Offensive/Non-offensive Tweets Michigan Detection

Jihye Moon, Hieu Nguyen, Bradshaw Pines

University of Connecticut

Abstract

As social media has became a popular form of communication, it is also a platform to spread misinformation and hate. Due to the recent Michigan Protest in 2020, there are evidences of hate and offensive tweets. Although people have the right to the freedom of speech, it is a duty for Twitter to detect these action and place sensitive settings on their platform. In this research, we collect data from the Michigan Protest with the goal is to use Natural Language Processing technique to detect offensive tweets. Our results provide a moderate accuracy of 84 percents which is superior compare to Twitter's default engine.

Keywords: , Machine Learning, Feature Selection, Natural Language Processing

Email addresses: jihye.moon@uconn.edu (Jihye Moon), hieu.nguyen@uconn.edu (Hieu Nguyen), Bradshaw.Pines@uconn.edu (Bradshaw Pines)

Preprint submitted to CSE 5095: Social Mining and Analysis *December 24, 2020*

Contents

1. Introduction & Motivation

Social Media is used to quickly create and share content with the public. Specifically, Twitter has become a platform for leaders, politicians, and news reporters to get information directly to the public. Along with the information is the opportunity for instant reactions and hashtags to group similar information. The coronavirus pandemic has demonstrated these aspects of Twitter with information from all sources being spread on Twitter with instant reactions. A specific disagreement that has been argued on Twitter is if lockdowns are the best decision during the coronavirus pandemic. Many left-leaning liberals believe states should stay in lockdown to help stop the spread of the coronavirus. In opposition, many right-learning conservatives think the lockdown is unconstitutional, a power grabs for elites, not worth the negative impact on the economy, and question its effectiveness in stopping the spread of the coronavirus. This disagreement goes all the way to the highest level of politics with right-leaning President Trump calling for the end of these lockdowns specifically targeting states with left-leaning governors who are in support of the lockdowns such as Michigan, Virginia, Minnesota, Pennsylvania, and North Carolina. This disagreement was so strong that there was a plot to kidnap the governor of Michigan in an attempt to "LIBERATE" Michigan which stemmed from a tweet from the president. In addition, protests broke out in support of opening up the mentioned states. As expected, the strong divide in opinions on lockdowns and twitter's ability to give the public instant reactions to events created an outpouring of support and opposition to the protests.

This paper focuses on predicting if a reaction to the protests is offensive or non-offensive. Both arguments have tweets of both types and in these turbulent times, there has been an increase in the offensive tweets. This is a problem because Twitter has struggled to detect these offensive tweets leading to them not being censored and escalating tension across the country. Specifically, according to the EU code of conduct Twitter only removes about 43 percent of offensive tweets. Our goal is to use machine learning to find these tweets at a rate significantly higher than Twitter. To do so we will evaluate the data, preprocess the data, extract features, and train Models. For evaluating the data we made three-word clouds (entire dataset, non-offensive and offensive tweets), look at the most common words in non-offensive and offensive tweets, and look at words unique to non-offensive and offensive tweets. For the preprocessing, we removed junk words, tokenize the data, stem the data, remove stop words, and extract parts of speech. For the feature extraction,

we used TF-IDF, and DistilledBERT. To select informative features, we also used Random Forest and Principal component analysis (PCA). For classification tasks, we used Decision Tree, Random Forest, Logistic Regression, Support Vector Machines, and Deep Neural Networks. Finally, we discuss the results and conclusions.

2. Data Collection

The specific context of the data collection is tweets reacting to two protests in Michigan. The first protest is referred to as "Operation Gridlock" as it caused a gridlock in Lansing, MI, and blocked the entrance to a level 1 trauma center. For this event the hashtag used to collect data was operationgridlock and amounted to approximately 3000 tweets. The other protest used for data collection was approximately one month later with the name "Michigan Protest" which involved protesters storming the capital building with guns wearing camouflage and military garb. The hashtag used for this protest was michiganprotest and resulted in approximately 3000 tweets. In an attempt to help machine learning, which requires extensive training to become effective, we hope to increase our data set. We believe that doing so it would result in better accuracy from our machine learning techniques.

3. Data Labeling

The data is labeled with the goal of differentiating between offensive and non-offensive tweets. This decision is based on the idea of hate speech being offensive and its definition of "Hate speech are words that are generally targeted to be hurtful towards a person or a group of people on the basis of race, caste, gender, color, ethnicity and also support and promote violence against the targeted groups". Specifically, we looked for content that made people feel inferior, insult, bully, humiliate, incite anger and violence. For example, comparing people to animals such as rats or evil historical figures such as Hitler would be considered offensive. It is important to label tweets offensive that expressed hostility or animosity towards specific groups without incorrectly labeling tweets that included criticism to protect freedom of speech as offensive. One area of offensive tweets is insults based on people's immutable characteristics such as skin color, nationality, gender and race. In addition, we labeled tweets offensive if they insult people's intellect and way of life such as culture, geographical stereotypes, political opinions, affiliation and views. For example, calling someone names such as MAGAts because of their political view is offensive. Another example is

calling someone white trash or a red neck is also offensive. Another area of offensive tweets is based on people's appearance, intelligence, mental capability, maturity and masculinity. An example of this is fat shaming or insulting a hair style. There are also offensive tweets specific to COVID-19 such as wishing someone gets COVID-19 or to a further extent dies from COVID-19. It is also important to not label based on specific words alone as the intent of the word is needed to know if it is offensive. An example of this is profanity which may be expected in offensive tweets but can also occur in non-offensive tweets when being used for emphasis. There is no consensus on exactly what is offensive or non-offensive but with these guidelines we feel this data set is labeled accurately.

4. Preliminary Analysis

4.1. Preprocessing

4.1.1. Remove Junk Words

Figure 1: Example of Removing Junk Words

	text	O/NO	Y/N	S/NS	new_text
0	Guys check out this drone footage over Lansing...	NO	Y	NS	Guys check out this drone footage over yester...
1	@muffnbear Our governor @GovWhitmer has a pres...	NO	Y	NS	Our governor has a press conf and made it a ...
2	@PoitinJar @DonaldJTrumpJr I grew up in Lansin...	O	N	S	I grew up in . I m horrified ders would bloc...
3	As a Michigander for my entire life, I support...	NO	N	NS	As a der for my entire life, I support thatwom...
4	To all the mfks I know that live in Michigan I...	O	N	S	To all the mfks I know that live in Im just w...

To get started with preprocessing we removed words that will not help determine if a tweet is offensive or not. Some of them are general to tweets such as @, #, hyperlinks, and RT while others are specific to the topic such as Lansing, lansing, Michigan, people, MI, and today. Since the protests took place in Lansing, MI and they involve people the tweets are highly likely to include these words whether they are offensive or not. In the example, this is shown with @PoitinJar, @DonaldJTrumpJr, and Lansing on line 2 being in "text" but removed from "new_text".

4.1.2. Tokenization

Figure 2: Example of Tokenization

	text	O/NO	Y/N	S/NS	new_text
0	Guys check out this drone footage over Lansing...	NO	Y	NS	[Guys, check, out, this, drone, footage, over,...
1	@muffnbear Our governor @GovWhitmer has a pres...	NO	Y	NS	[Our, governor, has, a, press, conf, and, made...
2	@PoitinJar @DonaldJTrumpJr I grew up in Lansin...	O	N	S	[I, grew, up, in, ., I, m, horrified, ders, wo...
3	As a Michigander for my entire life, I support...	NO	N	NS	[As, a, der, for, my, entire, life„ I, suppor...
4	To all the mfks I know that live in Michigan I...	O	N	S	[To, all, the, mfks, I, know, that, live, in, ...

Tokenization is breaking the tweet into a list of words so that each word is a feature. Thus, we will able to preprocess the data better and feed these features to the classification models.

4.1.3. Stemming

Figure 3: Example of Stemming

	text	O/NO	Y/N	S/NS	new_text
0	Guys check out this drone footage over Lansing...	NO	Y	NS	[guy, check, out, this, drone, footag, over, y...
1	@muffnbear Our governor @GovWhitmer has a pres...	NO	Y	NS	[our, governor, has, a, press, conf, and, made...
2	@PoitinJar @DonaldJTrumpJr I grew up in Lansin...	O	N	S	[i, grew, up, in, ., i, m, horrifi, der, would...
3	As a Michigander for my entire life, I support...	NO	N	NS	[as, a, der, for, my, entir, life„ i, support...
4	To all the mfks I know that live in Michigan I...	O	N	S	[to, all, the, mfks, i, know, that, live, in, ...

Stemming is the process of reducing all of the words in the dataset down to their stem which is important to make the training data denser. There are some concerns with stemming as the ending of words can be useful. For example, the word 'fucking' is not offensive but the word 'fuck' is, if we reduce all uses of the word fuck down to its stem, this distinction will be eliminated. Our current decision is that the positives of stemming outweigh the negatives so we continue to use it.

4.1.4. Remove Stop words

Figure 4: Example of Removing Stop Words

	text	O/NO	Y/N	S/NS	new_text	new_text_filter
0	Guys check out this drone footage over Lansing...	NO	Y	NS	[guy, check, out, this, drone, footag, over, y...	[gridlock, patman, oper, check, protest, yeste...
1	@muffnbear Our governor @GovWhitmer has a pres...	NO	Y	NS	[our, governor, has, a, press, conf, and, made...	[point, traffic, org, devo, press, support, ca...
2	@PoitinJar @DonaldJTrumpJr I grew up in Lansin...	O	N	S	[i, grew, up, in, ., i, m, horrifi, der, would...	[fat, fool, horrifi, would, protest,, walk, la...
3	As a Michigander for my entire life, I support...	NO	N	NS	[as, a, der, for, my, entir, life„ i, support...	[entir, state, demonstr, support, hope, home, ...
4	To all the mfks I know that live in Michigan I...	O	N	S	[to, all, the, mfks, i, know, that, live, in, ...	[stupid, look, downtown, live, hope, life...st...

Another important part of preprocessing is eliminating stop words as they are common words that have little influence if a tweet is offensive or non-offensive. We used nltk's English stop words

to get a broad list of stop words to remove such as "me", "my", "we", and "our". In addition, we added to that list of words that we find have no influence on if a tweet is offensive and words such as "u" and "ur" that are common in tweets but are not proper English so are not in nltks stop words.

4.1.5. Part-of-speech (POS) Tagging

Figure 5: Amount of each Part of Speech in Dataset

'NN': 36288 (noun, singular)	'JJS': 116 (adjective, superlative)
'JJ': 2175 (adjective)	'DT': 84 (determiner)
'VB': 1131 (verb)	'POS': 70 (possessive ending)
'CD': 1084 (cardinal number)	':': 68 (colon)
'RB': 775 (adverb)	'RBR': 54 (adverb, comparative)
'IN': 554 (preposition)	'VBG': 35 (verb, present participle)
'NNS': 528 (noun, plural)	'JJR': 25 (adjective, comparative)
'MD': 313 (modal)	'CC': 8 (coordinating conjunction)
'VBD': 183 (verb,past tense)	'LS': 2 (list item marker)
'VBN': 164 (verb,past participle)	'WP$': 1 (possessive wh-pronoun)

POS tagging is attaching each word's part of speech to it for the purpose of finding if specific parts of speech do not influence if a tweet is offensive or non-offensive. We use nltk's POS tagger and find each part of speech and the amount of each to be what is shown above. We are still trying to figure out the best way to implement the pos tag, we know that specific parts of speech such as adverbs are unlikely to impact if a tweet is offensive but we do not believe to have found the most effective use yet. In addition removing parts of speech such as colons and cardinal numbers could help efficiency because they will never indicate offensiveness.

4.2. Word Cloud

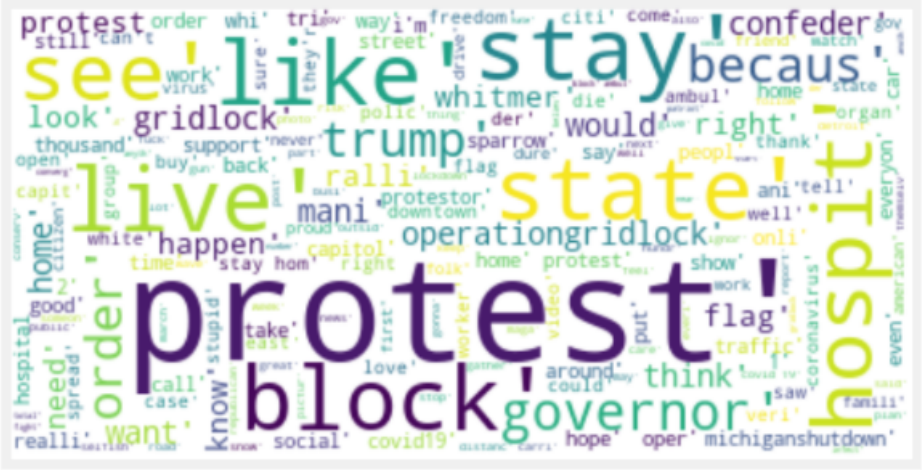
4.2.1. Entire Dataset

Figure 6: Entire Dataset Word Cloud

This word cloud is used to represent the most common words of the entire data set. Many of the larger words, representing the most common words, are what we would expect such as protest, protesting, COVID, American, Governor, stay, Trump, home, and state. With the topic in mind, these words come as no surprise and would not be expected to influence if a tweet is offensive or non-offensive. An interesting large word is 'terrorists', as that is a bold term which in many cases would be offensive. In addition, we see 'idiot' and 'stupid' at a decent size which definitely would be expected to be part of an offensive tweet. Overall, the entire data set word cloud is a great visual to get a feel for what our data looks like and understand some of the noise words that should be removed.

4.2.2. Non-Offensive Tweets

Figure 7: Non-Offensive Tweets Word Cloud



Taking a look at the word could of non-offensive words starts to give us an idea of the words that are more common in non-offensive tweets compared to offensive ones. Once again we see common words such as protest, state, live, stay, like, see, and governor. Some of the more interesting large words are the "hospital", "block", "operationgridlock", and "michiganshutdown". These are nice to see because they are directly related to the exact protest we are looking at and therefore give a good feeling that these tweets are on the topic of interest. When getting data based on hashtags it is not a guarantee the data will be exactly what we are looking for but seeing these words in the word cloud gives confidence that these tweets are on topic.

4.2.3. Offensive Tweets

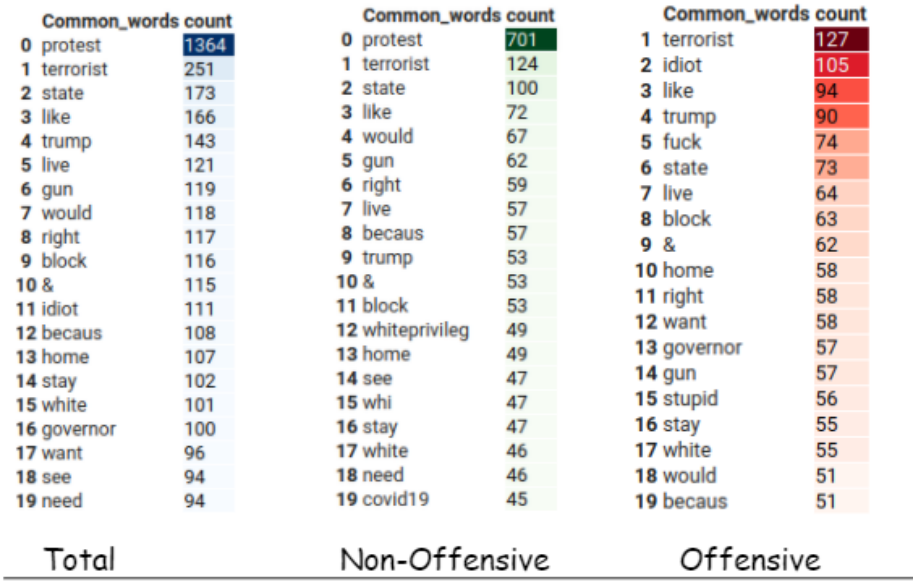
Figure 8: Offensive Tweets Word Cloud

In what could be considered the most informative word cloud, this word cloud gives us a look at the most common offensive words. Looking past the common words of protest, like, live, and trump we can see words that should be in consideration for being offensive such as "stupid", "idiot", "fuck", "dumb", "ass", "asshole", and "moron". With the combination of our logic on what could be offensive and the common words in offensive tweets we can get an idea of what keywords signify that a tweet is likely to be offensive. For complete accuracy, it is necessary to get the meaning of the entire tweet but as a starting point noticing these words and using them in the algorithm can be an effective way to increase accuracy.

4.3. Statistical Analysis of Words

4.3.1. Common words

Figure 9: Most Common Words in All, Non-offensive Offensive Tweets

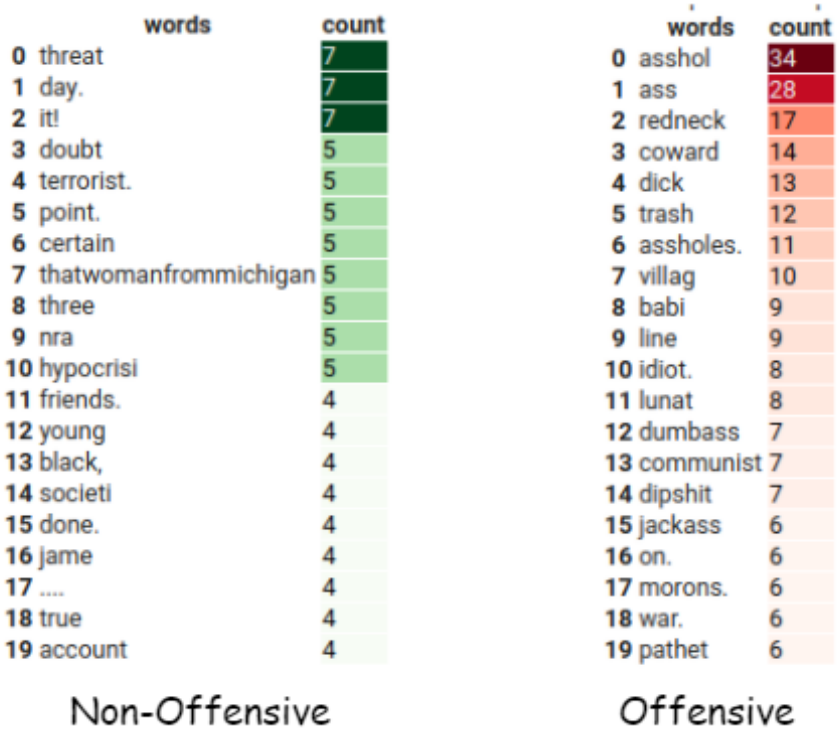


Once we have viewed the word cloud for a visual of the data, it is helpful to look at the exact numbers on the most common words. Specifically, looking at words that are common in offensive tweets but are not common in general or in non-offensive tweets. We can see idiot is the third most common word in offensive tweets at 105 and is twelfth most common on the total with 111, which we can do the math and find 105/111 = 94.6 percent of the time if "idiot" is used it is in an offensive tweet. With a goal of 90 percent accuracy, this knowledge is more helpful as we know that idiot is near perfect as an offensive tweet indicator. We also find that fuck is the sixth most common in offensive words but is not in the top 20 of total or non-offensive words. Thus, it is good to keep in mind going forward as it could be an indicator of an offensive tweet. Finally, we can see the same is the case for stupid as for fuck expect stupid has fewer uses (fuck can be a noun, pronoun, adjective, adverb, or interjection when stupid is only an adjective) and therefore is even more likely to be an indicator of an offensive tweet.

4.3.2. Unique words

Figure 10: Most Unique words for Non-Offensive Offensive Tweets

Figure 11: Remove Common Words Example



	words	count
0	threat	7
1	day.	7
2	it!	7
3	doubt	5
4	terrorist.	5
5	point.	5
6	certain	5
7	thatwomanfrommichigan	5
8	three	5
9	nra	5
10	hypocrisi	5
11	friends.	4
12	young	4
13	black,	4
14	societi	4
15	done.	4
16	jame	4
17	....	4
18	true	4
19	account	4

Non-Offensive

	words	count
0	asshol	34
1	ass	28
2	redneck	17
3	coward	14
4	dick	13
5	trash	12
6	assholes.	11
7	villag	10
8	babi	9
9	line	9
10	idiot.	8
11	lunat	8
12	dumbass	7
13	communist	7
14	dipshit	7
15	jackass	6
16	on.	6
17	morons.	6
18	war.	6
19	pathet	6

Offensive

Once we know the words that are most common in offensive and non-offensive tweets, we can look into what words are most common in being unique to offensive or non-offensive tweets. This is very helpful as it tells what words are most likely to be indicators of a tweet being offensive. From figure 6 we can see that the common words in non-offensive tweets are less helpful as it seems very possible they could be used in offensive tweets. On the other hand, the offensive words give us a feel for the type of language that is being used in the offensive tweets and it is clear that these words would be very likely to be only in offensive tweets. For example, its hard to imagine a tweet that uses the word asshole and is non-offensive so that could be an earlier indicator that a tweet is offensive. So, this table does a great job of showing the words that are in some of the more obvious offensive tweets.

5. Social Features

One advantage of getting data from tweets is that it allows you to also obtain all of the social features. These social features can help identify if a tweet is expected to be offensive or not. The specific social features used in this paper are the length of the tweets, the amount of favorites the tweet gets, the number of followers of the person who posted the tweet and how many retweets the tweet got. If there is a significant difference between the amount of the social feature for offensive tweets and non-offensive tweets then that feature can be used to help predict if the tweet will be offensive or not before even evaluating the words. The features are shown in Table 1.

Table 1: Social Feature List

Information	Non-offensive	Offensive
Label Percentage	0.72	0.28
Ave. text width	166.97	170.23
Ave. favorite count	20.99	24.56
Ave. followers count	6705.55	5306.05
Ave. friends count	2651.79	2477.29
Ave. listed count	71.62	53.26
Ave. is_retweet	0.17	0.17
Ave. number of hashtags	1.68	1.45
Ave. number of mentions	0.77	0.58
Ave. number of urls	0.47	0.42

Ave. text width: The number of characters in each tweet. There is no different in average text width between non-offensive and offensive tweets.

Favorite count: The number of "like" the tweet gets. From the table above, offensive tweets get more favorite count then non-offensive tweet.

Followers Count: The number of followers that follow that tweet owner. Owners of non-offensive tweets have higher followers count.

Friend count: The number of friends that the owner have (they both follow each other). We see offensive tweet's account has fewer friends on average.

Listed count: A indicator of popularity that show how many people have added the tweet's owner to the list. Non-offensive tweet account has a higher listed count on average compare to offensive tweet's account.

is retweet: Determine whether the tweet is retweeted from another account. There is not difference between both classes.

Hashtag count: Determine whether the tweet include hashtag. Non-office tweets have a slightly higher numbers of hashtag than offensive tweets on average.

Number of Mentions: Determine whether the tweet mention other twitter accounts. Non-offensive tweets mention other people more.

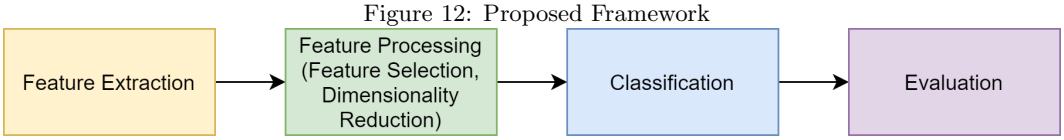
Number of urls: Attached links included in the tweet such as link to news. There is no big difference between both classes.

In addition, we use Sarcasm-based features consist of Lexical Features (POS-based TF-IDF), Sentiment Features (positive/negative/natural sentiment) as one of the extension of the social features on this paper. The Sentiment features were extracted by using **vaderSentiment** packages in Python.

6. Classification

In this research, our goal is to compare and obtain the best methods from our experiments. We analyze and classify tweets using a combination of feature selection and classification methods. Our work consists of feature extraction, feature processing (feature selection, dimensionality reduction), and classification parts shown in Figure. 12.

Figure 12: Proposed Framework



We use TF-IDF, DistilledBERT for the features extraction, Random Forest for the feature selection, Principal Component Analysis (PCA) for the dimensionality reduction, and six machine learning models to classify the offensive/non-offensive tweets. The details of each method are described below.

6.1. Feature Extraction

In this section, to select the best features for the offensive/non-offensive detection from tweets, we implemented three feature extraction methods: TF-IDF, and DistilledBERT. To train the MLs, the data sets were vectorized using those features, respectively. The details are described below.

6.1.1. TF-IDF

The Term Frequency-Inverse Document Frequency (TF-IDF) is the n-grams-based method. It represents the word appearance about each document by using the most frequent instances of every unique word. In this paper, the document is defined as a tweet, and the tweets were represented by the unigram 1-gram vectors. Each TF-IDF score is calculated by

$$TF - IDF = tf * log\frac{N}{df}$$

The TF is the number of times a unique word appearance from the tweets, df is the number of tweets, which contains the unique words and N is the total number of tweets. The TFIDF score allocates a higher weight to the words which occur more frequently over the data sets in determining whether a tweet is offensive or non-offensive. We extracted the TF-IDF features-based unigrams by using NLTK library, Python3.

6.1.2. DistilledBERT

The DistilledBERT is revised from Bidirectional Encoder Representations from Transformers (BERT), has the same architecture as the BERT (1). The BERT consists of a multilayer bi-directional transformer-encoder, where the transformer with parallel attention layers, which extract contextual information from text. Thus the BERT model outperformed other popular models such as ElMo, Word2Vec, and Glove for 11 natural language processing(NLP) tasks. The result shows that language-based word embedding can have a deeper sense of contextual information than a single language-based models. The DistilledBERT is the lighter version of the BERT, however, the performance is similar to the original BERT. The DistilledBERT was trained on the BooksCorpus dataset (800 million words) and English Wikipedia and, with a dimension of 764.

6.2. Supervised Models

6.2.1. Decision Tree

The Decision tree (DT) is one of the well-known data mining methods for forecasting task which provide reasonable classification and regression performances. It is constructed start from the root node, and split into left and right child nodes. These child nodes can then recursively split into more child nodes. Every node represents an output class and every branch represents the process which lead to the decision output, and the end node is the result. To improve the performance after the model is constructed, we compute the error rate and prune the tree. Decision trees have

a very low bias and high variance and tend model tend to over-fit if the training data is complex and present irregular pattern especially with data with a high dimensionality.

6.2.2. Random Forest

The Random Forest (RF) is extended from Decision Trees described above. The model is based on an ensemble learning classification technique, by using bagging. The bagging enables RF to decide the best model, which generates a variety of decision trees with different training sets and parameters, and evaluate their performance. To a certain degree it eliminates the overfitting problem that often occurs when using decision trees. Thus it has reported that the model outperformed other ML models when their data sets are small.

6.2.3. Logistic Regression

Logistic regression (LR) models are usually fit by maxinum likelihood, using the conditional likeluhood of G given X. since $Pr(G|X)$ completely specifies the conditional distribution, the multinomial distribution is appropriate. The log-likelihood for N observations is:

$$l(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

where $p_k(x_i; \theta) = Pr(G = k|X = x_i; \theta)$. For our research, we are classifying two classes, so the log-likelihood can be written:

$$l(\beta) = \sum_{i=1}^{N} y_i \beta^T x_i - \log(1 + e^{\beta^T} x_i)$$

where $\beta = \{\beta_0, \beta_1\}$ are the vector o the inputs x_i. To maximize the log-likelihood, we set the derivative to zero.

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i; \beta)) = 0$$

6.2.4. Support Vector Machines

SVR is a non-parametric kernel-based regression method used for extrapolating future values. More specifically, we shall be focusing on ϵ-SVR, a form of SVR in which a hyperplane is constructed with a loss function within precision. The SVR function can be expressed as:

$$f(x) = w^T \phi(x) + b$$

where $\phi(x)$ maps data from the input space to the feature space, w is a weight vector, and b is a bias constant. w and b are estimated by satisfying the following:

Minimizing: $\frac{1}{2}\|x\|^2$

Subject to:

$$y_i - (\langle w, \phi(x_i)\rangle + b) \leq \epsilon$$

$$(\langle w, \phi(x_i)\rangle + b - y_i \leq \epsilon$$

where x_i and y_i represent input and target values obtained from the training set. To address points outside this ϵ-insensitive band, we introduce slack variables ξ_i, ξ_i^* :

Minimizing: $\frac{1}{2}\|x\|^2 + C\sum_i^n \cdot(\xi_i + \xi_i^*)$

Subject to:

$$y_i - (\langle w, \phi(x_i)\rangle + b) \leq \epsilon + \xi_i$$

$$(\langle w, \phi(x_i)\rangle + b - y_i \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \leq 0$$

The constant $C > 0$ is used to represent the trade off between model complexity and training error. After taking the Lagrangian and optimizing with the above constraints, we are left with:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b$$

where $K(\cdot, \cdot)$ is a kernel function, α_i and α_i^* are nonzero Lagrangian multipliers and solutions to the dual problem.

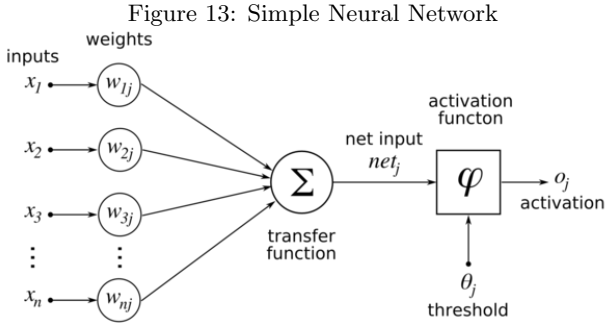
6.2.5. Deep Neural Networks

The deep neural network (DNN) is revised from artificial neural network (ANN). The theory of ANN has been developed more than 20 years ago. Due to the recent advancement in computing infrastructure and the availability of training data, ANNs have become popular in the recent years with their great performance in many different tasks especially in computer vision(CV) and natural language processing (NLP). Deep learning is a type of ANN in which it has multiple layers that enable the model to extract features from input data and use them for prediction. A multi-layer perception is a one of the classic feed forward artificial neural network. The model consists of input layer, hidden layers, and output layers. The hyper-parameters of the network is the numbers of the neurons in each layers and the number of layers. Each neuron has input (x), weight (w) and bias

(b) terms. Each layer also has an activation function φ which produce an output of the preceding neurons. An example of an output of a neuron:

$$y_i = b + \varphi \sum (W_i x_i)$$

Figure 13: Simple Neural Network



There are different types of nonlinear activation functions such as sigmoid, tangent, ReLu, leaky-ReLU, and softmax. A detail comparison of nonlinear activation functions are in (2).

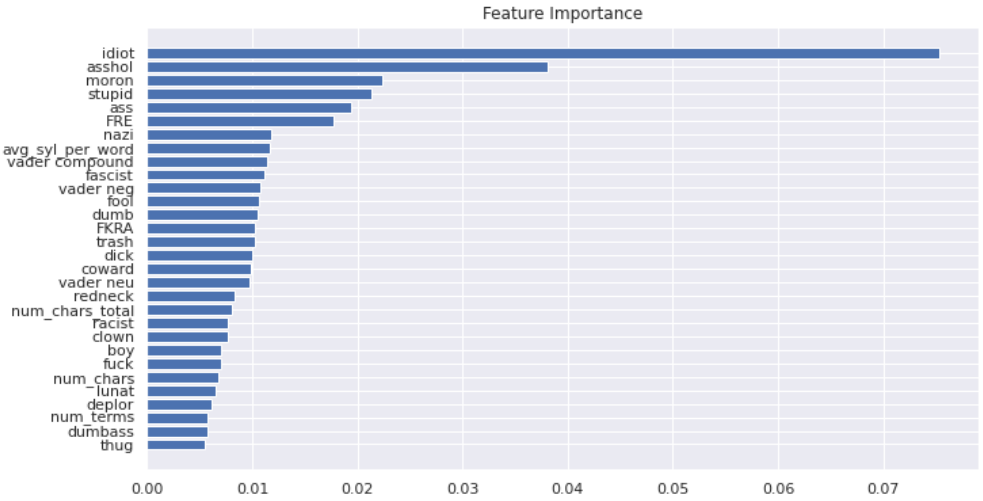
The learning stage of a neural network is through back-propagation. Once we output for each neuron is computed, the errors are also calculated and propagated back to the preceding layers. Then, an optimization functions will use that information and compute and adjust the weights until it achieve the optimal value which minimize the output error. Most common optimization algorithms are Stochastic Gradient Descent (SGD) (3), Adaptive Gradient Algorithm (AdaGrad) (4), Root Mean Square Propagation(RMSP)(5), and Adaptive Moment Estimation (ADAM)(6). Neural networks has proved to solve non-linear problems in many major fields in classification and regression due to its complexity and high number of parameters. However, a deep neural network with high number of parameters which could cause some well-known problem such as vanishing gradient in the feed forward-backpropagation, and might also cause over-fitting the data. Thus, it is important to fine-tuning the the hyper-parameters of the network. Hyper-parameters are the variables that affect the networks such as the number of neurons, hidden layers, regularization technique, activation function, learning rate, epochs size, optimization functions, etc. Finding the optimal hyper-parameters means better network performance. Some of the most well-know method for finetuning hyper-parameters are Grid Search, Mannual Search, and Automated Hyperparameter Tuning.

6.3. Feature Processing

6.3.1. Feature Selection

We utilize the Random Forest (RF) classification model to estimate the importance of those features. The feature importance analysis is used to determine which features are more useful. It is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability is calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. As shown in Fig. 14, the technical features are sorted as the most contributing in the classification for the offensive/non-offensive tweets. Shown in 14, the social features are sorted as having high importance for the labels: FRE, Vader neg(Negative sentiment), Vader neu(Neural sentiment), FKRA considered as Sarcasm-based features calculated by vaderSentiment packages. In addition, hate-words such as asshol, moron, stupid, and ass are also sorted as contributing to classifying non-offensive and offensive tweets. We select the best 300 features and train them with machine learning models for tweet classifications.

Figure 14: Selected Informative Features by Random Forest



6.3.2. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality-reduction methods that is used to reduce the dimensionality of the data sets. From section 4.3 and 6.4, we can observe that some words weight more than other, thus it is possible that many features in our data do not contribute to the fitting of the model. Traditional methods like DT and RF suffer from the curse of dimensionality. Thus, we perform PCA to our features to reduce the number of features while maintain

95 percents of the variances. We apply the PCA training and test sets individually after the feature selection by RF.

6.4. Experimental Setup

In this paper, we conducted three feature extractions - TF-IDF, and DistilledBERT. For classification tasks, DT, RF, LR, SVM with different three kernels (linear, poly, and RBF), and DNN were implemented, respectively. 80% of the data was utilized as the training sets. The remaining 20% was used for test sets. The data was split by using Stratified sampling. The sampling is used to improve sampling efficiently by putting tweets with similar characteristics together into stratas. The goal is to have the data within each strata to be as similar as possible and each strata to be as different from the rest as possible. This is done using StratifiedShuffleSplit from sklearn.modelselection and in this paper we used 5 stratas. Before training, up-sampling technique was utilized. The up-sampling is used in this paper to help deal with the issue of the uneven data set. The data itself has 2,909 non-offensive tweets and only 1,104 offensive tweets creating a problem for the machine learning models. To solve this, the training data is manipulated using a process known as up-sampling that produces a sequence of samples that would have been attained if there had been a higher rate of sampling. In a sense, it helps make up for a smaller data set by creating more samples of the minority set which in this case in the offensive tweets. This is done using the feature re-sampling from sklearn.utils and results in a balanced data set for the non-offensive and offensive tweets with both having 2,036 tweets. Those training/test sets were scaled from -1 to 1 using the linear regression, individually. All parameters of those models are set by the default of the sklearn Library. These models were trained using Python3 on Google Colab.

6.5. Performance Metrics

We validate proposed methods using 5-fold cross-validation. The result of the validation is an average of five results using the same model but different training and test data. The performance is evaluated using accuracy, recall, precision, and F1 score. These metrics are calculated through as below equations.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

In the Eqs, TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) are utilized. To define them as good performance, The metrics should be close to 1.

6.6. Results and Analysis

We implement two experiments to compare different options. The first experiment is to evaluate the performance of ML models using DistilledBERT and TF-IDF features, resspectively. The second one is to evaluate the performance depending on whether applying the feature processing task or not.

6.6.1. DistilledBERT vs. TF-IDF features (without features selection)

Table 2: Performance Comparison between DistilledBERT vs. TF-IDF features

Feature	Model	Accuracy	Precision	Recall	F1
DistilledBERT without social features	LinearSVC	0.72	0.68	0.66	**0.67**
	SVC	0.73	0.63	0.64	0.63
	DT	0.66	0.57	0.57	0.57
	RF	**0.75**	0.57	**0.71**	0.57
	LR	0.71	**0.65**	0.65	0.65
	MLP	0.72	0.63	0.65	0.64
DistilledBERT wth social features	LinearSVC	0.73	0.71	0.68	**0.69**
	SVC	0.74	0.70	0.69	**0.69**
	DT	0.66	0.57	0.57	0.57
	RF	**0.75**	0.59	**0.72**	0.58
	LR	0.74	**0.70**	0.68	**0.69**
	MLP	**0.75**	0.67	0.68	0.67
TF-IDF without social features	LinearSVC	0.78	0.72	0.72	0.72
	SVC	0.81	0.71	0.79	0.73
	DT	0.77	0.71	0.71	0.71
	RF	**0.81**	**0.76**	**0.77**	**0.76**
	LR	0.76	0.70	0.70	0.70
	MLP	0.78	0.69	0.72	0.70
TF-IDF wth social features	LinearSVC	0.78	0.71	0.72	0.72
	SVC	**0.80**	**0.67**	**0.77**	**0.69**
	DT	0.74	0.67	0.67	0.67
	RF	0.79	0.65	0.80	0.67
	LR	0.78	0.71	0.72	0.72
	MLP	0.77	0.69	0.72	0.70

As shown in Table. 2. the best accuracy 0.81 was obtained by using TF-IDF (excluding social features) with Random Forest. The combination also provides the best F1 scores of 0.76. All results from TF-IDF based models provide superior performance compared with the DistilledBERT-based models at all cases. Typically neural network-based models like BERT require a large data set to train them, however, in this case, our data set is small so statistics-based TF-IDF model provides the better performance. Note that the case of using social features provides less performance than those using only original features. The size of the social feature is 2,696, thus, the cases including the social features have higher dimension than the cases excluding those features. Therefore, the results prove that the Curse of dimensionality problems that the performance of the machine learning model could be reduced if the data size is small but features are complicated.

6.6.2. TF-IDF+social features with/without Feature Processings

Table 3: Performance Comparison of TF-IDF with/without Feature Processings

Feature	Model	Accuracy	Precision	Recall	F1
TF-IDF wth social features	LinearSVC	0.78	0.71	0.72	0.72
	SVC	**0.80**	**0.67**	**0.77**	**0.69**
	DT	0.74	0.67	0.67	0.67
	RF	0.79	0.65	0.80	0.67
	LR	0.78	0.71	0.72	0.72
	MLP	0.77	0.69	0.72	0.70
TF-IDF wth social features+RF	LinearSVC	**0.84**	0.78	**0.80**	**0.79**
	SVC	**0.84**	**0.79**	**0.80**	**0.79**
	DT	0.74	0.67	0.68	0.67
	RF	0.81	0.71	0.78	0.73
	LR	**0.84**	0.78	**0.80**	**0.79**
	MLP	0.80	0.74	0.75	0.74
TF-IDF wth social features+RF+PCA	LinearSVC	0.83	0.78	0.79	0.78
	SVC	**0.84**	**0.78**	**0.79**	**0.79**
	DT	0.74	0.66	0.67	0.66
	RF	0.83	0.76	0.79	0.77
	LR	0.83	0.77	0.79	0.78
	MLP	0.81	0.73	0.76	0.74

As shown in Table 3, the performance improves when feature processing takes are applied. The best performance was obtained by TF-IDF with the feature processing using only RF, which provides the best accuracy of 0.84 and F1 score of 0.79. The size of the TF-IDF with social features is 4,991, and the size is reduced by random forest to 300. In the case of applying PCA, the feature size is decreased to 224. The case of using the RF+PCA provides less performance than those

using of only RF, but the size of dimension is smaller but the performance is just reduced to 1%. Using PCA has high potential to run model faster than other cases. These results suggest that selecting informative features by reducing the size can reduce the data complexity and improve the performance, which prevent it from the Curse of dimensionality issues.

7. Related Research

Our first paper for related work is titled "Fine-grained Sentiment Classification using BERT" which uses a BERT model to build a sentiment classifier. This paper described how most papers use a binary sentiment classification but found that the BERT model is simple and better performing. Specifically, the paper did preprocessing, sequence embedding from BERT, used a dropout with probability factor of 0.1 to regularize and prevent overfitting and used Softmax classification to layer the output where Softmax classification layer output is the probabilities of the input text belonging to each of the classs labels such that the sum of probabilities is 1. As shown in the results bellow this process got a higher accuracy than all other models in each trial. This led us to include the BERT model in our process. (7)

7.1. "Fine-grained Sentiment Classification using BERT"

Figure 15: Results

TABLE II
ACCURACY (%) OF OUR MODELS ON SST DATASET COMPARED TO OTHER MODELS.[1]

Model	SST-2		SST-5	
	All	Root	All	Root
Avg word vectors [9]	85.1	80.1	73.3	32.7
RNN [8]	86.1	82.4	79.0	43.2
RNTN [9]	87.6	85.4	80.7	45.7
Paragraph vectors [2]	–	87.8	–	48.7
LSTM [10]	–	84.9	–	46.4
BiLSTM [10]	–	87.5	–	49.1
CNN [11]	–	87.2	–	48.0
$BERT_{BASE}$	94.0	91.2	83.9	53.2
$BERT_{LARGE}$	**94.7**	**93.1**	**84.2**	**55.5**

[1] Some values are blank in "All" columns because the original authors of those paper did not publish their result on all phrases.

The second paper of related work is "BB twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs" which took 100 million un-labeled tweets and pre-trained with word embedding to be used in the CNNs and LSTMs. It used 10 CNNs and 10 LSTMs with different wright initializations, epochs, filter sizes and embeddings. This got results shown bellow that are better than the rest of the models in each of the five English subtracts and includes comparisons of 38 models for subtract A showing just how great this model performed. This paper showed us the use of CNN and helped us decided to use it in out model. (8)

7.2. "BB twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs"

Figure 16: Results

Subtask	Metric	Rank	BB_twtr submission	Next best submission
A	Macroaveraged recall	1/38	0.681	0.681
B	Macroaveraged recall	1/23	0.882	0.856
C	Macroaveraged mean absolute error	1/15	0.481	0.555
D	Kullback-Leibler divergence	1/15	0.036	0.048
E	Earth movers distance	1/12	0.245	0.269

Table 3: Results on the 2017 test set. The 2017 test set contains 12,379 tweets. For a descripti the subtasks and metrics used, see (Rosenthal et al., 2017). For subtask A and B, higher is better, for subtask C, D and E, lower is better.

The third paper of related work is "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models" which is a language model based on BERT used to for NLPs in the financial domain. This paper used a dropout probability of 0.1, warm-up proportion of 0.2, maximum sequence length of 64 tokens, and trained 6 epochs. It then evaluated on the validation set and chose the best one. This paper concluded that even with a smaller training set and fine-tuning only a part of the model, FinBERT outperforms state-of-the-art machine learning methods. This showed up that once again BERT is an effective model and can be used in a variation of ways and fields. (9)

The final related work is a comparison of a paper that has a similar topic. The paper is "Comparative Studies of Detecting Abusive Language on Twitter" which we found finding abusive language to be similar to finding offensive language. This paper used a variety of models including Naïve Bayes, Linear Regression, Support Vector Machine, Random Forest, Gradient Boosted Trees, CNN and RNN. Our paper also had very similar models of DNN, CNN, Decision Tree, Random Forest, Linear Regression and SVM leading to a very good comparison. So, we can used their results as a similar comparison and as shown below their accuracies for abusive language were all between .78 and .89 with a high of .887 showing a goal for our paper should be around their average of .854 while understanding there is still differences between the circumstances of the papers. We can also look at particular models that both papers used for a more specific comparison. (10)

7.3. "Comparative Studies of Detecting Abusive Language on Twitter"

Figure 17: Results

	Normal			Spam			Hateful			Abusive			Total		
Model	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
NB (word)	.776	.916	.840	.573	.378	.456	.502	.034	.063	.828	.744	.784	.747	.767	.741
NB (char)	.827	.805	.815	.467	**.609**	.528	.452	.061	.107	.788	.832	.803	.752	.751	.744
LR (word)	.807	.933	.865	.616	.365	.458	.620	.161	.254	.868	.844	.856	.786	.802	.780
LR (char)	.808	.934	.866	.618	.363	.457	.636	.183	.283	**.873**	.848	.860	.788	.804	.783
SVM (word)	.757	.967	.850	**.678**	.190	.296	**.836**	.034	.065	.865	.757	.807	.773	.775	.730
SVM (char)	.763	**.968**	.853	**.680**	.198	.306	**.805**	.070	.129	**.876**	.775	.822	.778	.781	.740
RF (word)	.776	.945	.853	.581	.213	.311	.556	.109	.182	.852	.819	.835	.757	.781	.745
RF (char)	.793	.934	.857	.568	.252	.349	.563	.150	.236	.853	.856	.854	.765	.789	.760
GBT (word)	.806	.921	.860	.581	.320	.413	.506	.194	.279	.854	.863	.858	.772	.794	.773
GBT (char)	.807	.913	.857	.560	.346	.428	.472	.187	.267	.859	.859	.859	.770	.791	.772
CNN (word)	.822	.925	.870	.625	.323	.418	.563	.182	.263	.846	.916	.879	.789	.808	.783
CNN (char)	.784	.946	.857	.604	.180	.264	.663	.124	.204	.848	.864	.856	.768	.787	.747
CNN (hybrid)	.820	.926	.869	.616	.322	.407	.628	.180	.265	.853	.910	.880	.790	.807	.781
RNN (word)	.856	.887	.870	.589	.514	.547	.577	.194	.287	.844	**.934**	**.887**	**.804**	**.815**	**.804**
RNN (char)	.606	**.999**	.754	.000	.000	.000	.000	.000	.000	.000	.000	.000	.367	.605	.457
RNN-attn (word)	.846	.898	**.872**	.593	.469	.520	.579	.194	.283	.849	.925	.886	.800	.814	.800
RNN-LTC (word)	**.857**	.884	**.871**	.583	.525	**.551**	.564	**.210**	**.302**	.846	.932	**.887**	**.804**	**.815**	**.805**
CNN (w/context)	.828	.910	.867	.609	.341	.429	.505	**.246**	**.309**	.840	.914	.875	.786	.804	.784
RNN (w/context)	**.858**	.880	.869	.577	**.527**	**.549**	.534	.175	.256	.840	**.937**	.885	.801	.813	.801

8. Conclusion and Future Work

In conclusion, this paper focused on finding the best features and machine learning models for detecting offensive or non-offensive from tweets For this, we implemented three feature extraction models: TF-IDF, and DistilledBERT, and six classification models: DNN, CNN, DT, RF, LR, and SVM (linear, RBF kernels) as a preliminary experiment. For the pre-processing tasks, POS tagging and features selection were applied before model training. As a result, we obtained the best accuracy of **0.84 and F1 of 0.79** by using TF-IDF with importance features selection. We found that sentiment features like Vader negative (negative sentiment) shown in Fig. 14 contribute to classifying non-offensive and offensive tweets from feature importance analysis. The appearance of some bad words like asshole, stupid also affects the performance. Unlike the case, POS-based tagging does not have high importance. As other findings, It was found that statistics-based features like TF-IDF outperformed neural-network-based features like BERT in the case of using small data set shown in Table 2. Also, we found that the performance of the machine learning model could be reduced if they contain a large feature set shown in Table 3. From those results, it was obvious that reducing features by preserving informative features improves their performance.

Due to the small sample of data, we cannot enhance the model future with advanced methods like LSTM and transformers. Future research will be collecting a larger size of data and fine-tuning

transformer models. However, with our current performance results, it is adequate to deploy it into application.

References

[1] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[2] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[3] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[5] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Chris Pal. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Manish Munikar, Sushil Shakya, and Aakash Shrestha. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5. IEEE, 2019.

[8] Mathieu Cliche. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*, 2017.

[9] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

[10] Younghun Lee, Seunghyun Yoon, and Kyomin Jung. Comparative studies of detecting abusive language on twitter. *arXiv preprint arXiv:1808.10245*, 2018.