Exploring Deep Learning in Combating Internet Toxicity

Susan Bhattarai
Department of Computer Science
Stanford University
Stanford, CA 94305
sushanb@stanford.edu

Abstract

Growing online communities like Reddit and Wikipedia has enabled us to share our opinions and increase our knowledge base. However, the freedom of expression has also led to online abuse and harassment which can deter people to leave the communities. These toxic conversation leading to abuse, harassment and hate-speech can be classified into types such as toxic, severe-toxic, obscene, threat, insult and identity-hate. The variants of comments are all over the web which have prevented people from participating and contributing in online discussion. In this paper,we develop two models deep learning based models for classification namely convolutional neural network with multiple filter sizes and deep bidirectional LSTM neural work for classifying sentences having different forms of toxicity. We use dataset of comments from Wikipedia's talk page edits for training and testing our models. Our best model uses a bidirectional LSTM over maxpooling with Glove vectors as word embeddings and obtains 0.9820 mean column-wise AUC score.

1 Introduction

The massive increase in the accessibility of the web and freedom of speech has caused in toxic, vulgar and hate speech conversation in the internet. Social networking sites like Facebook and Twitter and online communities sites like Reddit and Wikipedia have become key victims of online toxicity and hate speech. Since these forms of the comments are created in a huge number on a daily basis, it is very hard for a site admin to moderate and block these comments. As a result, these platforms struggle to facilitate conversations for all other users which cause people to leave the platform.

According to Center for Innovative Public Health Research, 47% of American have experienced online harassment or abuse [9] . In fact any sites that allow user contents and opinions struggle with toxic and hate speech contents. The effect of online hate speech and offensive messages can affect the people psychologically and mentally. To provide a scale of cyber-bullying and toxic comments, a report stated that there were 10 million player reports on 1.46 million toxic players in the famous game The League of Legends(LoL) [7] . Social media companies Facebook , Twitter and Googles YouTube have greatly accelerated their removals of online hate speech, reviewing over two thirds of complaints within 24 hours, new EU figures show [4]. Thus, it would be great if we can explore deep learning models to explore the classification of comments into different toxicity labels. It can help us reduce the content of hate speech and toxicity in the internet and lead to more discussion and user participation in online settings.

The problem of comments classification falls under one of the widely researched area in Natural Language Processing (NLP). Text classification tasks such as sentiment classification have been widely researched problem in NLP literature dating back to 2000s [11]. Although this task closely resembles the classic classification tasks in Natural Language Processing, it is different and more complex in two ways. Toxic comment classification is a multi-labeled text classification tasks and the classification ranges over six toxicity labels namely toxic, severe-toxic, insult, identity-hate, threat and obscene. Classification of a comment as toxic and non-toxic can be easy with the state of art deep learning model but the classification into sub-categories mentioned above seems complex as there does not exists a well-set decision boundary between labels such severe-toxic and toxic or threat and obscene.

In this paper, we will be exploring two models in the deep learning and NLP literature that have been used in the sentence classification tasks with success. The input to our model is a text and we use our deep learning models

to predict the probability of occurrence of each of six toxicity labels. One of the models is a variant of Bidirectional Long Short Term Memory (BiLSTM) and another is a variant of simple single layer convolutional network (CNN). We also examine the question of how successful are these deep learning models in comparision to the non-neural model. Therefore, we also train a non-neural baseline for this comparision. Our non-neural baseline is a logistic regression classifier using the tf-idf vector of the each word and bigrams of the sentence.

The rest of this paper is organized as follows: Section 2 briefly describes the work done in the field of the sentiment classification and the related work done in the deep learning model we will be using for the multi-headed toxicity classification. Section 3 presents our with pooling model and layer convolution network for the classification task and the underlying architecture behind it. Section 4 presents our experiments with both of the models and our results. Section 5 presents the conclusion and the future works for fine tuning our models.

2 Background

2.1 Deep Learning Work

The classification of the toxicity and hate-speech is an emerging field in the Natural Language Processing. It closely resembles the sentiment analysis field but the spectrum of the labels in this classification approach is broad as we can have many types of toxicity in comments. Currently, a lot of state of the art text classification system in NLP use variants of CNNs and RNNs. RNNs perform the same task for every word present in the sequence and pass the hidden computation to the next layer. Given how RNNs are able to preserve the temporal locality of the sentence and able to cater the variable length of the text, the advanced forms of RNNs like GRUs and LSTMs are widely in NLP research for classification tasks.

Wang and et.al proposed the use of Long Short Term Memory recurrent network for predicting the polarity of the tweets using word embeddings in 2015 [15]. Zhou et.al also presented the way how bidirectional LSTM combined with maxpooling can improve the accuracy of text classification [17]. Next, CNNs currently are the building blocks of every robust state of the art deep learning computer vision models. In 2011, Collobert and Weston proposed a general convolutional network architecture and describe its application to many well known NLP tasks like Name Entity Recognition and so on. In 2013, Kim proposed a simple CNN model with little hyper-parameter tuning which achieves state of the art in many text classification tasks.

2.2 Work on Web Toxicity

We researched the ongoing work on the web toxicity related field. It seems there are two main projects going on at Google and Wikipedia. Google has released a Perspective API that uses machine learning techniques to find abuse and toxicity online. A joint paper by Google Jigsaw and Wikipedia states that they have generated over 100k high quality human-labeled comments and 63M machine-labeled ones from a classifier [16]. These datasets will be used for further research in toxicity classification. Not a surprise at all, the ongoing Kaggle competition from where we are using this dataset uses the data from the Jigsaw-Detox project.

3 Approach

3.1 Non Neural Baseline

To evaluate the performance of the deep learning models, we decided to use the results from the non-neural model as the benchmark. We implemented a logistic regression with Stochastic Average Gradient as the solver and we created two weighted tf-idf vectors for each of the sentence one of the word tokens and another vector for the bigram tokens in the sentence. We did not do any fancy prepossessing like stemming and lemmatizaton as we were truly looking for a benchmark and trying to focus on the deep learning models.

3.2 Neural Models

3.2.1 Convolutional Neural Networks

The structure of our CNN model is based on the model proposed by Kim in his 2011 paper *Convolution Neural Networks for Sentence Classification* [6]. We chose this model for this classification task is because the paper states that the CNN model proposed was able to improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

The input layer of this CNN model is the word embedding. The total number of rows of this embedding matrix is given by **N** where **N** is the number of comments in the training set. Each row of the embedding matrix contains vector of size corresponding to an unique word in the vocabulary. If a sentence has less than **n** words, we pad the sentences and if a sentence has more than **n** words, we only take the **n** words.Thus, the sentence can be represented as:

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$$

where $\mathbf{x}_i \in \mathbb{R}^k$ is the embedding of the i-th word in the sentence, and $\mathbf{x}_{i:i+j} \in \mathbb{R}^{kj}$ is used to denote the embedding matrix representing the i-th word to the j-th word.

In the original paper, it uses two channels to represent the static and non-static word embeddings. Static embeddings are not trainable but non-static word embeddings can be trained via the backpropation along with the paramters of the model. We only use the non-static channel in this architecture as the static channel did not perform well in our classification task.

We apply the convolution on the top of the embedding layer. Given a window size of **h**, this layer is encoded by a conv. filter $\mathbf{w} \in \mathbb{R}^{hk}$. When this filter is applied to the h-gram $\mathbf{x}_{i:i+h-1}$ of the input sentence, a feature c_i is generated by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

where $b \in \mathbb{R}$ is bias and f is the non linear activation unit. The filter slides over all the possible h-grams of the input sentence to produce a feature map

$$\mathbf{c} = (c_1, c_2, \ldots, c_{n-h+1})$$

for $\mathbf{c} \in \mathbb{R}^{n-h+1}$.

Then, we apply max-over-time pooling (pooling over the entire sequence window) to summarize each filter by producing a single feature $\hat{c} = \max(\mathbf{c})$.

The last layer is a fully-connected layer with sigmoid activation function and the input for this layer is the single feature selected by multiple filters with different window sizes. We apply sigmoid activation to calculate the six class probabilities for each of the toxicity labels. The original paper uses softmax activation in the output layer as the classification is not multilabeled.

$$y_i = \frac{1}{1 + \exp(-x_i)}$$

The architecture for our CNN is shown in Figure 1.

3.2.2 BiDirectional LSTM with Maxpooling

Though RNN is adapted to make use of sequential data, it is very difficult to train the model because our length of sentences can be large. It experiences vanishing and exploding gradient problems. We will be making use of the LSTM architecture proposed by Hochreiter et al. [**?**]. The LSTM cell can be mathematically defined by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

LSTM cell introduces an adaptive gating mechanism, which decides the degree to keep the previous state and memorize the extracted features of the current data input.That's the reason why they are better than vanilla RNNs and they do not suffer from vanishing gradient problem. The input of our model is very similar to the CNN model. We transform the discrete features of the sentence words into continuous vectors representation using the pre-trained word embeddings and concatenate them and feed into both the forward LSTM and backward LSTM. The primary intuition on using both forward LSTM and backward LSTM is to get the future input information from the current state. As a result, the output layer can get context from both past and future states. We gather the output from both forward and backward hidden states and concatenate to form an intermediate layer. To reshape the dimension for the final output, we run the maxpool over the 3D tensor intermediate output using the global max pooling. After that, we feed the output to the final sigmoid layer for the classification.

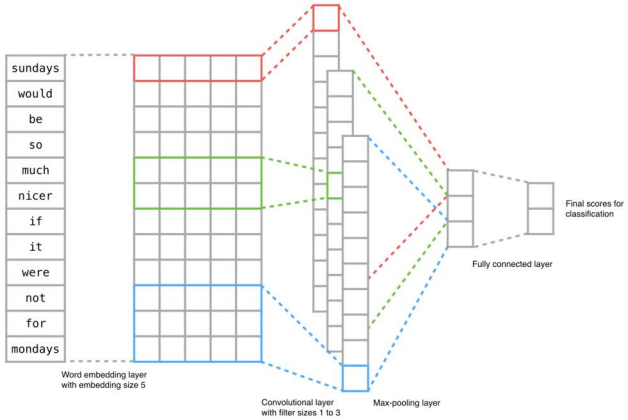

Figure 1: CNN Architecture of our model (the softmax layer is replaced by sigmoid layer)
(Taken from Kim Yoon Paper)

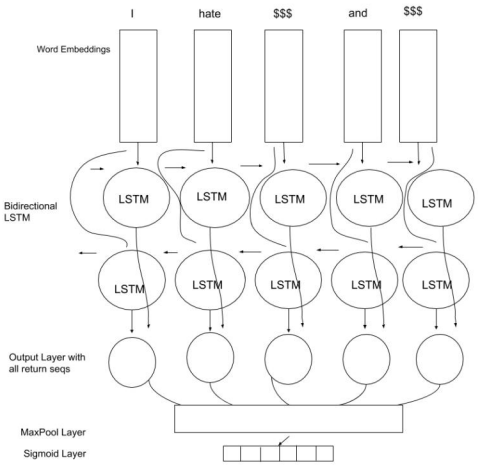

Figure 2: Bidirectional LSTM architecture of our model

4 Experiment

4.1 Dataset

We collected the dataset from an ongoing Kaggle competition Toxic Comment Classification.[1]. The dataset contains 157K Wikipedia's Talk page edits comments which are annotated by 5000 crowd-workers on the basis of their toxicity. Each comment is rated by 10 crowd-workers. Each comment is labeled with 6 labels which are not mutually exclusive i.e each sentences can have more than one positive toxicity labels. The labels are "toxic", "severe-toxic", "obscene", "threat", "insult" and "identity-hate". The hidden test data set contains 153k comments and we have to submit our predictions on Kaggle to evaluate our model performance on hidden dataset. Figure 3 and Figure 4 shows the frequency of distributions of labels and Figure shows the correlation existing between each of the labels. Almost 130k of sentences do not have any toxicity labels i.e they are free from the toxicity.

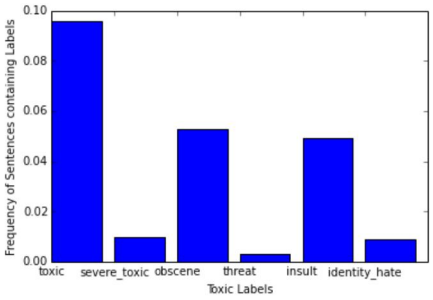

Figure 3: Distribution of toxic labels among training data

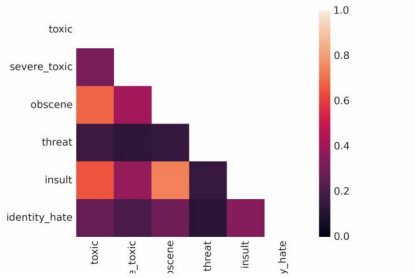

Figure 4: Correlation between each of the labels

4.2 Word Embeddings

We experimented with two different pre-trained word embeddings word2vec by Mikolov et al. [10] and GloVe vectors by Pennington et al. [13]. Using a few trials, we found 100 dimensional Glove vectors trained on 2B tweets works for the best model. The reason why the glove vectors trained on Twitter data worked for us because tweets closely resembles our comments in our dataset. Given the small volume of the dataset 150k, 300d word vectors were not efficient enough as 100d word vectors.

[1]The competition ended yesterday (March 20th).

4.3 HyperParameter Used

Table 1: BiLSTM+Maxpooling Parameter

Hyperparameters	Value
max sentence length	100
word embedding size	100
dropout after word embedding	0.1
num of BILSTM Cells	60
rows in embedding matrix(features)	40000
dropout after maxpooling	0.1
learning Rate	0.01 - 0.001
optimizer	adam
batch-size	64

Table 2: CNN Parameters

Hyperparameters	Value
sentence length	80
word embedding size	100
filter window size	3, 4, 5
number of filters per window size	128
dropout probability	0.5
multi channel	off
batch-size	50

We tried to use the hyper parameter used by Kim in the CNN paper. One exception was the use of adam as optimizer instead of sgd. Use of sgd made the training time longer significantly. At last, we tuned the dropout probability for BiLSTM to be 0.1 and CNN to be 0.5.

4.4 Dropout

We experimented how our model will react when we do not perform dropout in any layer. Both of our models did not perform well and suffered from over fitting. CNN suffered from over fitting badly as compared to BiLSTM model. The training-validation loss with epoch for both models are given in 5 and 6.

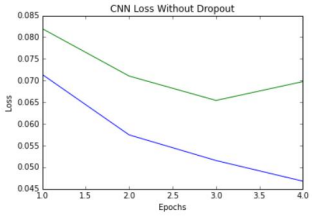

Figure 5: Loss of CNN without dropout

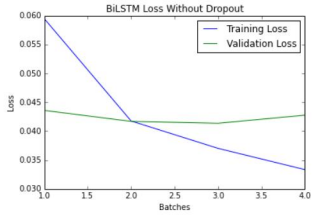

Figure 6: Loss of BILSTM without dropout

4.5 ROC Curves and AUC Score

ROC-Curve is a plot of true positive rates and the false positive rates for every possible classification threshold. It is widely used for visualizing the performance of a binary classifier. The main reason behind using ROC curves and AUC score for evaluating this model is the unequal class distribution of toxicity labels. In binary classification, the class prediction for each instance is often made based on a continuous random variable **X**, which is a score computed for the instance. In our model, it is the sigmoid distribution output for each of the toxicity labels. Given a threshold parameter **T**, the instance is classified as "positive" if $\mathbf{X} > \mathbf{T}$ and "negative" otherwise. X follows a probability density $f_1(x)$ if the instance actually belongs to class "positive" and $f_0(x)$ if otherwise. Therefore, true positive rates is given by $\int_T^{\inf} f_1(x)dx$ and false positive rate is given by $\int_T^{\inf} f_0(x)dx$. ROC curve plots the true positive rates and false positive rates with T as the varying parameters [3]. AUC score is the simply percentage of ratio of the area under ROC curve to the total area of the plot. A random predictor will have an AUC score of 0.5 and the model having good classifier will have an AUC score close to 1.

4.6 Training Time

Almost all variants of two models ran for 4-5 epochs. We used EarlyStopping feature on Keras [1] monitoring the validation accuracy and loss with a patience of 1. The BiLSTM model's loss stabilizes around epoch 3 and CNN loss stabilizes around epoch 4. If we train our model in excess of epoch 4, it starts over fitting. As we can see in 7 and 8, both models starts to experience overfitting after epoch 4 which leads to early stopping. Since the dataset is only 150K and given the lower complexity of the models, it takes around 45 minutes to train the BiLSTM model on 2.7 GHz Intel Core i5 Macbook and 30-34 minutes to train the CNN Model.

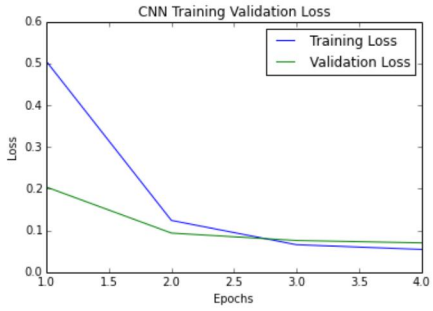

Figure 7: CNN Loss

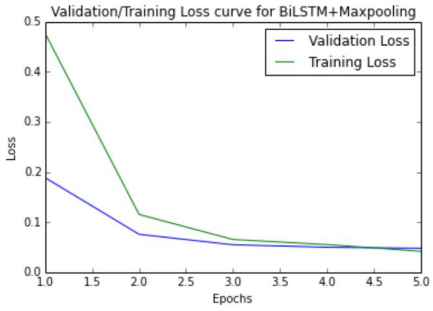

Figure 8: BiLSTM Loss

4.7 Results:

4.7.1 Quantitative Analysis

We primarily evaluate the loss and AUC metric on our validation data and the hidden data for the metrics.

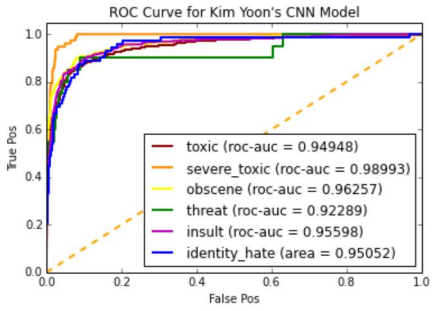

Figure 9: ROC Curve for CNN Model

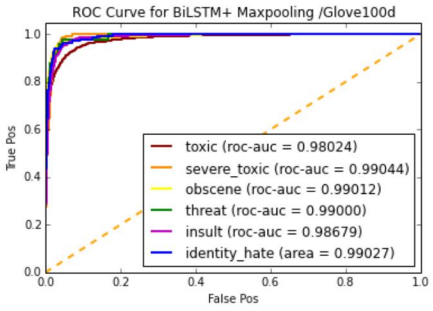

Figure 10: ROC Curve for BiLSTM Model

Table 3: Models Accuracy

Models	AUC Score(Hidden Test)	AUC Score(Validation Data)
BiLSTM+MaxPooling	0.9829	0.9859
CNN	0.96	0.958
Logistic Regression	0.9721	0.974

Our best model is the BiLSTM + MaxPooling model and it achieves 0.9829 in the hidden test set submitted to the Kaggle Leaderboard [2]. The training loss for our BiLSTM model is around 0.052763 and the validation loss is around 0.0666 using multi-labeled loss function. Respectively for CNNs, the training loss function is 0.0627 and the validation loss is 0.0754.

4.7.2 Qualitative Analysis

Insult Text Categorization Since insult is always directed to another person, both of the Neural Network suffered from classifying a comment as insult or non-insult whenever there is a presence of pronoun "We" or "I" in the sentence.

For example:
Gold Label: Insult Positive
Predicted Label for Insult: Negative
Sentence: if I want your stinky opinion I would have flushed three times pal!

[2]The winning submission has a score of 0.9885 and uses ensemble/blending of both non-neural and neural models with advanced text preprocessing.

Gold Label: Insult Positive
Predicted Label for Insult: Negative
Sentence:Get out my talk page you smelling wog!

Thus, we can conclude the insult category suffers from the errors if we have possesive pronouns like "my", "myself" , "I" and "we".

Foreign Language KeyWords Depicting Toxicity Both CNNs and LSTMs seems to be confused when the text input was foreign. We saw a lot of examples where the strong toxicity words were in other languages than English. Due to the sparsity of non-English emotion words in the training dataset, our model makes mistakes when we give text having non-english words.

CNN not working as desired As we know that ordering of the word plays a crucial value in the meaning of the sentence. However, since we are maxpooling on the CNN model, we do not differentiate the ordering of the words. That's our main intuititon why we think our CNN model suffered from more loss as compared to BiLSTM.

Nature of Conversation in Long Comments We only took the first 100 words of the comments as the input. We were partially correct in our reasoning since we saw the trend that toxic comments ends up in less than 50 words. However, we also found a new pattern where toxicity arises after long peaceful conversation. Hence, a new way to approach the problem will be select the first 50 words from the beginning and last 50 words from the end to capture the change in the toxicity of the sentence in the later half.

Using Name Entity Pairs For increasing the accuracy of toxic labels like insult, hate-speech and threat, we can leverage the Name Entity Pairs. If we augment the sentence with all the name entity pairs, the model can definitely do better in predicting insult, hate-speech and threat as these toxic labels are always directed to a person or a group.

5 Conclusion

Thus, we presented an experimentation of using deep learning models on combating internet toxicity. We had supposed that the CNN model will perform better than the Bidirectional LSTM model but it did not happen. The CNN based on [6] is fairly simple but it requires extensive use of the hyper parameter tuning. It might be the reason why it did not work out as expected. Our Bidirectional LSTM gives us the best result of 0.9820 AUC score.

A lot of future work can be done on this model. We can apply the idea of Hierarchial Network in the BiLSTM layer as proposed by Pappas and et al. in [12]. We did not try kfold validation and grid search on the CNN model. Use of kfold validation and grid search on the model can augment our accuracy and help us in fine tuning our hyper parameter. We can try experimenting with 2D MaxPooling after BiLSTM layer as proposed in [17].

Also, we can try augmenting the datasets by translating the comments into three-fours languages like German, Spanish, French and training our model collectively on these datasets. The winning solution did this and it would be interesting to see how our BiLSTM and CNN will perform with this change.

6 Acknowledgement

We would like to acknowledge our CS224n project mentor Abi See for great mentoring and rest of CS224n project for the great class. We would like to thank Denny Britz for his great blog post on different ways of implemnating CNN proposed by Kim.

References

[1] François Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

[2] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[3] Wikipedia contributors. Receiver operating characteristic — wikipedia, the free encyclopedia, 2018. [Online; accessed 22-March-2018].

[4] Olga Jubany and Malin Roiha. Backgrounds, experiences and responses to online hate speech: A comparative cross-country analysis. *Universitat de Barcelona*, abs/1504.02305, 2016.

[5] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[6] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[7] Haewoon Kwak, Jeremy Blackburn, and Seungyeop Han. Exploring cyberbullying and other toxic behavior in team competition online games. *CoRR*, abs/1504.02305, 2015.

[8] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[9] Zickuhr Price-Feeney Lenhart, Ybarra. Online harassment, digital abuse, and cyberstalking in america. *DataSociety Research*, 11.21.16(10):2–3, 2016.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[11] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[12] Nikolaos Pappas and Andrei Popescu-Belis. Multilingual hierarchical attention networks for document classification. *CoRR*, abs/1707.00896, 2017.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[15] Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. Predicting polarities of tweets by composing word embeddings with long short-term memory. pages 1343–1353, 01 2015.

[16] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. *CoRR*, abs/1610.08914, 2016.

[17] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *CoRR*, abs/1611.06639, 2016.