and validate on the remainder, so we can collect AUC scores for each label. From the AUC curves, we identify the tags that are not generalizing well, and try to solve these bottlenecks.

2 Preprocessing

The max length of the comment can be 5000 but they are mostly repetation of the same string due to some really angry comments. The meaningful comments rarely exceed 200 words. However, many comments are very short, so we have to pad the sequence and make the input of the same length. Then for the words with pre-trained embeddings, we are having size of dimensions up to 300. For the ones we tried without pre-trained embeddings, we have 35 dimensions efficient update.

3 Benchmark

The benchmark is implemented by using Naive Bayesian log-count ratios as features into Support Vector Machine [2] according to the paper[5]. NBSVM is an improvement on SVM that is better and more stable than SVM and Naive Bayes. The benchmark result is summarized in Table 2.

Table 2: NBSVM benchmark

Label Name	AUC Score
toxic	0.9324
severe_toxic	0.9736
obscene	0.9544
threat	0.9623
insult	0.9482
identify_hate	0.9490

This model achieves an average AUC score of 0.9533, and the plot of AUC curves is shown in the following figure. From the plot, we can see that "toxic" and "identity hate" are the lower curves. We identify these two as bottlenecks, and then implement different methods to improve on their accuracy.

[2]NBSVM: https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline

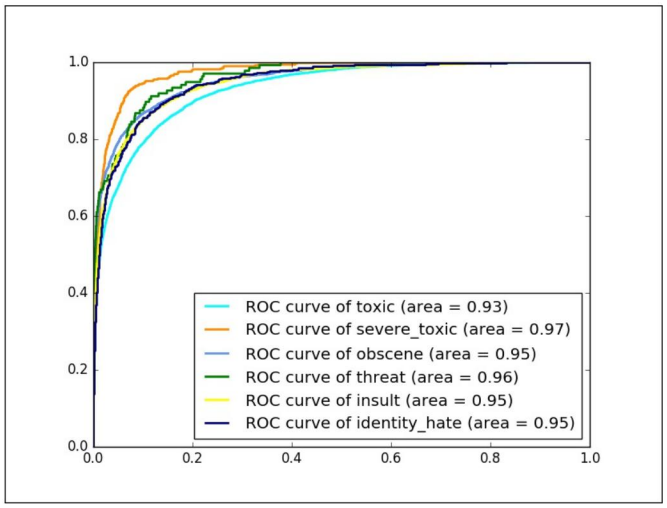

Figure 1: AUC plot for NBSVM

4 Motivations for implementation

The data is very unbalanced and small in size, and the baseline is strong. We tried even-sampling so that the dataset is balanced, but the performance decreased. We want to use Augmentation to correct the behavior and use embeddings to make more sense of the words than the vocabulary in the dataset.

We also want to utilize the LSTM internal states to extract information out of the sequence, so we are building ReLU layer on top. Regularizer and neural ensembling are the other attempts to increase performance. We were successful to beat the benchmark by combining embeddings and more ReLU layers on recurrent LSTM states. We also use regularizer to stabilize the performance.

We were then recommended to use char-CNN and attention, so these are also implemented to compare the performance of different models.

5 Model Implementation

The bidirectional LSTM is the basis of our model, and all other attempts are built on top of this LSTM. We break the implementation process into modules, including preprocessing, training, prediciton, evaluation, and submission. Preprocessing handles the augmentation and partition of the data. Training is about isolating each model into its own module and load only what is needed, plug in and run. Prediciton step applies the trained model to generate predictions. The local evaluation with a split of data is important in understanding the source of error and identify the bottlenecks. Finally, we generate submission files and submit to Kaggle competition. Figure 2 shows a system map of our implementation.

5.1 Measure of good of fitting

Since groundtruth labels are only provided for the training set, we need to use 10% of training data as validation set in order to have a measure of how good each label is predicted. We calculate AUC scores of each label on this set to use as a local approximation to the competition score. After the local training and testing, we submit to Kaggle to get an evaluation on the hidden test data.

5.2 Baseline LSTM Implementation

The baseline Bi-LSTM is trained without pretained word embeddings. In order to have enough meaningful updates, the dimension of the word embeddings needs to be small. In our case, the size

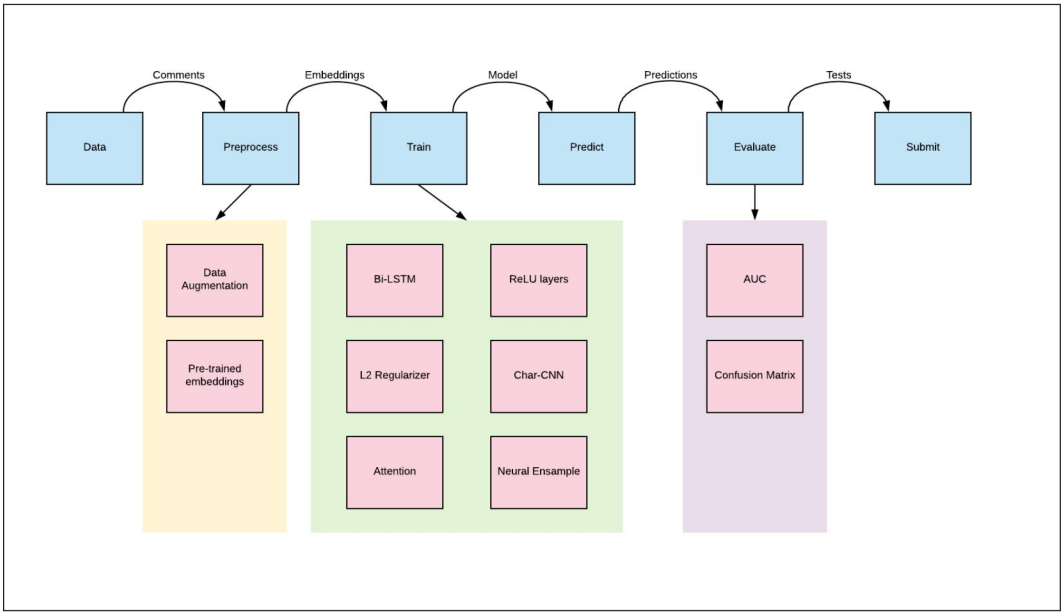

Figure 2: System Pipeline

is 35. We are generating the dense layer right out of the recurrent states of LSTM of length 100. The result of the baseline is summarized in Table 3, and the average AUC score is 0.7612. The performance is not as good as NBSVM in the benchmark.

Table 3: bi-LSTM baseline + 35 embedding (not pre-trained)

Label Name	**AUC Score**
toxic	0.7329
severe_toxic	0.8053
obscene	0.7500
threat	0.7620
insult	0.7628
identify_hate	0.7542

5.3 More ReLU layers

We realize that a dense layer right on top of LSTM is linear in nature. We want to capture as much non-linear complexity as possible. So, we attempted ReLU to capture the non-linear aspects of the LSTM recurrent states. We tried 1, 2, and 3 layers and it turned out that 2 layers produce good result for dimension 100 and dense dimension 30. This alone is not improving the result because we do not have good enough features coming in. We will have to combine this with embeddings to produce good result.

5.4 Pre-trained embeddings

In order to solve the problem that our data is small, we use 300 dimension GloVe vectors trained on 6B tokens[3] as pre-trained embeddings into our model. This approach is combined with the ReLU layer approach in the previous subsection and produces the result in Table 4. The Kaggle score for this model is 0.9728 and it is close to our best model.

[3]Pretrained embeddings: https://nlp.stanford.edu/projects/glove/

Table 4: bi-LSTM baseline + 300D GloVe + 2 ReLU layers

Label Name	AUC Score
toxic	0.9876
severe_toxic	0.9911
obscene	0.9921
threat	0.9812
insult	0.9881
identify_hate	0.9845

5.5 L2 regularizer

We found that our training score is much higher than the Kaggle evaluation score and we are overfitting, so we applied L2 regularizer. The hard thing is how small we should set this value to be and we were not experienced enough to utilize this to close the gap between training and testing accuracy. The smallest value we tried was 0.01, which is giving us 0.9648 on Kaggle. Trying smaller value regularizers could be our future work if we have more time.

5.6 Char-CNN input layer

We were advised that Character-level Convolutional Neural Networks can help with words that are not appearing in the dictionary. We then tried this approach by building on top of an online implementation of Char-CNN [4], but it was slow to run and consumed two much memory. We only ran it with LSTM of length 50 and one convolutional layer. It was giving us a AUC of near 0.4991 on Kaggle. We decided not to explore further on this approach.

5.7 Attention mechanism

The attention mechanism is built on top of LSTM recurrent state outputs as well. We want to know which of the states should be paid more attention. We implemented this based on an online approach[5]. It uses linear weighting on top of the recurrent states, so in terms of the ability to capture non-linear behaviors, it may not exceed non-linear methods such as multi-layer ReLU. The AUC score on Kaggle is 0.7505, so we did not explore further.

5.8 Neural ensemble

We have trained multiple LSTM models with different configurations, so we want to see if combining them together will help. Therefore, we combined 6 LSTMs with averaging and a one-layer feed-forward network, but the result is as mediocre as any of these models. It is the end to end process in the model that does the trick, and ensembling here is not as helpful as in the traditional Machine Learning models such as Random Forest Trees and Gradient Boosted Trees.

5.9 Data augmentation

Since ”threat” and ”identity_hate” have much less data than other labels, we use data augmentation [1] to generate more data for these two labels. We first select the comments that are labeled as ”threat”, extract word entities from these comments by using Stanford Named Entity Recognizer (NER)[3]. This will give us a list of ”PERSON”, ”ORGANIZATION” or ”LOCATION” words. Then we randomly substitute some of these words back into the original sentence. For example, a sentence that contains ”New York City” could be changed to ”Los Angelos”, and the comment may still be labeled as ”threat”. In this way, each comment could be replicated multiple times and we could have more training data on the weaker labels. One assumption is that many of these words

[4]Char level CNN: https://github.com/blues-lin/Char-level-CNN-for-Text-Classification-in-Keras

[5]Attention model https://github.com/richliao/textClassifier

do not affect toxic comment classification, so generating more data in this way could let the model place less weight on the name entities but more on the context.

We found that using PERSON name entities on "identity_hate" label is helpful. However, the "threat" label is not that sensitive to the augmentation. The reason may be that some of the person or location names have political meanings, and are useful in identifying "threat", so substituting these words will damage the performance. Therefore, our best model only uses PERSON name entity augmentation on "identity_hate" label and achieves 0.55% increase in Kaggle AUC score (defeated 385 teams with this strike).

6 Model of choice and Error Analysis

Our best model uses 2 ReLU layer on Bi-LSTM with small (0.01) L2 regularizer, GloVe pretrained embeddings plus augmentation on PERSON name entity on "identity_hate" labeled data. And the highest Kaggle AUC score with this model is 0.9782. The AUC's for validation split locally is in Table 5.

Table 5: bi-LSTM + 300D GloVe + 2 ReLU layers + Augmentation on id_hate

Label Name	**AUC Score**
toxic	0.9918
severe_toxic	0.9907
obscene	0.9946
threat	0.9609
insult	0.9901
identify_hate	0.9712

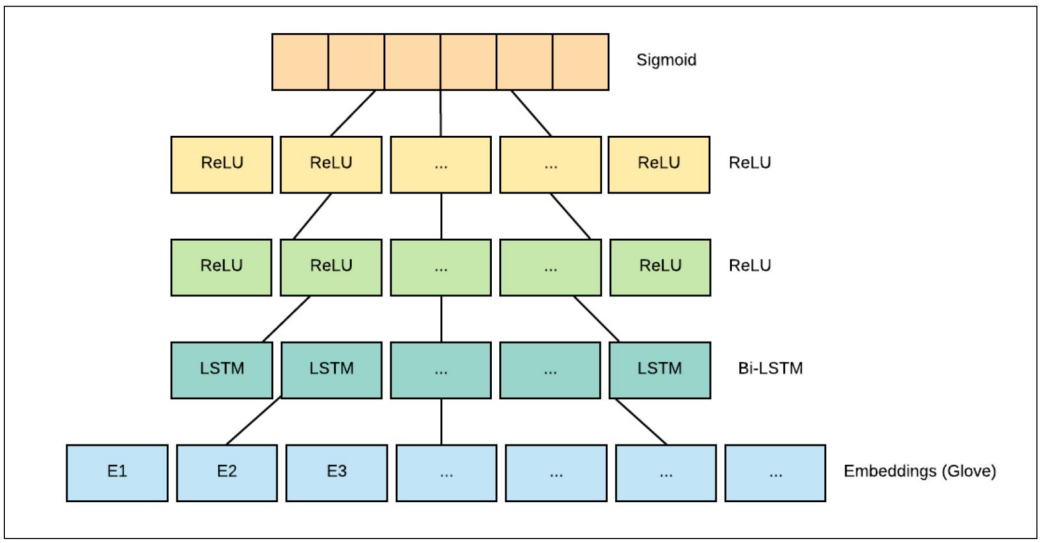

Figure 3: Best model flow

We use this model to make predictions on the training set to see what it does well and what goes wrong. From the AUC curves we can see that "obscene" is the label that has the highest accuracy, and "threat" is not performing well.

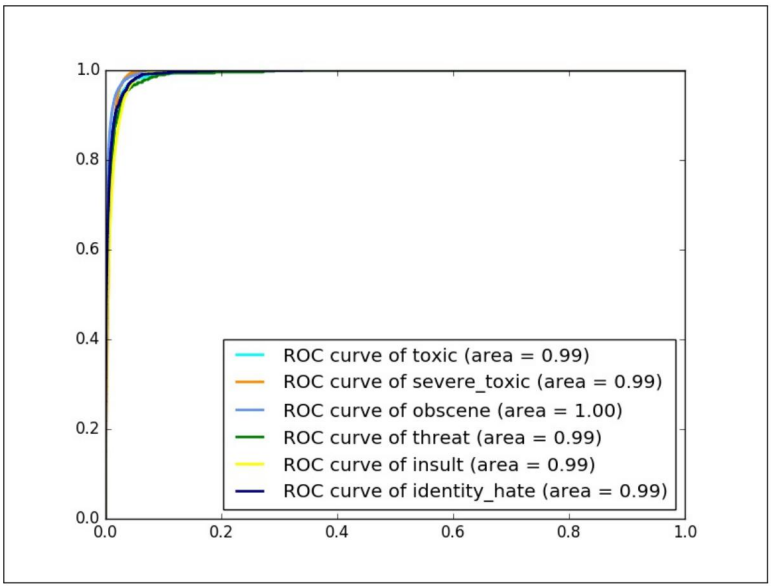

Figure 4: Caption

By comparing the true labels and our predictions on "threat" label, we found specific examples that predict wrong. In the following example (figure 5), we see that the comment contains lots of special characters, and these words may not have pretrained embeddings. Our model learns very little from these words, which results in wrong prediction.

Hello, you turd.

Firśt, I aṃ ǵoinǵ ţo ţie ŷou uṗ and keeṗ ŷou çonśçiouś durinǵ ţhe followînǵ ṗroçeśś ţhrough ţhe uśe of aṃṃonia. Nexţ, I ŵill ṃuţilaţe ŷour ǵeniţalś and forçe ŷou ţo eaţ ţheṃ. Then I ŵill çuţ off biţś of ŷour śkin and ṃake a ǵlove ouţ of ţhe śkin from ŷour hand. I ŵill ţhen çuţ oṗen ŷour ǵuţ and ṗull ouţ ŷour çolon. I ŵill uśe ŷour çolon for ṃaśţurbaţion, and ŵhen I am done I ŵill śhove iţ down ŷour ţhroaţ. Finally, I ŵill diśṗośe of ŷour body by deṗosiţinǵ iţ in a duṃṗ ŵhere iţ belonǵś.

Figure 5: Example wrong prediction of "threat"

7 Summary and Future work

The embedding and Bi-LSTM recurrent states are proven to be great source of information because the symantic information for the toxic words and phrases are well captured. Multiple layers of ReLU is the best way to capture the non-linear behavior on top of such features.

However, the number of layers added do not translate into better performance and that extra layers are causing the regularizer of moderately large scale to damage the AUC score a lot. The gap between the best result on leaderboard and us is about 1 percent. It is very likely that hyper parameter training and additional data augmentation will help but we could not afford the time and resources for multiple rounds of training process at this time.

Our future work that may be meaningful is to do the hyper parameter tuning on regularizer, adjuste the dense dimensions of ReLU layers and calibrating the length of bi-directional LSTM layers. These steps require plenty of time and hardware resources, but may be helpful to further improve our Kaggle score.

References

[1] Tanner, Martin A., and Wing Hung Wong. "The calculation of posterior distributions by data augmentation." *Journal of the American statistical Association* 82.398 (1987): 528-540.

[2] Hochreiter, Sepp & Schmidhuber, Jrgen (1997) Long Short-term Memory *Neural Computation 9(8):1735-80*

[3] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370. http://nlp.stanford.edu/ manning/papers/gibbscrf3.pdf

[4] Nair, Vinod & Hinton, Geoffrey E. (2010) Rectified Linear Units Improve Restricted Boltzmann Machines *International Conference on Machine Learning* pages 807-814

[5] Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012.

[6] Pennington, Jeffrey & Socher, Richard & and Manning, Christopher D. (2014) GloVe: Global Vectors for Word Representation*https://nlp.stanford.edu/pubs/glove.pdf*

[7] Luong, Minh Thang & Pham, Hieu & Manning, Christopher (2015) Effective Approaches to Attention-based Neural Machine Translation. , *https://nlp.stanford.edu/pubs/emnlp15_attn.pdf7*

[8] Ma, Xuezhe & Hovy, Eduard (2016) End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF *https://arxiv.org/pdf/1603.01354.pdf*