Natural Language Processing with Deep Learning Final Report: Automated Hate Speech Detection with LSTM Model

Xiaoyan HU
x.hu@mpp.hertie-school.org

Abstract

Social networks exert significant influence in spreading and shaping public opinions. Due to the harmful social impact of hate speech content, leading social network companies have designed policies to remove those content. Currently, one conventional way of keeping the platform "clean" is to hire human moderators according to a feature from online media. While the contractors are working under enormous pressure and invest many efforts, their efficiency cannot compare to the explosive increase of hateful content on social media. Furthermore, the moderators must cope with the blatant exposure to traumatic images, out-cry racism, violent xenophobia, and other aggressive content, which introduces extreme mental pressure upon the persons. Thus, this manual labour is not sustainable and should be replaced by more efficient and less costly technological approach.

Recent years, researchers dive into an alternative to manual moderation: automated hate speech detection. The Natural Language Processing (NLP) and Machine Learning (ML) communities worked together closely and came up with classical methods for text classification such as Naive Bayes, Logistic Regression and Random Forest. The improvement of optimization techniques made it possible to explore neural networks and develop deep learning methods based on that. Some of the most popular approaches include Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), typically Long Short-Term Memory network (LSTM) (Shanita Biere,2018) [5].

This team project aims at understanding the mechanism behind automated hate speech detection by building a deep-learning detector with the dataset from the open web. The project applies a widespread RNN approach for classification task on a Twitter dataset with Pytorch framework and compares the result with a baseline model using the older technique called Naïve Bayer. The results from both models show that LSTM achieves a rather high accuracy rate on the validation set (84.9%) while the baseline model yields an accuracy of 79.2%. an accuracy of 79.2%. However, the author notes that the LSTM model probably suffers from over-fitting. During the implementation of this project, the author learns more about the technical barriers in front of a more powerful classifier and the non-technical challenge faced by algorithms.

1. Introduction[1]

While the rapid popularity of social networks has enlarged the cyberspace for voicing out individual opinions, mainstream platforms such as Twitter, Facebook, and Instagram have also facilitated the promotion of hateful speech and the incitement of hate crime. This cyber violence often targets individuals or group of individuals on the ground of their ethnicity, nationality, gender, sexual orientation, colour or religion. Discrimination as such is no news to the policy domain. For the past 50 years, the international communities have made progress in combating intolerance via establishing global conventions or implementing domestic laws (Pálmadóttir and Kalenikova, 2018) [20]. However, what distinguishes cyber hate speech and conventional ones is the speed of its spread. Thus, besides existing legal frameworks, we also have to come up with new instruments to curb the further spread of hateful content.

Currently, legislators believe that platforms should take the primary responsibilities of cracking down hate speech. One common practice of keeping the platform "clean" is to hire human moderators. These contractors are working under enormous pressure and invest many efforts, but their efficiency cannot compare to the explosive increase of hateful content on social media. Furthermore, the moderators must cope with the blatant exposure to traumatic images, out-cry racism, violent xenophobia, and other aggressive content, which introduces extreme mental pressure upon the persons (Verge, 2018) [17]. Therefore, this manual labour is not sustainable and should be replaced by more efficient and less costly technological approach.

Automated detection using a computer algorithm is one of the potential alternatives to manual categorization. There

[1]GitHub page link:https://github.com/Xiaoyan-Adele/Automated-Hate-Speech-Detection

is ample research into automated hate speech detection over the last decade. It makes this topic a very dynamic field, and researchers have yielded an abundance of achievement. Many state-of-art models evolved from statistics-based traditional natural language processing technics and deployed recent advancement in the deep neural network. In order to understand the mechanism behind automated hate speech detection, this project aims at applying one of the popular classification methods: Long Short Time Memory networks (LSTM) with the Python-based open source machine learning library called Pytorch (Paszke et al. 2016). To understand power of LSTM, the author compares the result with a baseline model using an older technique called Naïve Bayer. The results from both models show that LSTM achieves a rather high accuracy rate on the validation set (84.95%) while the baseline model yields an accuracy of 79.2%. an accuracy of 79.2%. However, the author notes that the LSTM model probably suffers from overfitting.

Although this project demonstrates that the LSTM model can obtain decent accuracy rate, it is still far from the moon that many scientists wish to reach. The scientific community still faces challenges in differentiating context, sarcasm and systematic detection evasion (MacAvaney et al., 2019) [14]. Besides technical constraints, there is also an ethical dilemma in automated hate-speech quarantine. Researchers and policymakers are still exploring the appropriate balance between freedom of speech and cyber authoritarianism (Ullmann and Tomalin, 2019) [21]. We shall discuss those unsolved tasks in the final part of this report.

2. Related Work

Research on automated hate speech detection first differentiates each other according to what it defines to be the "targeted hate speech". European Commission against Racism and Intolerance (ECRI) defines hate speech as the forms of expressions which spread, incite, promote or justify hatred, violence and discrimination against a person or group of persons for a variety of reasons [1]ite. Such definition is too abstract to set standards for all hate speech detection model. In one of the Microsoft research, they sorted the ECRI's overarching concept on hate speech into four types based on the targets of aggression: 1) physical threat; 2) sexual threat/aggression; 3) identity threat/aggression; 4) non-threatening aggression (2017) [11]. This typology is far from being the industry standard for all classification models. The lack of a unified definition of hate speech introduces interoperability problems to detection models. Each model is formulated only according to its own understanding of the match point (MacAvaney, 2019) [14]. Since this project is not going to collect and label the dataset by itself, we can only accept the definition of the available dataset as given.

However, the lack of universal definition did not prevent the detection model from improving. Like the field of sentiment analysis, automatic hate speech identification research has stemmed from two types of approaches: those based on the use of lexicons and those based on machine learning (Arango et al., 2019) [2]. This literature review focuses on the latter. Classical machine learning methods are based on statistics. They started with a feature selection of preprocessed content by using the bag-of-words or TF-IDF. Apart from those two distributional features, classic machine learning also uses word embedding (e.g. word2vec). However, assigning a vector to a word is not restricted to classic machine learning. Some deep learning architectures also adopt word embedding as the initial step (Kuncoro A, 2018) [12]. Then many classic detection approaches put the generated features into commonly used models such as Naïve Bayes, Support Vector Machine (SVM) and Logistic Regression. Davidson et al. proposed a state-of-the-art feature-based hate speech classification model that incorporates distributional TF-IDF features, part-of-speech tags, and other linguistic features using support vector machines (2017) [7]. As the neural network is gaining trend, these classic approaches are now customarily served as the baselines for more advanced machine learning techniques, but it is not entirely abandoned. MacAvaney et al continued this path and proposed a multi-view SVM approach that achieves near state-of-the-art performance while being more straightforward and producing more easily interpretable decisions than neural methods (2019) [14].

The biological structure of brains inspired the concept of a neural network. In computation, the basic unit of a neural network is the neuron, often referred to as a node. It receives input from other neuron and computes an output to pass on. Neural networks often consist of three categories of neuron layers: input, hidden and output. Back propagation is one of the most common methods to train a neural network. This method follows the chain rule and calculates the loss function of each parameter in the network. It is the loss function that the neural network learns to minimize (Liu, 2018) [13].

There are two main types of deep neural network architectures when handling natural language processing tasks, namely the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN). Proposed in 1998 by LeCun et al., CNN utilizes layers with convolving filters that are applied to local features. Kim (2013) trained a simple CNN model for sentence classification and received excellent results on multiple benchmarks [10]. In a reproducibility work, Zimmerman et al. (2017) improved the CNN with what they called an ensemble method, combining the decisions of ten convolutional neural networks with different weight initializations [25]. In a comparative study, Yin et al. (2017characterized the difference between the two as "hierarchical (CNN) vs sequential (RNN)" [24]. The main fea-

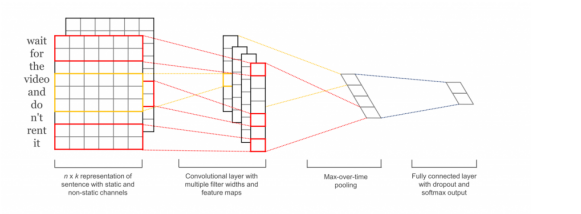

Figure 1. the architecture of CNNs quoted from Y. Kim (2014) Convolutional Neural Networks for Sentence Classification

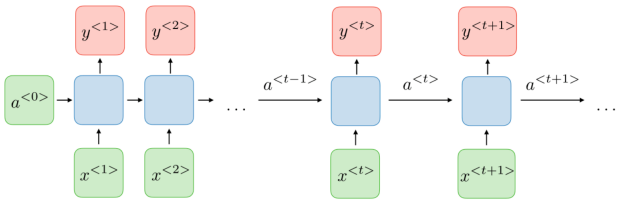

Figure 2. the architecture of a traditional RNN quoted from Stanford Cheatsheet page for class CS 230 – Deep Learning designed by its teaching assistant Afshine Amidi

ture of RNN is that this network setting emphasizes sequential computation, meaning that each node performs the same task for every element in the sequence, with the output dependent on the previous result (Liu, 2018) [13]. Manolescu et al. (2019) applied Long Short Term Memory (LSTM) on hate speech tweets in English and Spanish [15]. In the literature review, we noted that there are conflicting views about the performance between CNN and RNN (Tang et al. 2015; Dauphin et al., 2016) [6]. Many past pieces of research did not rely exclusively on one method between CNN and RNN. They often investigated both approaches and fine-tuning some parameters in order to beat the state-of-art methods (Badjatiya et al., 2017) [3].

3. Proposed Method

While there are many publicly available classifications models, this project focused on implementing a state-of-art neural network deep learning detection method. Both CNNs and RNNs yield promising results, and their performance difference often depends on the choice of the dataset, embedding methods, and the tuning of some parameters. Without a particular reason to convince us to opt for one instead of the other, we decided to reproduce the Long Short Term Memory (LSTM) on the publicly available dataset labelled by Davidson et al. (2017) [7].

3.1. Defining hate speech

Davidson et al. (2017) [7] defined hate speech as the language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group. It is important to note that

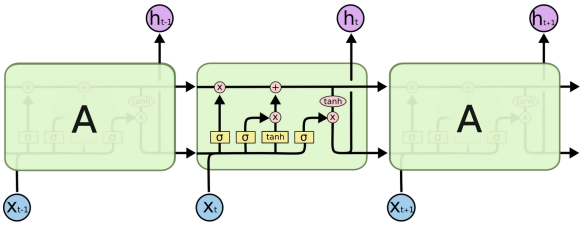

Figure 3. illustration of LSTM mechanism from Colah's Blog

Davidson's hate speech data does not include some highly contextual and frequently used phrase such as "b*tch" or f*g even though they are prevalent on social media.

3.2. Deep Learning Model: Long Short Term Memory (LSTM)

RNN models were made in a way that allows it to connect previous information to the present task. The academia invested high hope for its performance. However, Bengio et al. pointed in 1994 that practical difficulties have been reported in training recurrent neural networks to perform tasks in which the temporal contingencies present in the input/output sequences span long intervals [4]. Three years after the publication of Bengio's paper, LSTM is designed as an answer to the drawbacks of the previous RNN models. It was introduced by Hochreiter Schmidhuber (1997) and was explicitly designed to avoid long term dependency problem. LSTM can learn to bridge minimal time lags in excess of 1000 discrete-time steps by enforcing constant error flow through constant error carousels within special units. Multiplicative gate units learn to open and close access to the constant error flow [9]. In a clear and detailed introduction of the LSTM model, Olah (2015) described the improvement of LSTM over RNN: "(although LSTM also has a chain-like structure like the traditional RNN), the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way." [18]

This the design of gates (nodes) with sigmoid neural net layer and a pointwise multiplication operation can be understood as memory cells. Those nodes give LSTM the ability to remove or add information to the cell. Information first goes through the forget gate layer. This gate decides what is unnecessary to be kept and passes the filtered results to the next stage. In the subsequent step, the layer decides what information to be updated. A tanh layer creates a vector of new information with the processed value. Eventually, the information is passed through an output layer and sent to the next module.

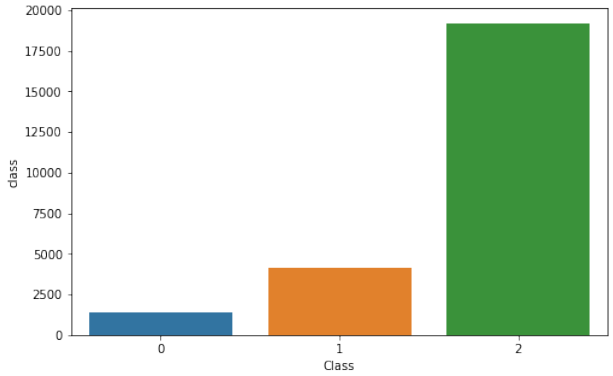

Figure 4. the distribution of each text class

tweet
!!! RT @mayasolovely: As a woman you shouldn't...
!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

Figure 5. Tweets before Preprocessing

tweet
[woman, complain, clean, house, man, always, t...
[boy, bad, hoe, place]
[ever, fuck, bitch, start, cry, confuse, shit]
[look, tranny]
[shit, hear, may, true, may, faker, bitch, tel...
[shit, blow, meclaim, faithful, still, fuck, h...

Figure 6. Tweets after Preprocessing

4. Experiments

Data: The data set this project used for training the LSTM model came from the research published by Davidson et al. (2017). He and the co-authors used the Twitter API we searched for tweets containing terms from the lexicon, resulting in a sample of tweets from 33,458 Twitter users. Then they extracted the timeline for each user, resulting in a set of 85.4 million tweets. From this corpus we then took a random sample of 25k tweets containing terms from the lexicon and had them manually coded by CrowdFlower (CF) workers. Workers were asked to label each tweet as one of three categories: hate speech, offensive but not hate speech, or neither offensive nor hate speech [7]. This hand labeling functioned as the golden standard for the model validation. In this project, I only focused on the "tweet" column which contained the text samples and the "class" column which was labeled by the CF workers. The histgraph [figure 4] shows the distribution of each text class: 0 for hate speech, 1 for offensive language, and 2 for neither.

Experimental details: Before applying any machine learning models, our project first inspects the data and making sure that they are feasible for analysis. Out cloned dataset has a sample of 24,802 labelled tweets stored in CSV format. After importing the CSV into Google Colab, we replaced the mentions, URLs, excessive white spaces and mentions with a single gap. After that, the tweets were lemmatized, keeping only nones, adjectives, verbs and adverbs using the spacy library. Onwards, I defined a remove_stopwords function which filtered out all stopwords contained in the nltk library. After finishing the text preprocessing, this project printed out again the ten first row of the CSV file to inspect the outcome. The following two screenshots illustrate the difference between the pre-preprocessed and post-preprocessed text (figure 5 and figure 6).

This project consulted the article *Build Your First Text Classification model using PyTorch* by Aravind Pai [19] when constructing and training the LSTM model. It first built the vocabulary for the text and converted them into integer sequences embedding). Vocabulary contained unique words in the entire text. Each unique word is assigned an index. Then this project created an iterator using in the BucketIterator form to format the embeddings into mini-batches and padded them into the same length. The LSTM model was built with PyTorch by defining two functions: the init and the forward. The init function would be invoked when an instance is created. This function will initialize the arguments and pass it to the class. The forward function was set to perform forward pass of the input we then define the hyperparameters and instantiate the model. Then this project set the optimization model to Adam and the loss function to CrossEntropyLoss. I left my initial attempts with Adam and BCELoss. BCELoss only processed binary target and I was not aware that the "class" column had 3 types of input. Next, this project defined the training function and the evaluation function. Finally, this project trained the model for 20 epochs and saved the best model of every epoch.

Upon completion of the LSTM model, this project constructed our baseline model. The preprocessing of the text remained the same as the LSTM classifier. Instead of using the deep learning, here the project adopted a simple statistics-based classification method Multinomial Naive Bayer. This project decided to use Multinomial model because it was aware that the target labels had three elements. Thus, the Gaussian Bayer would yield very unsatisfactory results while Multinomial Bayer would be a more suitable

function. The project performed vectorization and generated text features with Tfidf modules from the sklearn library. Then it fed the split dataset to fit the model.

Evaluation method: In the LSTM classifier, the evaluation function was integrated in the training process. For every epoch, the project activated the evaluation function to inspect the changes in training loss, training accuracy, validation loss and validation accuracy. In the baseline model, the author compared predicted results with the labelled result to obtain the accuracy score.

Results: The baseline model achieved an accuracy rate of 79.22%, while the LSTM model outperformed the former with 84.68% on the validation dataset. The result from every epoch experiment is shown in graph 7.

5. Analysis

From graph 7, this project observes a high probability of over-fitting in this LSTM model. While the training loss steadily decreases from 0.361 to 0.218, the validation loss gradually increases from 0.3 to 0.445. At the same time, Training accuracy improved from 86.33% to 91.87% while the validation set accuracy stagnated around 84% through the whole course. To minimize this potential overfitting, this project first reduced weighs in the hidden layer and added some dropouts randomly removing certain features by setting them to zero. However, those efforts did not seem to improve the model performance significantly. Another redress method to overfitting is regularization. This will add a cost to the loss function of the network for large weights (or parameter values). As a result, we would get a simpler model that will be forced to learn only the relevant patterns in the train data. However, for this project, the most probably reason for overfitting could be the size of our training data. For a sophisticated and powerful model such as a neutral network comparing to Naïve Bayer, it requires a bigger dataset to reduce overfitting. However, due to time constrain, this project is not able to enlarge our dataset at this stage.

6. Conclusions and Reflection

This project performed a deep learning automated hate speech detection using PyTorch library and the LSTM model. The model is trained with 70% of the dataset and tested by the remaining 30%. As a work of reproducibility, this project managed to adapt a simple LSTM model on a different dataset labelled by Davidson. The project achieved an accuracy rate of 84.68%, which is a state-of-the-art result. The implementation process provided the author with an opportunity to dive deeper into the theory behind text classification and the rationale behind automated

```
Epoch: 0
        Train Loss: 0.477 | Train Acc: 82.05%
         Val. Loss: 0.405 |  Val. Acc: 82.76%
Epoch: 1
        Train Loss: 0.376 | Train Acc: 85.51%
         Val. Loss: 0.376 |  Val. Acc: 85.70%
Epoch: 2
        Train Loss: 0.361 | Train Acc: 86.33%
         Val. Loss: 0.400 |  Val. Acc: 83.81%
Epoch: 3
        Train Loss: 0.345 | Train Acc: 87.35%
         Val. Loss: 0.383 |  Val. Acc: 85.46%
Epoch: 4
        Train Loss: 0.350 | Train Acc: 86.87%
         Val. Loss: 0.406 |  Val. Acc: 84.79%
Epoch: 5
        Train Loss: 0.327 | Train Acc: 87.93%
         Val. Loss: 0.384 |  Val. Acc: 85.90%
Epoch: 6
        Train Loss: 0.303 | Train Acc: 88.92%
         Val. Loss: 0.392 |  Val. Acc: 85.49%
Epoch: 7
        Train Loss: 0.292 | Train Acc: 89.18%
         Val. Loss: 0.556 |  Val. Acc: 80.47%
Epoch: 8
        Train Loss: 0.285 | Train Acc: 89.37%
         Val. Loss: 0.399 |  Val. Acc: 85.31%
Epoch: 9
        Train Loss: 0.272 | Train Acc: 89.90%
         Val. Loss: 0.392 |  Val. Acc: 84.63%
Epoch: 10
        Train Loss: 0.265 | Train Acc: 90.20%
         Val. Loss: 0.412 |  Val. Acc: 85.42%
Epoch: 11
        Train Loss: 0.275 | Train Acc: 89.90%
         Val. Loss: 0.417 |  Val. Acc: 85.25%
Epoch: 12
        Train Loss: 0.255 | Train Acc: 90.21%
         Val. Loss: 0.407 |  Val. Acc: 85.49%
Epoch: 13
        Train Loss: 0.248 | Train Acc: 90.74%
         Val. Loss: 0.420 |  Val. Acc: 84.79%
Epoch: 14
        Train Loss: 0.247 | Train Acc: 90.67%
         Val. Loss: 0.417 |  Val. Acc: 84.34%
Epoch: 15
        Train Loss: 0.241 | Train Acc: 91.26%
         Val. Loss: 0.441 |  Val. Acc: 84.70%
Epoch: 16
        Train Loss: 0.230 | Train Acc: 91.40%
         Val. Loss: 0.428 |  Val. Acc: 84.82%
Epoch: 17
        Train Loss: 0.236 | Train Acc: 91.04%
         Val. Loss: 0.431 |  Val. Acc: 84.95%
Epoch: 18
        Train Loss: 0.229 | Train Acc: 91.63%
         Val. Loss: 0.432 |  Val. Acc: 84.35%
Epoch: 19
        Train Loss: 0.218 | Train Acc: 91.87%
         Val. Loss: 0.445 |  Val. Acc: 84.68%
```

Figure 7. Epochs Results from LSTM Model

hate speech detection. As a modest project which only satisfied with reproducibility, it will mark a starting point of my adventure in natural language processing.

Despite the popularity of the LSTM and other RNN variants in deep learning applications, scientists soon realized its drawbacks and called for alternatives. Although LSTM was designed to learn long term information, it still cannot memorize sequences that scale beyond a certain range. Moreover, the RNN models are not hardware friendly which means that it takes many computation resources. To enhance the performance of the LSTM output, some earlier efforts were spent on combining the model with "attention" (Wang et al. 2016) [23]. In 2017, Vaswani proposed a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely [22]. Transformer soon became popular. Another research that worth the reader's attention is the 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction created by Elbayad et al. (2018) [8]. This model made the attention-like properties pervasive throughout the network instead of concentrating on few steps.

Apart from efficiency and accuracy improvement, the scientific community also faces other technical challenges. One is to tell the subtle contextual differences. For example, "the Nazi organization was great." This would be considered hate speech because it shows support for a hate group. However, "the Nazi's organization was great" is not supporting their ideals but instead commenting on how well the group was organized. However, after removing some stop words or lemmatization, both text corpus would be the same and make the automated detection model yield the same judgement. Another issue is systematic detection evasion. Since the detection model is a closed-loop system, once individuals are aware of the mechanism, they can actively evade the detection. In fact, we can observe more circumvention approach in actions against online censorship. Their logic is fundamentally similar. One example of a simple avoidance took place in New Zealand when government auto-removed some hateful posts, and their content reappeared as images containing the text (MacAvaney et al., 2019) [14]. Another example came from the "Love Attack" as this example: "MartiansAreDisgustingAndShouldBeKilled love." The message remains easy for humans to understand, but the algorithms don't know what to do with it. The only thing they can really process is the word "love." (Louise Matsakis, 2018) [16].

The challenges of automated hate speech detection even extend beyond the technical front. In the Western world, researchers and policymakers are still exploring the appropriate balance between freedom of speech and cyber authoritarianism. Therefore, beyond algorithm, community rules on social networks and post quarantine mechanisms also matter, according to Ullmann and Tomalin (2019). They proposed that instead of banning the senders of hate speech in a crude unilateral matter, the recipients of the hate speech should be given the agency to determine how they wish to handle the hate speech they received (e.g. deblurring the content, choose to read option). This approach potentially preserves the freedom of expression, but it poses a question on what society ultimately stands for when we encountered hate speech. If the initial motivation behind our hate speech policy is to remove hateful behaviours as much as possible, then Ullmann and Tomalin's proposal functions merely as a buffer. Alternatively, if the society believes that we should protect the potential target of hate crimes from too much exposure, Ullmann and Tomalin's proposal meets this goal while mitigating the restrictions on expression liberty. Humanities need to come up with a position first before we can design better algorithms accordingly.

7. Acknowledgements

Although this project is not an ambitious work and contains ample flaws that would make the future me laugh about, I would like to thank many people for supporting me throughout my struggle with remote participation and coding. My classmates Maximilian and Sebastian helped me to dial in the class via Skype in the first half of this semester when I was quarantined alone in Wuhan. Maximilian, Nikolas and Allison answered my questions on piazza regarding the text preprocessing and SparseTensor error. Vic Cheng, a friend in Wuhan, pointed out some room for improvement in my LSTM and Naïve Bayer models. Without their encouragement and generous support, I would not be able to go as far by myself in this team project.

References

[1] European Commission against Racism and Intolerance (ECRI). Hate speech and violence. `https://www.coe.int/en/web/european-commission-against-racism-and-intolerance/hate-speech-and-violence`.

[2] Aymé Arango, Jorge Pérez, and Barbara Poblete. Hate speech detection is not as easy as you may think: A closer look at model validation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–54, 2019.

[3] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Shanita Biere, Sandjai Bhulai, and Master Business Analytics. *Hate Speech Detection Using Natural Language Processing Techniques*. PhD thesis, Master's dissertation, Vrije Universiteit Amsterdam, 2018.

[6] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

[7] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*, 2017.

[8] Maha Elbayad, Laurent Besacier, and Jakob Verbeek. Pervasive attention: 2d convolutional neural networks for sequence-to-sequence prediction. *arXiv preprint arXiv:1808.03867*, 2018.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[11] Ritesh Kumar, Ramprashanth V, Aishwarya Reganti, Purav Shah, and Akshit Bhatia. Video: Detection of aggressive behaviour on social media, 2017. https://www.microsoft.com/en-us/research/video/detection-aggressive-behaviour-social-media/.

[12] Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, 2018.

[13] Anna Liu. *Neural Network Models for Hate Speech Classification in Tweets*. PhD thesis, 2018.

[14] Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. Hate speech detection: Challenges and solutions. *PloS one*, 14(8), 2019.

[15] Mihai Manolescu, Denise Löfflad, Adham Nasser Mohamed Saber, and Masoumeh Moradipour Tari. Tueval at semeval-2019 task 5: Lstm approach to hate speech detection in english and spanish. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 498–502, 2019.

[16] Louise Matsakis. To break a hate-speech detection algorithm, try 'love', 2018. https://www.wired.com/story/break-hate-speech-algorithm-try-love/.

[17] Casey Newton. The trauma floor: The secret lives of facebook moderators in america, 2019. The Verge.

[18] Christopher Olah. Understanding lstm networks, 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[19] Aravind Pai. Build your first text classification model using pytorch, 2020. https://www.analyticsvidhya.com/blog/2020/01/first-text-classification-in-pytorch/.

[20] J Pálmadóttir and Iuliana Kalenikova. Hate speech an overview and recommendations for combating it. *Icelandic Human Rights Centre*, pages 1–27, 2018.

[21] Stefanie Ullmann and Marcus Tomalin. Quarantining online hate speech: technical and ethical perspectives. *Ethics and Information Technology*, pages 1–12, 2019.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[23] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[24] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[25] Steven Zimmerman, Udo Kruschwitz, and Chris Fox. Improving hate speech detection with deep learning ensembles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.