ERASeD: Exposing Racism And Sexism using Deep Learning

Suvadip Paul
Computer Science
Stanford University
suvadip@stanford.edu

Jayadev Bhaskaran
ICME
Stanford University
jayadev@stanford.edu

Abstract

The rise of hatred and vitriol on online forums and social networks has been a significant problem in the Internet Age. The aim of our project is to train an end to end deep learning models to classify a given Tweet as racist, sexist or neither. We first undertake a complete literature survey of the deep learning models used for this task till date and compare various approaches for this particular problem by implementing all the approaches. Our models surpass current state of the art performance on this dataset (for end to end deep learning models) in literature reaching a weighted macro F1 score of 0.85 and an AUROC of 0.91, primarily through improved model architecture and representations.

1 Introduction

We planned to build a set of deep neural network based classifiers to detect racism and sexism across different Tweets and analyze them qualitatively and quantitatively. We used the most popular dataset for this task released by [1] for Tweet classification. The goal of the model is to be able to classify Tweets as Racist, Sexist or Neither with a certain degree of confidence. All neural models shown so far only achieve a 0.81 weighted macro F1 score at most on this dataset, and we surpass that with our models. Note that this score is purely from the deep learning models used for the task, rather than the other possible traditional machine learning algorithms which have been shown to perform better (when coupled with embeddings generated from neural networks). We have also analyzed these models using the area under the ROC curve. Given that AUROC is threshold agnostic, we truly show improvement in performance and believe that this is a significant advantage of our deep learning models. Traditional baselines such as Logistic Regression and SVMs are reasonably good at this task but they have poor AUROC scores, which had been undocumented across all the literature till date.

The nature of conversational language on Twitter also presents its own challenges. We noticed that around 20% of the clean tokens we generated from the dataset did not have pretrained word embeddings (thanks to online as well as Twitter-specific idiosyncrasies), which is one of the biggest challenges of racist/sexist Tweet classification. The dataset was also very noisy and had a lot of Tweets relating to the same topic or set of topics. We have also not ignored or removed tweets from the training set if we are not able to find representations for it, as some of the models we propose work around this fact. This includes tweets with only emojis, minimal words and hyperlinks.

2 Background/Related Work

2.1 Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter

This work [1] was the first occurrence of the main dataset on which most work around this area has been centered on. The authors provided a dataset of around 16k human-labeled Tweets. They showed

that despite presumed differences in the geographic and word-length distribution, hand generated features had little to no positive effect on performance, and rarely improved over character-level features. They also showed that the sole exception to this rule is gender.

2.2 Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter

This work [2] examined the influence and bias of annotators on creating the ground truth for the data, by contrasting results of classification obtained from training on an expert as well as amateur annotators. This was able to provide insight to understand if a model could pick up annotator bias, and what kind of annotator bias existed across datasets.

2.3 Deep Learning for Hate Speech Detection in Tweets

This was the first significant usage of deep learning models on this dataset. The authors [3] experimented with MLP, LSTM and CNN based architectures and had results that were above the non-neural baselines. However, their best results were through a combination of embeddings generated through a neural network coupled with a non-neural model (such as gradient boosted decision trees). Also, in order to overcome the lack of embeddings for many words they used random embeddings instead of training something more specific towards this task.

2.4 Using Convolutional Neural Networks to Classify Hate-Speech

The authors [4] introduced the concept of using CNNs for this task as well as randomized embeddings. They showed how character 4-grams, word vectors based on semantic information, randomly generated word vectors, and word vectors combined with character n-grams performed when compared to each other, and why the right embedding choice was important. The other takeaway was how one could use typical computer Vision architectures out of the box to build a CNN that performed well in classifying these Tweets.

2.5 One-step and Two-step Classification for Abusive Language Detection on Twitter

The authors [5] attempted a two-step approach of detecting abusive language, first classifying between normal and abusive Tweets and then classifying into racist and sexist, and compared this with a one-step approach of multi-class classification on sexist and racist language. They used CNNs that using both character-level and word-level inputs to perform classification. Their main contribution was using a 2-level classification approach (CNN followed by logistic regression).

3 Approach

Based on the literature review and given the fact that the various papers we have mentioned above have not been benchmarked against each other (although they all work on the same dataset), our first approach was to test the various different hypotheses and improve upon the best results seen in literature. We have only focused on purely end to end neural networks. We found significant performance improvements over our baseline models by using stacked bidirectional LSTMs with attention, as well as CNN-based models with a range of filter kernel sizes. Our models produced F1 scores on this dataset that were better than any existing end to end deep learning models.

3.1 Input

Preprocessing the data is one of the key steps in achieving good results especially for a dataset full of spelling mistakes and abusive comments which are very difficult for a machine to comprehend. For Example "ILIKESLEEP", "I like sleep", "I likeSlleeeeepp" are similar yet different in their own ways. Misspelled words pose a challenge primarily due to the lack of representation in the embeddings. The following are the Twitter-specific preprocessing steps that we used; the methodology followed was a variant of the preprocessing steps used in generating GloVe [11] embeddings from Twitter.

- URLs, hyperlinks, user mentions (@user) were replaced with appropriate placeholder tokens.

- Smileys of all kinds and other special symbols - we identified a few and ignored most.
- Hashtags were replaced by a "<hashtag>" token followed by the hashtag text.
- All capitals before or after were replaced as '<ALLCAPS> word'
- Dates, other numbers and other special characters were handled separately.

The following were the different neural network layers that we used in our models:

1. Input layer: Inputs a given Tweet to the model.
2. Embedding layer: Maps each token to a word embedding, using either pretrained or task-specific generated embeddings.
3. Dense layer: A fully-connected set of neurons with a ReLU activation function.
4. LSTM layer: Contains a set of LSTM cells that help predict short and long-term dependencies between words.
5. Stacked BiLSTM layer: Stacked BiLSTM used to generate a representation for each time step as well as the whole sentence over multiple stacked LSTMs.
6. Attention layer: Used to apply attention weights to individual words within the Tweet, to highlight importance of certain terms during classification.
7. CNN layer: Contains a set of convolutional units with different kernel sizes, applied across the given Tweet.
8. Output layer: Converts the sentence representation into a probability distribution across the three classes (none/racist/sexist), using a softmax function.

3.2 Embedding Layer

This layer converts a set of preprocessed words from the inputs into embeddings. This layer can be represented as a fully connected layer (implemented as a lookup table as the vocabulary is quite large). We tried out the following methods:

- Pretrained Embeddings
  - GloVe (Twitter)
  - GloVe (Common Crawl)

  The biggest drawback of using pretrained embeddings is that 20% of the words we see in the dataset from Twitter do not have embeddings.
- Character Level Embeddings
  - We tried using an embedding for each character in a sentence but the results were not promising.
  - Tweet2Vec embeddings: We used this [10] in two ways: (a) by individually converting all words like 'ILikeSleep' to embeddings and (b) by converting the whole sentence into an embedding using a character level word embedding and LSTM level sentence embedding.

3.3 Sentence CNNs

In recent times, 1-D convolutional neural networks have been found to perform well on sentence classification tasks [6]. We use a wide range of filter kernel sizes (2, 3, 4 and 5) in order to capture different representative n-grams across Tweets.

3.4 Stacked BiLSTM

First proposed by Hochreiter et al (1997) to overcome the problems of vanishing gradients, LSTMs have now become the standard benchmark when it comes to sequence classification. The underlying principle involves the use of gates which decide the degree to which the previous state is retained and the current state is forwarded. A lot of LSTM variants have been proposed; we have used bidirectional LSTMs with attention. We also stack multiple LSTM cells before feeding it to the attention layer.

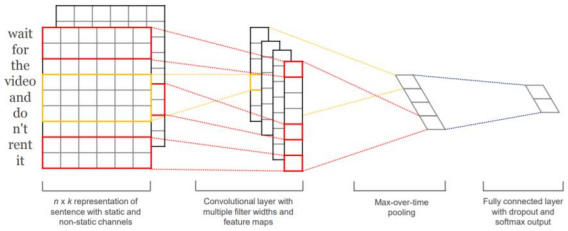
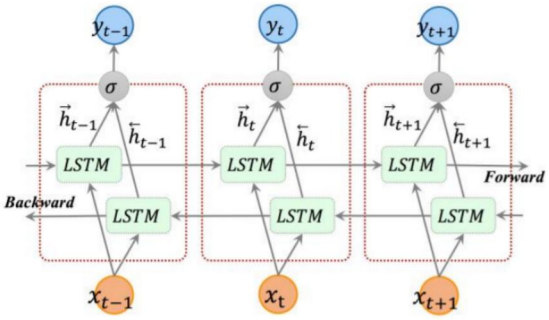


Figure 1: CNNs for sentence classification, from [6]

A stacked LSTM architecture is described below. Each pass onto the next LSTM layer concatenates the previous forward and backward outputs.



$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
\qquad h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}]
$$

Figure 2: Stacked LSTM architecture with LSTM equations, from [8]

3.5 Attention Layer

Attentive neural networks [7] learn how to focus on particular words over others based on their relative importance, and add another layer of understanding to the LSTMs. They have been used for a wide range of tasks like machine translation, question answering, image captioning and speech recognition. We found that this approach has not been done till now by any of the previous works on this dataset. Using bidirectional attention helps us improve performance significantly over existing models in literature.

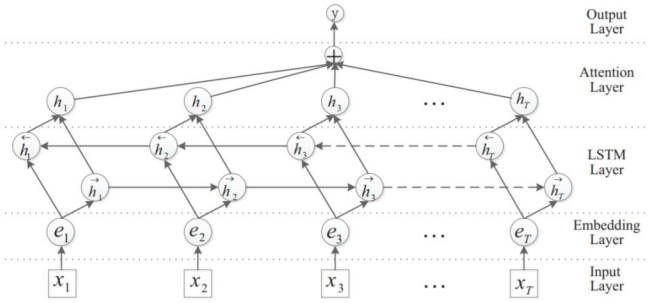


Figure 3: Bidirectional LSTM with Attention, from [9]

3.6 Output Layer

This is the final layer in the network. It takes the representation from the attention layer and is a fully connected layer to 3 outputs, over which we have a softmax function which predicts the required probabilities.

3.7 Regularization and Dropout

Before each of the densely connected layers, we added a dropout layer to automatically work as a regularization tool. We also implemented L2 regularization at each layer to avoid overfitting. Given the small size of our data set, deep neural networks tend to overfit very quickly; regularization and dropout helped us overcome this issue.

4 Experiments

4.1 Data

Waseem et. al. have an open dataset [???] with approximately 17000 tweet IDs and human labeled annotations for each tweet (Racism, Sexism or None). We wrote a parser script to scrape the actual tweet content for these 17000 tweets (through the Twitter API). Notably, some of the tweets have been deleted since the release of the dataset, and we managed to scrape about 15.5k tweets in total for our experiments. The distribution of data among normal/racist/sexist was approximately 70/10/20 (out of a possible 100%). We used a 90/10 train-dev split and reported our dev-set F1 and AUROC scores.

4.2 Baseline

We implemented two simple baselines using a simple bag-of-words approach, without considering sequential ordering or using any word embeddings. We picked non-neural baselines without incorporating sequential information so that we could show the additional performance gain due to each of these factors. For all baselines, we used TF-IDF weighting which was found to boost performance significantly. We also implemented two other baselines (Naive Bayes and Random Forest), but did not persist with them as they severely underperformed when compared to Logistic Regression and SVM,

The following baselines were implemented:

1. Logistic Regression (baseline): We used a multinomial classification scheme to predict the probability of each Tweet being racist, sexist or neither. We also used regularization to improve performance and avoid overfitting.
2. SVM: We tried using an SVM (linear kernel) on the same set of inputs. This was found to show comparable performance to the Logistic Regression model, and was the best performing baseline with respect to weighted macro F1 score.

4.3 Neural Network Based Models

4.3.1 Embedding Types

We experimented with many different representations for this problem. To start off, we used GloVe [11] Twitter embeddings, which we thought made the most sense given that our dataset comprised Tweets. While performance was generally good, we ran into some issues with the large number of out-of-vocabulary words encountered. To fix this, we tried three different approaches. First, we used a set of pretrained character embeddings and ran character-level neural network models, hoping that these would capture any information present in words with abnormal spellings. We also used Tweet2Vec [10], a vector representation of Tweets. We experimented with Tweet2Vec embeddings at both the word level and the Tweet level, as well as a summation of GloVe and Tweet2Vec embeddings (using the latter to capture meaning that was not well-understood by GloVe). At the word level, Twwet2vec uses char-RNNs to generate a representation for a word, and used word-RNNs to generate representation for sentences respectively. Finally, we also tried using GloVe embeddings trained

Name	F1 Score (%)	AUROC (%)
Logistic Regression	79.57	73.32
SVM (Linear Kernel)	81.21	N/A
MLP (Tweet2Vec)	75.45	81.20
MLP (Tweet2Vec Tweet-level embed)	79.45	87.27
MLP (GloVe Twitter + Tweet2Vec)	78.28	83.21
MLP (GloVe Twitter)	78.78	85.29
MLP	80.25	86.48
Char-LSTM (128) (Char embeddings)	75.83	82.83
LSTM (128) (Tweet2Vec)	77.91	89.37
LSTM (128) (GloVe Twitter + Tweet2Vec)	77.11	89.53
LSTM (128) (GloVe Twitter)	81.74	90.54
LSTM (128)	82.50	91.11
LSTM (128) + Attention	82.69	91.05
BiLSTM (128) + Attention	83.45	**91.14**
2-Stacked BiLSTM (128) + Attention	83.47	88.76
2-Stacked BiLSTM (256) + Attention	83.55	89.66
3-Stacked BiLSTM (256) + Attention	82.81	88.96
CNN (100*4, Kernel Size 2, 3, 4, 5) (GloVe Twitter)	83.21	90.80
CNN (100*4, Kernel Size 2, 3, 4, 5)	**84.83**	90.39

Table 1: Summary of Results (Embeddings: GloVe Common Crawl unless mentioned otherwise)

on the Common Crawl corpus, which we felt would improve performance due to a significantly increased training corpus size.

4.3.2 Model Architectures

We experimented with the following neural network models:

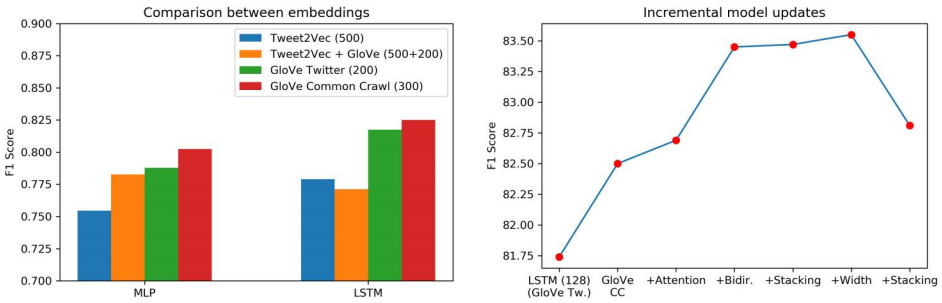
1. Dense Multi Layer Perceptron (Feedforward Neural Network): Two hidden dense layers, each of size 100
2. LSTM with hidden layer size-128
3. LSTM with Attention: hidden layer size-128, followed by an Attention layer
4. Bidirectional LSTM with Attention: Bi-directional with hidden layer size-128, followed by an Attention layer
5. Stacked Bidirectional LSTM with Attention: 2 layers of Bidirectional LSTM units, followed by an Attention layer
6. CNN: Kernel sizes of 2, 3, 4 and 5, with 100 filters for each kernel size (overall 400 filters)

Using a weighted loss function helped improve performance by countering the effects of class imbalance in our dataset. We performed hyperparameter tuning on learning rate (final value used was 0.001) and added dropout/regularization (specific to each network). To evaluate the performance of our models, we used weighted macro F1 scores (primary metric). We also looked at the area under the ROC curve (AUROC), which is a good measure of model generalization irrespective of class imbalance and is agnostic to picking a threshold (unlike F1 scores).

4.4 Results and Analysis

We found that non-neural baselines performed very well on the task, due to the small size of the dataset. Neural models showed a tendency to overfit very quickly, but we managed to get better performance than the state of the art deep learning models used on this dataset in prior works.

We started by representing the data using GloVe Twitter embeddings, as we felt that it would be the best match for a Twitter dataset. However, we found a lot of out-of-vocabulary tokens, primarily through misspellings, neologisms and hashtags.



(a) Embeddings (b) Ablative Analysis

Figure 4: Effect of embeddings and model architecture

Using pretrained char-level embeddings on a char-LSTM did not perform as well as pretrained GloVe vectors. The effect of Tweet2Vec embeddings (in both word and Tweet formats) did not help either, as the corpus size on which these were trained was much less than that of GloVe embeddings. We also tried concatenating GloVe and Tweet2Vec embeddings for words to capture information that each method might have missed. In an effort to improve performance, we switched to the GloVe Common Crawl embeddings (840B tokens, 300 dimensional embeddings) and this resulted in a noticeable boost in F1 scores (Fig. 4(a)). This encouraged us to try using this as the default embedding for further models. We tried training the embedding layer as well to create our own embeddings but due to the dataset size, it just ended up in reduction in performance.

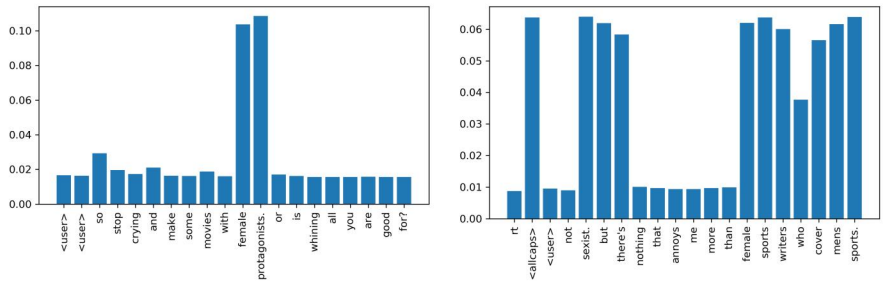


Figure 5: Attention Scores

The main reason why our LSTMs performed better than existing models was due to bidirectional attention and stacking. The effect of each of these factors is shown in Fig. 4(b). The attention mechanism was especially helpful in giving emphasis to parts of Tweets that were indicative of potential racism/sexism, by highlighting race-related/gendered terms (Fig. 5). An interesting observation was the high attention score given to "not sexist but" - we were surprised by the number of sexist Tweets that contained this phrase within them!

Our best results (in terms of F1 scores) were from a CNN-based model. Using a range of filter sizes helped capture 2, 3, 4, and 5-grams that were indicative of racist/sexist content. The hypothesis that we propose for this is that a lot of abusive content centers around specific n-word phrases, and CNNs do a great job of capturing this information.

In an effort to try out an alternate approach, we tried to split the problem into a two-stage classification problem (first separate normal from abusive, then racist from sexist). We observed that separating racist and sexist Tweets was an easy problem, but performance on separating normal Tweets from abusive ones was as hard as separating Tweets into normal, sexist and racist. To verify if this was the case, we plotted the tSNE projections of each Tweet (using Tweet2Vec embeddings) into a 2D-plane (Fig. 6). We observed that racist and sexist Tweets were clustered into separate regions (and not clustered together), validating our experimental observations.

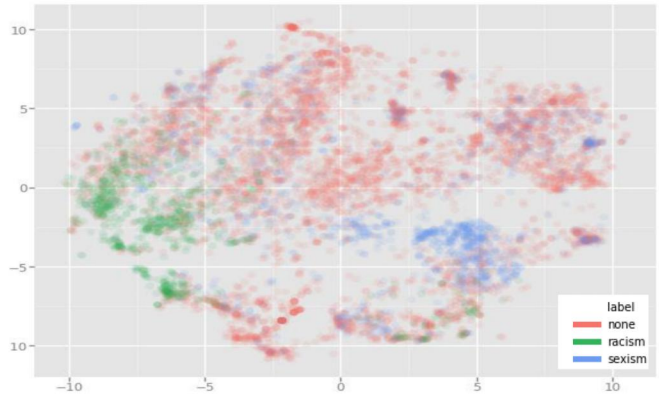


Figure 6: t-SNE Visualization

We used F1 scores primarily to compare with previous works on the same dataset. However, we think that the area under ROC curve (AUROC) is another metric that can highlight how well the classifier generalizes irrespective of class imbalance. To this extent, we observed that our neural models have significantly higher AUC scores than the non-neural baselines.

5 Conclusion

- F1 score was better than all previous end to end neural models, improvement was mainly through bidirectional attention (LSTM) and broad range of filter size (CNN), as well as choice of embeddings.
- AUROC was significantly higher for neural models than non-neural baselines (models generalize better in spite of class imbalance).
- Weighted loss function helped improve performance significantly. About 20% of the words in dataset do not have embeddings. Char-LSTMs were tried but did not help increase performance.
- CNN performed best – indicates presence of characteristic racist/sexist n-grams ("not sexist but" seen in many sexist Tweets).
- Two-stage classification: racism/sexism is an easy problem, but normal/abusive is not (tSNE shows the representation of each Tweet from the dataset visualized and why this is such a hard problem).

5.1 Future Work

- Generate better embeddings for a Twitter dataset with a lot of bad spellings and hate speech (Twitter GloVe is not good enough).
- Dataset too small to train embeddings for a lot of the missing words - create a large Twitter dataset and retrain our models to.
- Build an end to end char-level + word-level LSTM to achieve better results (Tweet2vec did not do well enough).

References

[1] Zeerak Waseem, Dirk Hovy. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter, Proceedings of NAACL-HLT, pages 88–93, 2016, Association for Computational Linguistics

[2] Zeerak Waseem. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter, Proceedings of 2016 EMNLP Workshop on Natural Language Processing and Computational Social Science, pages 138–142, 2016, Association for Computational Linguistics

[3] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, Vasudeva Varma. Deep Learning for Hate Speech Detection in Tweets. arXiv:1706.00188

[4] Bjorn Gamback, Utpal Kumar Sikdar. Using Convolutional Neural Networks to Classify Hate-Speech. Proceedings of the First Workshop on Abusive Language Online, pages 85–90, 2017, Association for Computational Linguistics.

[5] Ji Ho Park, Pascale Fung. One-step and Two-step Classification for Abusive Language Detection on Twitter. arXiv:1706.01206

[6] Yoon Kim. Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, 2014 Association for Computational Linguistics

[7] Colin Raffel, Daniel P. W. Ellis. Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems. arXiv:1512.08756

[8] Zhiyong Cui, Ruimin Ke, Yinhai Wang. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. arXiv:1801.02143

[9] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, Bo Xu. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 207–212, 2016 Association for Computational Linguistics

[10] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, William W. Cohen. Tweet2Vec: Character-Based Distributed Representations for Social Media. arXiv:1605.03481

[11] J. Pennington, R.Socher, and C. D. Manning. GloVe: Global vectors for word representation. Proceedings of the Empirical Methods in Natural Language Processing (EMNLP), 2014

[12] Christos Baziotis. Attention implementation using Keras. Github repository link: https://gist.github.com/cbaziotis/6428df359af27d58078ca5ed9792bd6d