Dynamically Deriving Comment Toxicity using Neural Networks and Word Embeddings

Abolfazl Farahani
Department of Computer Science
University of Georgia
Athens, GA 30602
a.farahani@uga.edu

Jonathan Myers
Department of Computer Science
University of Georgia
Athens, GA 30602
submyers@uga.edu

Saed Rezayi
Department of Computer Science
University of Georgia
Athens, GA 30602
saedr@uga.edu

Abstract

As society emboldens social media, intrinsic societal divisiveness emerges from, in part, the categorical goals of commonality algorithms embraced by social media systems. Addressing such concerns entails acknowledging and regulating potentially offensive comments. Autonomous derivation of how comments might be construed emerges as the primary goal for social media self-regulation. Following such continuing challenges, this document illustrates how certain techniques, specifically Convolutional neural networks (CNN) and Long Sort-Term Memory (LSTM) using word embedding, accurately derive potentially offensive interpretations of social media postings, thereby regulating user interpretation.

1 Introduction

The importance of how people interpret comments posted continues to rise in modern times as societal embarrassment of such communicative venues expands, and, in many respects, solidifying intrinsic societal divisiveness. Companies managing such ventures find regulating comment interpretation highly advantageous for many pragmatic facets, such as perception and potential regulation. Therefore, deriving a methodology by which predicting how such comments might be interpreted becomes intrinsically vital.

The Kaggle competition "Jigsaw Unintended Bias in Toxicity Classification" is one of recent sentence interpretation competitions. The Jigsaw company, a subsidiary of Google and Alphabet Inc., came about in 2010 (initially as "Google Ideas") as a research firm primarily focused on interpretative machine learning technologies for marketing and cyber security. Their "Civil Comments" web service allowed them to gather comments, and Berkeley's "Online Hate Index" research project surveyed the results to define appropriate toxic scores. The data simply consists of sentences paired with toxic scores divided into training and testing sets along with a specific scoring; the toxic scores and identity groups for test set members are hidden.

The challenge now entails deriving an approach to predict in how surveyor score comment toxicity. As computer science techniques have rapidly evolved over the past decades, addressing this problem gave us the opportunity to evaluate different data sources and analytic methodologies. Here we discuss a basic statistical evaluation of patterns in sentiment followed by the application of varying

Preprint. Under review.

degrees of machine learning techniques, CNN and LSTM (Hochreiter and Schmidhuber, 1997), on ever advancing word embedding techniques, e.g., word2vec (Bengio et al., 2000), Glove (Pennington et al., 2014), and fastText (Mikolov et al., 2018).

Conversation AI team is a collaborative research effort exploring ML as a tool for better discussion online. When they first built toxicity models, they realized that the model learned to associate the names of frequently attacked identities with toxicity resulting the high likelihood prediction of toxicity for comments containing those identities even when the comments were not toxic. In this challenge we are going to tackle the aforementioned problem by building a model that is able to recognize toxicity while minimizing unintended bias with respect to mentions of identities.

2 Related Work

The social media is growing rapidly and many people are spending large amount of time in online conversations and social networking sites to communicate with others. While users can benefit and learn from these social interactions, there exists a risk of being exposed to the offensive and toxic comments.

Many researches have been done to make this virtual environment safe by applying machine learning approaches to automatically detect toxic comments in online conversations. Manual feature engineering, (Burnap and Williams, 2015), (Mehdad and Tetreault, 2016), (Kennedy et al., 2017), (Samghabadi et al., 2017), (Robinson et al., 2018), is one of the techniques in which the manually selected features are combined to create the input vectors and will be directly used for classification.

While the previous approach manually select the features, neural networks especially Convolutional Neural Networks (CNN) seem to be more effective and appropriate since they automatically learn the abstract representations from labeled data regardless of syntactic or semantic knowledge of a language. (Zhang and Luo, 2018), uses character-level convolutional neural networks and show that using character-level features is a compelling method in classification tasks. (Lai et al., 2015), proposed recurrent CNN for text classification in which recurrent structure of model extracts contextual information and CNN part constructs the representation of text data. (Georgakopoulos et al., 2018), applies CNN to detect toxic comments in a large amount of documents provided by Kaggle's competition.

LSTM is another approach that treats text as sequences to capture its spatial pattern, (Mousa and Schuller, 2017), (Tai et al., 2015). (van Aken et al., 2018) Applies LSTM model in which embedding layer transforms one-hot-encoded words to dense vector representation. Bi-LSTM used in (van Aken et al., 2018) have two LSTM layers that can process the input sequence in opposite directions. Thus, it can compensate certain errors on long range dependencies that LSTM is not able to deal with and recognize relations between words in longer sentences where words are further apart from each other. In this projec we use CNN, LSTM and Bi-LSTM to find a solution to the aforementioned problem.

3 Methodology

In this section, we first discuss feature selection procedure, and then explain our model architecture in depth.

3.1 Feature Extraction

Since our only data source is text, we consider following three categories of features:

1. TF-IDF: Term Frequency-Inverse Document Frequency is a numerical approach that converts each word to a value that is a measure of importance with respect to its document in a bag of documents.
2. Sentiment: is a numerical value that measures the contextual polarity of a sentence it can be either positive, negative, or neutral.
3. Word embedding: Word embedding algorithms are a family of unsupervised learning methods for obtaining vector representations for words. We use existing pre-trained word embedding such as GloVe (Mikolov et al., 2018) and word2vec (Bengio et al., 2000).

3.2 Convolutional Neural Network

As discussed in Section 2, CNNs have been used for classifications tasks, such as Sentiment Analysis, Spam Detection or Topic Categorization, they capture semantic similarity within a certain window around words. So they are powerful to extract local structure of sentences. Another advantage of CNNs is that they are fast.

In this work we implement a deep CNN model in Python using Keras library. As explained in the previous subsection, we use GloVe word embedding to initialize the weights of the embedding layer. Following the Embedding layer we use a one dimensional convolutional layer with kernel size of 3. We apply a maxpooling filter on the output of convolution layer and feed the filter output to a fully connected network of 50 neurons with ReLu as the activation function. The final layer is a single neuron that performs sigmoid function. To train the model we minimize a binary crossentropy loss function with Adam optimizer. Binary crossentropy loss function is formulated as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

where N is the number of samples in the training set, y is the true label and $\hat{y}$ is the predicted laebl.

3.3 Long Short Term Memory

Recurrent Neural Networks (RNNs) are one of the most powerful techniques for sequential data such as text, audio, video, and many more. LSTM is a type of RNN that unlike other common neural networks, has feedback connections and hence learns the long dependency among words.

In this work we feed the text to an LSTM network after being embedded by the GloVe algorithm. Let the input sequence be denoted as $X_T = \{x^1, x^2, \cdots, x^{\mathrm{T}}\}$, where T is the maximum length of the comments, the output of LSTM layer, h^t, is calculated as follows:

$$\begin{aligned} a^t &= \tanh(W_i.[x^{t-1}; a^{t-1}] + b_i), \\ \hat{h}^t &= \sigma(W_a.a^t + b_a), \end{aligned}$$

where a^{t-1} is the output of the previous LSTM cell, a^{t-1} and x^{t-1} are the inputs of the current LSTM cell, W_i, b_i, W_a, b_a are the parameters of the model, and σ is the sigmoid function. The output of the LSTM layer is then supplied to a 2-layer dense network with a number of units of 256 and 256, respectively. Similar to the previous model, we minimize a binary crossentropy loss function with Adam optimizer to train the classifier.

We also trained a second LSTM model with a different initialization. For the second LSTM model we concatenated the 300 dimensional GloVe representation with another 300 dimensional word2vec representation and used it in the embedding layer.

4 Experiments and Results

4.1 Data Set

Viewing Jigsaw supplied training data, containing toxic real number measures in the range $[0, 1]$, immediately highlights skewed data. Non-toxic comments, i.e. entries with a 0 toxic score, account for $\sim$70% of the training data set. Filtering out the non-toxic examples shows certain scores account for a predominant portion of the samples, see figure 1.

The scores $1/6$ and $1/5$ together account for $\sim$14% of samples, and the following six most common values cover $\sim$10% of the population. Together the top nine categories explain $\sim$94% of the training population, and their values suggest the predominant survey scoring came from five or six surveyors. The remaining $\sim$6% of the population cover 2,904 different toxicity score values.

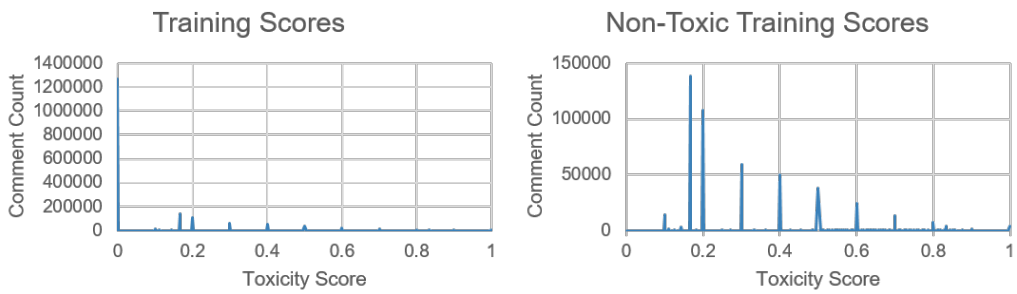

Figure 1: Toxic Score Sets

4.2 Evaluation Metric

As part of the competition, Kaggle evaluates predictions submitted by measuring unintended bias over multiple categories (Borkan et al., 2019). A submission file only consists of each test entry's id and the predicted toxicity score. Receiver Operating Characteristics Area Under the Curve (ROC-AUC) reflects an overlap between true positive and true negative values — ROC-AUC score 1 is optimal, score 0.5 reflects worst possible performance, and score 0 signifies complete accuracy supplying wrong classification — spanning over certain testing data set:

1. Entire test data set
2. Subgroup AUC – accounts for comments that contain reference to certain "identity" groups, including *male*, *female*, *homosexual gay or lesbian*, *christian*, *jewish*, *muslim*, *black*, *white*, and *psychiatric or mental illness*
3. Background Positive, Subgroup Negative (BPSN) AUC: This compares non-toxic occurrences with an identity with toxic occurrences without the identity
4. Background Negative, Subgroup Positive (BNSP) AUC: Compares the grouping of toxic occurrences with an identity with non-toxic occurrences without the identity

The process then applies the following generalized mean of bias equation to the identity group AUCs defined above for subgroups (entries 2-4):

$$M_p(m_s) = \left(\frac{1}{N} \sum_{s=1}^{N} m_s^p \right)^{\frac{1}{p}}$$

where:

$M_p =$ the pth power-mean function
$m_s =$ the bias metric m calculated for subgroup s
$N =$ the number of subgroups
$p =$ the constant (-5)

Combining the equations defined above result in the final score given to a submitted prediction:

$$score = w_0 AUC_{overall} + \sum_{a=1}^{A} w_a M_p(m_{s,a})$$

where

$A =$ number of submetrics (3 in this case — entries 2-4 above)
$m_{s,a} =$ bias metric for identity subgroup s using submetric a
$w_a =$ a weighting for the relative importance of each submtric, all have value 0.25 here

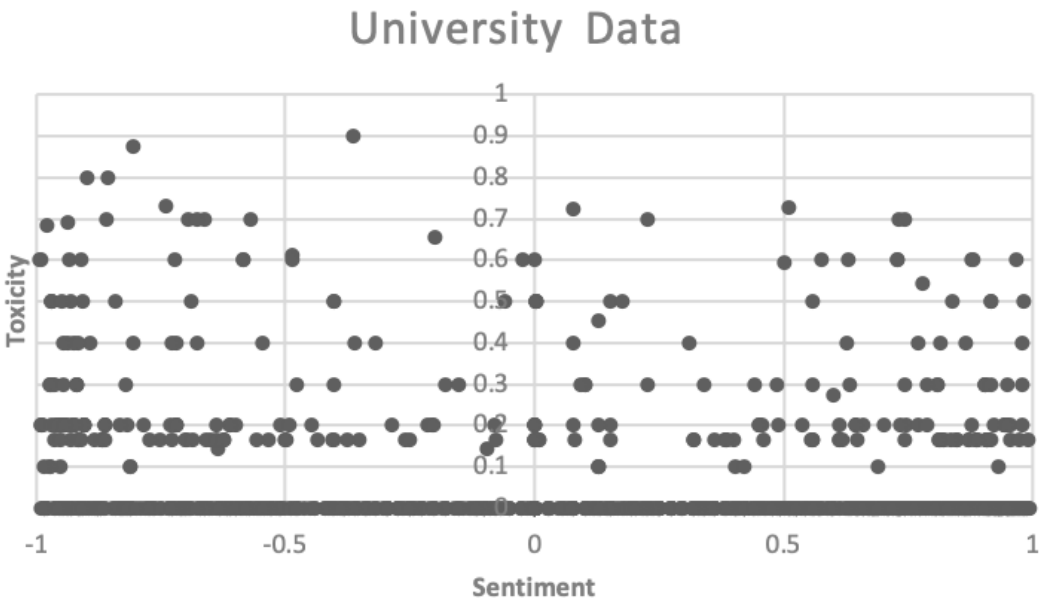

Figure 2: *university* cases

4.3 Sentiment Analysis

Due to the limited nature of data supplied by Kaggle, we initially looked to sentiment as a sentence property we could measure and use as another factor to evaluate potential causality. The recent Vader (Hutto and Gilbert, 2014) software from Georgia Tech provides positive, negative, and neutral, along with a compound score ranging over $[-1, 1]$.

After applying Vader to all training and testing data, we hypothesized finding correlation between a word's occurrence and the sentences sentiment could serve as a tool for deriving toxicity score predictions. Evaluating this approach demanded filtering and adjusting the data considered to boost the commonality of words between training and testing data and reduce the total number of words evaluated. Achieving such goals in sentence analysis often entails compromised filtering, which, in our case, consisted of the following options:

1. Lower the case of all letters
2. Remove special characters (optional: allow single quote inside word to remain)
3. Apply spell check and remove unrecognizable words
4. Remove words that don't occur in both training and testing data sets

The final number of words varied based on certain options used, but the word sets evaluated ranged in size from $\sim$61k to $\sim$68k. We then generates a directory consisting of CSV files for each word with one row for each occurrence of the word in the training set with values:

<toxicity score>,*<sentiment score>*

The resulting graph showed large variance; figure 2 illustrates the sentiment and toxicity scores for all occurrences of the word *university* in the training data set:

The wide variance in such sets made us consider regulating how and when such values should be used. For each comment in the test set, we would calculate the comment's sentiment, look for a subset of each word's occurrence in the training set in one of 11 fixed subset ranges $([-1, -0.95), [-0.95, -0.85), ..., [0.95, 1])$, and calculate statistics values for the subset, including mode, mean, median, std, and skew. We then filtered out words with high variance, greater than a predefined constant, to end up with the final word set W. A toxic score prediction for the comment came from the following equation:

$$predict = \sum_{w \in W} \frac{mean_{max} - mean_w}{(\sum_{w' in W} mean_{max} - mean'_w)} t_w$$

where t_w was conditionally selected from the following conditions:

$$t_w = \begin{cases} mode_w & \text{, if } std_w \geq v_1 \text{ and } skew_w \geq w_1 \\ median_w & \text{, if } std_w \in [v_2, v_1) \text{ and } skew_w \in [w_2, w_1) \\ mean_w & \text{, otherwise} \end{cases}$$

Despite testing a wide range different word filtering options and fluctuation in constants, such as v_1, v_2, and maximum variance allowed, our highest Keggle score was approximately $\sim$0.66. Knowing the basic statistical approach Naive Bayes TF-IDF scored $\sim$0.78, the incorporation of sentiment data with the conditional weighting function was not beneficial (see Table 1).

4.4 Results

Here we present our final results and we include Naive Bayes as a baseline. As table 1 demonstrates, the best result was obtained using LSTM and initializing the weight vectors by the concatenation of GloVe and word2vec embeddings.

Table 1: different methods and their associated scores

Model	Score
Sentiment	66%
Naive Bayes (TF-IDF)	78%
CNN (GloVe)	88%
LSTM (GloVe)	89%
LSTM (Glove & word2vec)	**93%**

5 Conclusion and Future Work

While our final score is not far from the top score on Kaggle, our approach used a very mild set of initial settings to compensate for the identity groups defined in the training set. Since Kaggle does not provide identity group information in the test level data set, we could use word2vec in CNNs or LSTMs: one for each identity group. Evaluation of such a pursuit would require 10-fold evaluation over the training set — possibly 5-fold or filtered data set since the inclusion identity of a group only demanded at least 500 occurrences of the property.

Second, since a dominant majority of the train set took one of nine clear values, we could discover if adjusting predicted values to near, dominant values would improve the final score. Testing different options for such adjustments would not be computationally demanding, but the five submission per day limitation of Kaggle would likely cap potential improvement possibilities.

References

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Borkan, D., Dixon, L., Sorensen, J., Thain, N., and Vasserman, L. (2019). Nuanced metrics for measuring unintended bias with real data for text classification.

Burnap, P. and Williams, M. L. (2015). Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Georgakopoulos, S. V., Tasoulis, S. K., Vrahatis, A. G., and Plagianakos, V. P. (2018). Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, page 35. ACM.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Hutto, C. J. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*.

Kennedy, G., McCollough, A., Dixon, E., Bastidas, A., Ryan, J., Loo, C., and Sahay, S. (2017). Technology solutions to combat online harassment. In *Proceedings of the first workshop on abusive language online*, pages 73–77.

Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Mehdad, Y. and Tetreault, J. (2016). 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.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mousa, A. and Schuller, B. (2017). Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1023–1032.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Robinson, D., Zhang, Z., and Tepper, J. (2018). Hate speech detection on twitter: feature engineering vs feature selection. In *European Semantic Web Conference*, pages 46–49. Springer.

Samghabadi, N. S., Maharjan, S., Sprague, A., Diaz-Sprague, R., and Solorio, T. (2017). Detecting nastiness in social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 63–72.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

van Aken, B., Risch, J., Krestel, R., and Löser, A. (2018). Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572*.

Zhang, Z. and Luo, L. (2018). Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, pages 1–21.