Implementation of a Deep Learning Based Approach for Abuse Detection

Mahdi Rahimi
marahimi@email.arizona.edu
https://github.com/mahrahimi1/CSC665_Project

In this project, I have implemented an approach which to some extent is based on the paper by Founta et al [1]. The paper proposes a deep learning approach to detect abusive language on social media, e.g. Twitter. They test their approach on different datasets each of which is for a certain kind of abusive language and report a successful results. All of the datasets are from Twitter. Their deep learning approach is comprised of training an LSTM model. They also gather some metadata from Twitter about the content of the tweets and the authors and their circle of friends on Twitter. Then, they merge this information with the output of the LSTM model and use it for classification.

In addition to the LSTM model, I have added two more models to the project: one hierarchical CNN model from [2] and one RNN-based autoencoder model. Furthermore, I have not included the metadata approach in my implementation since I don't have the metadata and I could not obtain it from the authors of that paper. In the rest of the report, the details of the datasets and each of the deep learning based approaches are explained. Furthemore, the results of the experiments are demonstrated.

Datasets:

Some of the datasets in the paper didn't contain the tweets themself and they only contained tweet IDs. As a result, I filled out a short application for a Twitter developer account. After I was approved by Twitter, I used a python program and a Twitter API to retrieve the text of the tweets using the tweet IDs. After the text of the tweets were obtained, the datasets were ready to be used in the project. In this project, I used three datasets:

1. **Cyberbullying Dataset** [3]

The first dataset was collected for the purpose of detecting two instances of abusive behavior on Twitter: cyberbullying and cyberaggression. In addition to a baseline, the authors collected a set of tweets between June and August 2016, using snowball sampling around the GamerGate controversy, which is known to have produced many instances of cyberbullying and cyber-aggression. The data was labeled via crowdsourced workers into one of four categories: 1) bullying, 2) aggressive, 3) spam, or 4) normal. The authors are careful to differentiate between aggressive and bullying behavior. An aggressor was defined as "someone who posts at least one tweet or retweet with negative meaning, with the intent to harm or insult other users" and a bully was defined as "someone who posts multiple tweets or retweets with negative meaning for the same topic and in a repeated fashion, with the intent to harm or insult other users." Furthermore, the authors of [1] removed samples with label of spam as they can be handled with more specialized techniques. I followed the same approach.

2. Offensive Dataset [4]
The second dataset is focused on racism and sexism. Collected over a two month period, the authors manually searched for common hateful terms targeting groups, e.g., ethnicity, sexual orientation, gender, religion, etc. The search results were narrowed down to a set of users that seemed to espouse a lot of racist and sexist views. After collection, the data were preprocessed to remove Twitter specific content (e.g., retweets and mentions), punctuation, and all stop words except “not.” The tweets were labeled as racist or sexist or none.

3. Hate Dataset [5]
This dataset contains tweets characterized as hateful, offensive, or neither. Here, hate speech is defined as language that is used to express hatred, insult, or to humiliate a targeted group or its members. Offensive language is less clearly defined as speech that uses offensive words, but does not necessarily have offensive meaning. Thus, this dataset makes the distinction that offensive language can be used in context that is not necessarily hateful.

Methodology:

Figure 1 demonstrates the methodology I used in this project. After preprocessing of the data, a pre-trained word embedding is used to convert word vectors to word embeddings. For this project I used the Twitter-based version of GloVe word embedding (2B tweets, 27B tokens, 1.2M vocab, uncased). It comes with 25d, 50d, 100d, and 200d vectors. In this project I used 200d vector.

After that, I used three different deep learning based approaches for feature extraction which are explained in more detail in the next section. Finally, a multi-layer perceptron (a.k.a fully connected neural network) is used for a multi-class classification. The network contains one hidden layer with 20 neurons and a final softmax layer with the size of the number of the classes.

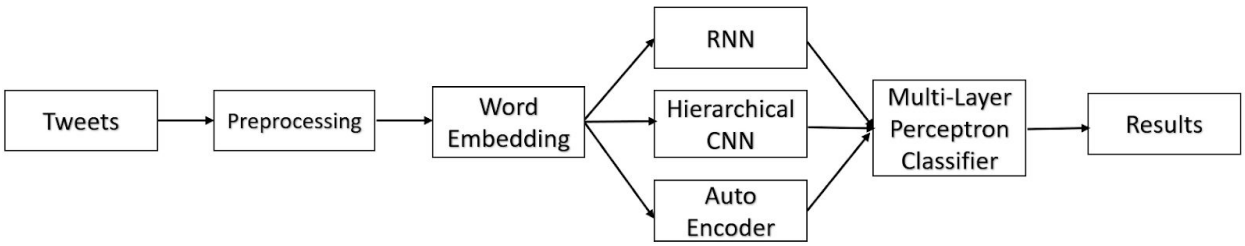

Figure 1: Steps taken in our approach

1. RNN Model:

After word embedding vector is obtained, we feed it into a Long Short-Term Memory (LSTM) model for feature extraction. The LSTM will learn underlying characteristics of the tweets and creates a vector which represents a tweet. This vector is then used for classification. The dimension of the output of the LSTM unit is 100. Figure 2 demonstrates the LSTM model.

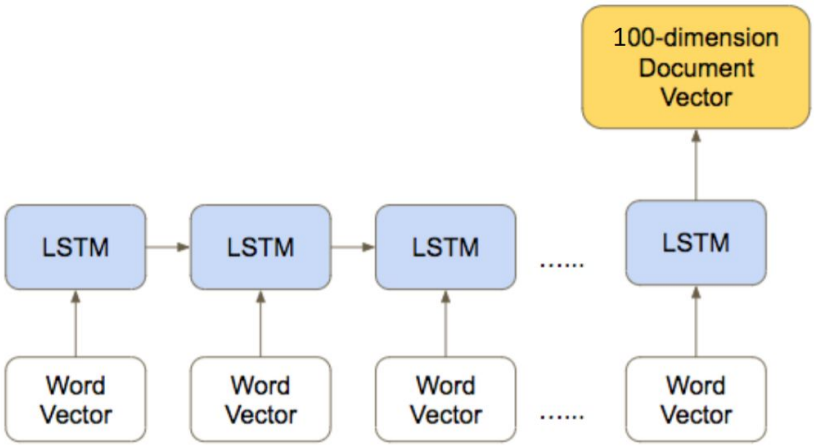

Figure 2: The architecture for RNN model for feature extraction

2. Hierarchical CNN Model:

This approach is inspired from the document modeling method presented in [2]. After word vectors are obtained using word embedding, three convolutional filters are applied on sentences to obtain unigram, bigram, and trigram features for each tweet. To extract a such n-gram feature, a convolutional filter of size nxE is applied on tweets where E denotes the dimension of embedding matrix (200). We use feature maps of size 200 for each of the n-grams where n = 1, 2, 3. Therefore, the convolutional filter applied is of size 200xnxE. We add a bias of size 200 to the output of the filter. Hence, we obtain three vectors of size 200x(W-n+1)x1, n = 1, 2, 3 for each tweet where W denotes the maximum number of words in a tweet. Also, in order to introduce nonlinearity, we apply the Rectified Linear Unit (ReLU) function to the vectors as well.

Next, we apply max pooling to these vectors to down-sample them to 200x1x1 vectors and then we flatten them to vectors of size 200. Finally, we concatenate the three n-gram vectors to obtain a vector of size 600 representing a tweet. Figure 3 demonstrated this approach.

3. RNN-based AutoEncoder Model:

An autoencoder attempts to learn the underlying features of a given input by trying to learn how to regenerate it as output. However, its intermediary layers are denser than the given input. Therefore, it learns the important feature of the input and store it in those layers. After that, we can use the “encoder” part of autoencoder for feature extraction.

In this project, I used a RNN-based autoencoder which is demonstrated in figure 4. The dimension of the output of both of the LSTMs is 100. It is noteworthy to mention that the word embedding vector is the input of the autoencoder. Therefore, it is outside of the trainable model and therefore the embedding matrix is not trained in this case.

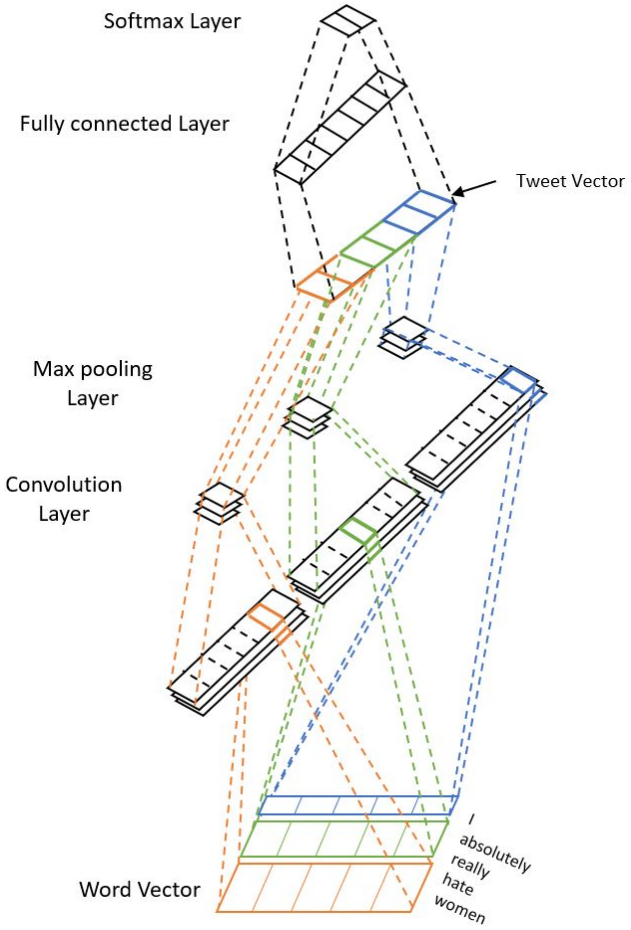

Figure 3: The architecture of the hierarchical CNN model inspired from the method presented in [1]

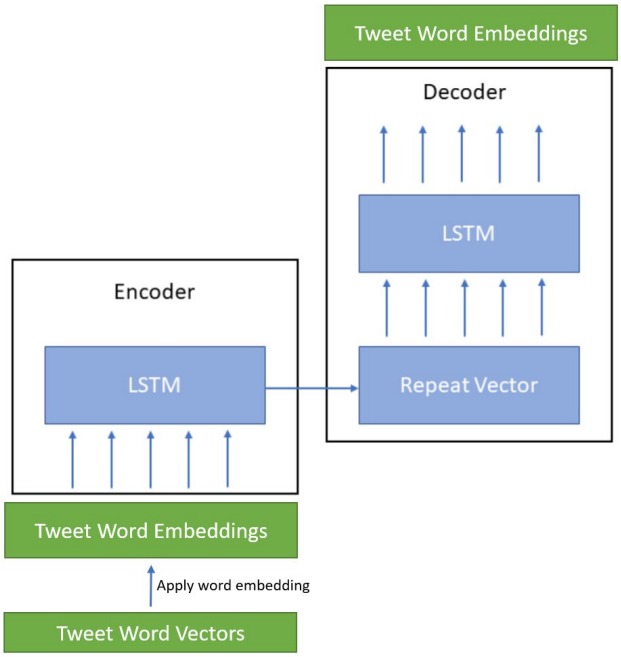

Figure 4: The RNN-based autoencoder used in this project consisted of an encoder and a decoder

After the autoencoder is trained, we use its encoder part for extracting features from the word embedding vector of the input data. This is demonstrated in figure 5.

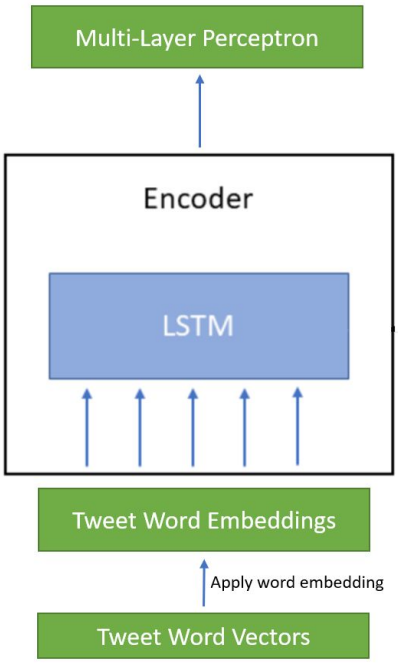

Figure 5: Using the “encoder” part of previously trained autoencoder for feature extraction.

Experiments and Results:

The successful performance of the explained models on each of the datasets are demonstrated below. The models were trained using stochastic gradient descent with batch size of 100. Furthermore, in order to avoid overfitting early stopping was used. In order to do that, the last 5 consecutive epochs are monitored; if no further improvement is observed in validation loss, then the training will end.

Model	Training Accuracy	Validation Accuracy	Test Accuracy
RNN	97.6	98.5	98.2
Hierarchical CNN	99.8	97.6	98.2
AutoEncoder	97.5	97.6	97.8

Tabel 1: The performance of the models on Cyberbullying dataset

Model	Training Accuracy	Validation Accuracy	Test Accuracy
RNN	98.3	83.9	83.2
Hierarchical CNN	99.9	89.7	90.0
AutoEncoder	83.5	83.6	84.7

Tabel 2: The performance of the models on Hate dataset

Model	Training Accuracy	Validation Accuracy	Test Accuracy
RNN	99.3	85.3	84.7
Hierarchical CNN	99.4	86.2	86.2
AutoEncoder	80.6	82.0	82.2

Tabel 3: The performance of the models on Offensive dataset

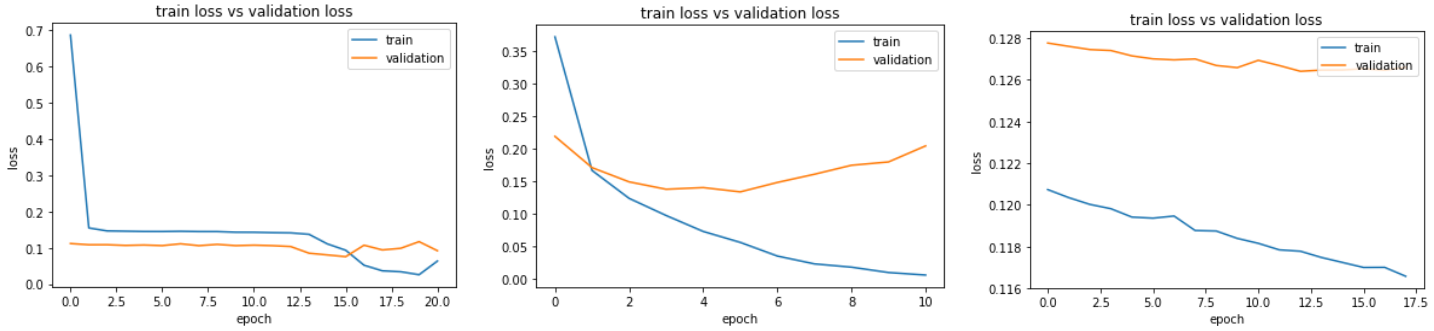

Figure 6: Training and validation loss for RNN model, hierarchical CNN model and Auto Encoder model on Cyberbullying dataset (from left to right)

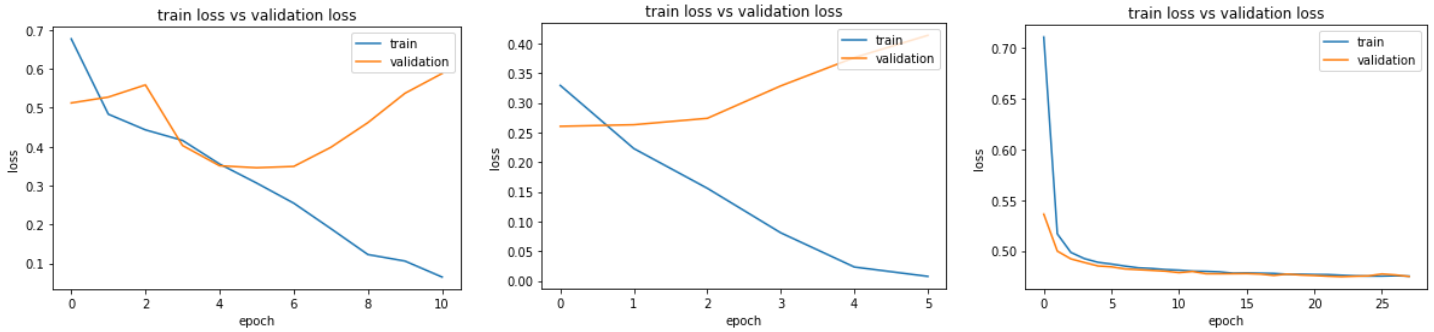

Figure 7: Training and validation loss for RNN model, hierarchical CNN model and Auto Encoder model on Hate dataset (from left to right)

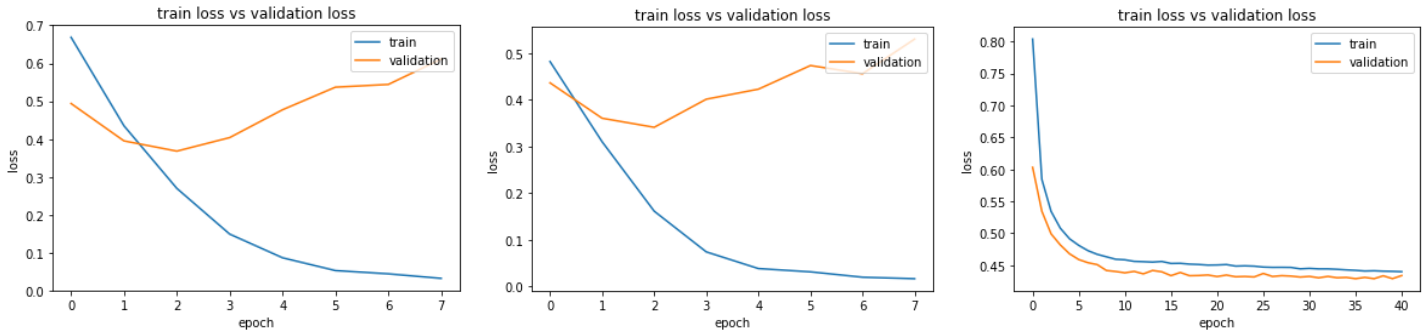

Figure 8: Training and validation loss for RNN model, hierarchical CNN model and Auto Encoder model on Offensive dataset (from left to right)

References:

[1] A.-M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis. A Unified Deep Learning Architecture for Abuse Detection. arXiv preprint arXiv:1802.00385, 2018.

[2] N. Majumder, S. Poria, A. Gelbukh, and E. Cambria, ‘‘Deep learning-based document modeling for personality detection from text,’’ IEEE Intell. Syst., vol. 32, no. 2, pp. 74–79, Mar./Apr. 2017

[3] Chatzakou, D.; Kourtellis, N.; Blackburn, J.; De Cristofaro, E.; Stringhini, G.; and Vakali, A. 2017. Mean birds: Detecting aggression and bullying on twitter. In 9th ACM WebScience.

[4] Waseem, Z., and Hovy, D. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In SRW@ HLT-NAACL, 88–93.

[5] Davidson, T.; Warmsley, D.; Macy, M.; and Weber, I. 2017. Automated hate speech detection and the problem of offensive language. arXiv preprint arXiv:1703.04009.