Structure-based Hate-speech detection

Ananya Arun (20171019), Vijayraj Shanmugaraj (20171026), Sumit Bhuin (2019201034), Virat Mishra (2019201033)

Abstract

The main objective of our project is "Structure-based hate speech detection". Traditional methods for hate speech detection use tons of training data to mine the hateful structure but due to disproportionate use of different terms, they are prone towards learning bias against specific objects, personalities or groups. Idea is to propose a method that takes into account the grammatical structure of the sentence to predict hatefulness.

1 Introduction

Hate speech is commonly defined as any communication that disparages a person or a group on the basis of some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics. With the rise of social media and user generated content, detecting and classifying hate speech is becoming quite important. To automate the process of hate-speech detection, we look at a system that tries to utilize the grammatical structure of the system as features in order to classify a sentence as hate-speech or not, in order to avoid bias towards certain named entities.

2 Dataset used

The dataset being used is text extracted from Stormfront, a white supremacist forum. (https://github.com/Vicomtech/hate-speech-dataset). The dataset has 10495 sentences labelled either as hate or non-hate. We also used a second dataset (https://github.com/t-davidson/hate-speech-and-offensive-language) which was used in the paper *Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber*. 2017. *"Automated Hate Speech Detection and the Problem of Offensive Language."* ICWSM. This consists of 25,297 tweets which have been labelled whether the tweet comes under hate-speech, uses offensive language or falls under neither of the two categories.

3 Experiment on Stormfront Dataset

3.1 Preprocessing

For the first part of the experiment we removed all stopwords, used a part-of-speech tagger and we fused every PoS tag with the word as shown, and was used :

Ex. The only thing more disgusting than a White woman with a groid is a White woman who drags her White child into the filth with her. The words 'black' and 'white' were removed for models that used sentence structure in the form of PoS tags to avoid the bias we would like to avoid in the first place.

```
TheDT thingNN disgustingVBG womanNN groidNN womanNN dragsVBZ childNN filthNN
```

Figure 1: A sample of a preprocessed sentence

We also used an alternative version of the above pre-processing where we stemmed the words before fusing them with their PoS label in the dataset before applying a bag of words. For the LSTM we fed in all three versions (N, P, P+S (See table for full forms of preprocessing)) of the sentences, and for the CNN, we used One-Hot encoding over the above preprocessing methods to feed into the CNN.

3.2 Models used in the first phase and issues with approach

Initially, we used the following baseline models: Naïve Bayes, Logistic Regression, Decision trees, Convolutional Neural networks and LSTM. We used the CountVectoriser to construct features for the first four, and One-hot encoding with padding for the CNNs and LSTMs. Our initial results were as follows:

Table 1. First experiment results. (N - normal (without PoS tags, only stopwords removed), P – with PoS tags, P+S – Stemming + PoS tagging)

Model Used	Accuracy	Precision	Recall	F1-score
Naïve Bayes (P)	0.879	0.778	0.554	0.567
SVM (P)	0.872	0.973	0.503	0.473
Logistic Regression (N)	0.876	0.72	0.609	0.636
Decision Trees (P+S)	0.854	0.65	0.609	0.624
CNN (Single convolution layer) (P+S)	0.856	0.447	0.199	0.276
CNN (Complex model) (P+S)	0.825	0.368	0.372	0.370
LSTM (P+S)	0.828	0.337	0.355	0.293
BERT (N)	0.904	0.913	0.892	0.855

Although the accuracy seems to be very high, the precision, recall and the F1-score seemed to be terrible. The reason for this was the underlying problem of all skewed datasets. There are 1437 sentences labelled as hate-speech, whereas there are 9507 sentences that were not hate-speech. We analyse their confusion matrices of the SVM and the LSTM:

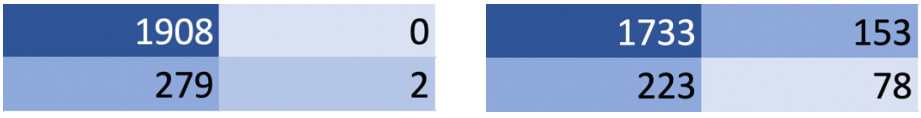

Figure 2a) and 2b): The confusion matrices of the SVM (left) and the LSTM (right)

The skew in the class balance forces the model to predict most of the sentences as non-hate as they consider the hatespeech part of the dataset as an outlier as it consists of about 14% of the dataset. (Note: The BERT was an exception, and it performed extremely well, although it was a skewed dataset.) To take care of this we oversampled the dataset using the SMOTE (Synthetic Minority Oversampling Technique) method, which synthesizes data of the minority class from minority class data points by adding some noise to them, and bringing in more minority class data points. By testing it on the SVM, the results were as follows: the accuracy and precision went down to 0.8, however, the recall and F1-score leaped up to 0.809.

We also additionally tried N-gram pre-processing along with Logistic Regression. The results were as follows:

Table 2. First round of experiments with N-Gram and LR (Using just PoS tags)

Model Used	F1-Score	Recall	Accuracy	Precision
Unigram/LR	0.835	0.587	0.862	0.701
Bigram/LR	0.825	0.555	0.865	0.733
3-gram/LR	0.818	0.541	0.864	0.733
4-gram/LR	0.816	0.536	0.863	0.732

The N-Gram models show similar setbacks as well, with poor recall, but as far as precision and F1-score are concerned, they seem to perform better than their vanilla counterpart in the first run of experiments/

In the case of CNNs, various possible models when tried out gave extremely sharp spikes in validation loss. This was after the SMOTE technique was applied to the input. This dataset seems to be extremely skewed, and additionally, SMOTE was not able to introduce much noise into the minor class data points in order to avoid the skewing problem, and hence simpler model seemed to resort to overfitting. On adding more dropout layers and increasing the number of epochs, we were luckily able to achieve one steep drop of validation loss towards the end (from nearly 4.0 to 2.5). Increasing the number of epochs further increased the validation loss, hence, running the model for more epochs led to overfitting.

An additional problem was the small size of the dataset. Hence, getting a model that would not overfit to a small number of data points was a really tough challenge faced during the experiment.

```
model = Sequential()
model.add(Embedding(vocab_size, 32, input_length=sen_len))
model.add(Conv1D(32, 3, padding='same', activation='relu'))
model.add(MaxPooling1D())
model.add(Conv1D(64, 2, padding='same', activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=4)
```

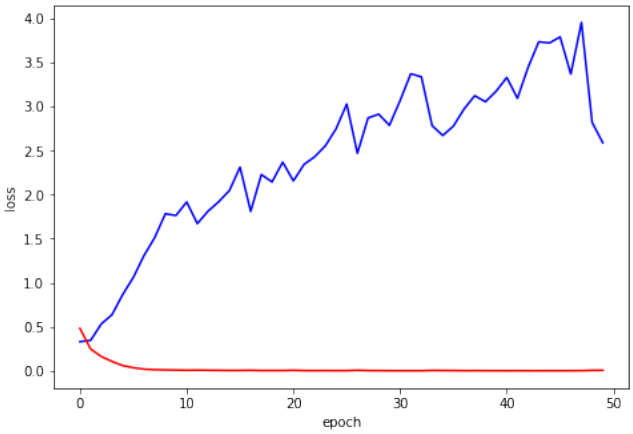

Fig 3a) and 3b): The above model was trained in batches of 50 points for 40 epochs. Although the model did show a few dips in validation loss, the trend is clearly upward (with the exception of the steep drop towards the end).

Similarly, the SMOTE method was tried on the LSTM as well, but this time, only censoring named entities and retaining the sentence structure as it is. The final results were as follows (Since the LSTM takes into account long term dependencies, we decided to feed in the actual sentences from the datset into the LSTM):

Table 3. Second round of experiments with SMOTE (U – un-preprocessed sentences)

Model Used	Accuracy	Precision	Recall	F1-score
SVM (P)	0.800	0.800	0.809	0.809
3-gram/LR (P)	0.829	0.833	0.828	0.828
CNN (Larger model) (P)	0.797	0.816	0.771	0.793
LSTM (U)	0.871	0.865	0.879	0.872
LSTM (with attention) (U)	0.877	0.897	0.851	0.873

Table 4. First round of experiments with N-Gram and LR (Using just PoS tags)

Model Used	F1-Score	Recall	Accuracy	Precision
Unigram/LR	0.821	0.822	0.823	0.830
Bigram/LR	0.827	0.827	0.828	0.833
3-gram/LR	0.828	0.828	0.829	0.833
4-gram/LR	0.826	0.826	0.826	0.830

Clearly SMOTE has improved the scores of all the models, and hence it seems to have fixed the issues faced in the first experiment.

Note: 1) We did not apply TreeLSTM on the Stormfront dataset because the dataset did not have additional dependency parse taggings that were present in the second tweets dataset. 2) (BERT wasn't tested here because it took extremely long to run, and hence wasn't tried out for the second dataset either.

4 Experiment on Tweet Dataset

4.1 Preprocessing

For the second dataset, before the preprocessing done in the first dataset, we additionally removed hashtags, '@' mentions and links to avoid any bias due to the same, since hashtags and mentions usually refer to, or are related to some particular named entity.

4.2 Details about the dataset

The dataset has 19,190 sentences that are neutral, 1430 that are hate-speech and 4163 containing offensive language. This dataset was obtained by making users tag a dataset on a crowdsourcing platform called CloudFlower. Each data point contains 5 columns:

- *count* : number of CrowdFlower users who coded each tweet
- *hate_speech* : number of CF users who judged the tweet to be hate speech.

- *offensive_language* : number of CF users who judged the tweet to be offensive.
- *neither* : number of CF users who judged the tweet to be neither offensive nor non-offensive.
- *class* : class label based with the majority of tags from the users who have tagged the tweet. (0 - hate speech, 1 - offensive language, 2 – neither)

This dataset was picked up because it contains more number of data points than the Stormfront dataset, and the source this was picked up from contained annotations for dependency parsing of the sentence, and would help us to experiment with Tree-LSTMs.

4.3 Models used in the experiment

We ran the same models that we ran on the Stormfront dataset. The results are as follows. (We also ran the n-gram models on this dataset as well)

Note: We did run BERT on the 2nd model due to lack of time, since the training takes a lot of time to run.

Table 5. First experiment results. (Without oversampling) (N - normal (without PoS tags, only stopwords removed), P – with PoS tags, P+S – Stemming + PoS tagging)

Model Used	Accuracy	Precision	Recall	F1-score
Naïve Bayes (P)	0.86	0.751	0.518	0.548
SVM (P)	0.84	0.57	0.344	0.312
Logistic Regression (P)	0.904	0.735	0.687	0.699
Decision Trees (P+S)	0.884	0.68	0.668	0.673
CNN (Single convolution layer) (P)	0.857	0.672	0.638	0.654
CNN (Complex model) (P)	0.775	0.259	0.334	0.291
LSTM (P)	0.685	0.337	0.355	0.293
Bi-LSTM + Attention (P)	0.874	0.680	0.651	0.665

Table 6. First round of experiments with N-Gram and LR (Using just PoS tags)

Model Used	F1-Score	Recall	Accuracy	Precision
Unigram/LR	0.835	0.587	0.862	0.701
Bigram/LR	0.825	0.555	0.865	0.733
3-gram/LR	0.818	0.541	0.864	0.733
4-gram/LR	0.816	0.536	0.863	0.732

Not oversampling clearly messes with the precision, recall and F1-score. The same problems as the first experiment still show up. Moreover, this problem is a multiclass classification problem. The ratio of sentences in the dataset (in the order neither : offensive language : hate-speech) is 13.4 : 2.9 : 1, and hence the problem persists for both the minority classes.

Hence we went on to use SMOTE to oversample this dataset as well. The results are as follows:

Table 7. Second round of experiments with N-Gram and LR (Using just PoS tags) and SMOTE

Model Used	F1-Score	Recall	Accuracy	Precision
Unigram/LR	0.835	0.834	0.834	0.854
Bigram/LR	0.774	0.781	0.782	0.827
3-gram/LR	0.771	0.779	0.780	0.824
4-gram/LR	0.767	0.776	0.777	0.823

Table 8. Second experiment results. (With SMOTE) (N - normal (without PoS tags, only stopwords removed), P – with PoS tags, P+S – Stemming + PoS tagging)

Model Used	Accuracy	Precision	Recall	F1-score
SVM (P)	0.765	0.791	0.765	0.760
Logistic Regression (P)	0.861	0.865	0.860	0.861
CNN (Complex Model) (P)	0.674	0.669	0.673	0.671
LSTM (P)	0.685	0.686	0.685	0.685
Bi-LSTM + Attention (P)	0.683	0.681	0.682	0.681

Again, the use of Over-Sampling in datasets solved the issue of skewed datasets and increased the precision and recall of the model, thereby increasing the F1-score as well. Additionally, we also implemented the TreeLSTM for this dataset, which incorporates non-linear semantic features such as Dependency Trees into our model. While using TreeLSTM, we are able to use the Dependency Parse Trees as features in our model. The difference between the standard LSTM unit and Tree-LSTM units is that gating vectors and memory cell updates are dependent on the states of possibly many child units. Additionally, instead of a single forget gate, the Tree-LSTM unit contains one forget gate f_{jk} for each child k. This allows the Tree-LSTM unit to selectively incorporate information from each child.

Table 8. TreeLSTM experiment results.

Accuracy	Precision	Recall	F1-score
0.896	0.893	0.896	0.895