Spring 5-22-2019

TSAR : A System for Defending Hate Speech Detection Models Against Adversaries

Brian Tuan Khieu
San Jose State University

TSAR : A System for Defending Hate Speech Detection Models Against Adversaries

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Brian Tuan Khieu

May 2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Although current state-of-the-art hate speech detection models achieve praiseworthy results, these models have shown themselves to be vulnerable to attack. Easy to execute lexical manipulations such as the removal of whitespace from a given text create significant issues for word-based hate speech detection models. In this paper, we reproduce the results of five cutting edge models as well as four significant evasion schemes from prior work. Only a limited amount of evasion schemes that also maintain readability exists, and this works to our advantage in the recreation of the original data. Furthermore, we demonstrate that each lexical attack or evasion scheme can be overcome with our new defense mechanisms with some reducing the effectiveness of the scheme to 1%. We also propose a new evasion scheme that outperforms the those in previous work along with a corresponding defense. Using our results as a foundation, we contend that hate speech detection models can be defended against lexically attacked data without the need for significant retraining.

ACKNOWLEDGEMENTS

I would like to sincerely thank Professor Melody Moh, my project advisor, for the enormous amount of help and guidance she bestowed upon me. Without her help or her belief in my abilities, this project never would have come to fruition. I would like to thank Prof. Robert Chun, my committee member, for his support and advice. I would also like to thank Professor Teng Moh for his many helpful suggestions and feedback regarding the project. His advice steered this project towards the correct direction and helped me catch some issues I may have overlooked otherwise.

I would like to thank all of the friends I have made at San José State University for the help and advice they gave me on using machine learning tools.

And finally, I want to thank my family for their continuous support and encouragement throughout my studies.

I.Introduction

Armed with the power of anonymity, many groups and individuals abuse social media tools in order to proliferate messages advocating for violence and suppression of another people. Such language and its spread across online social networks (OSNs) presents several issues to the managers of such networks, but there is an inherent issue with rooting out such speech. The definition and classification of hate speech has not reached a universal consensus. Merriam-Webster designates hate speech as language "expressing hatred of a particular group of people" while the European Union defined it as speech that "spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance". There is a notable distinction between the two definitions above; the latter places an emphasis on hateful "calls to action" while the former does not. This difference, the "call to action", typically is the borderline between merely offensive speech and hateful speech in the eyes of OSN moderators and admins. Even with this boundary, the problem of classifying hate speech remains a troublesome one since declaring whether one has crossed the border is still murky. Regardless, the identification of hate speech a persistent problem for OSN moderators, and due to the sheer volume of generated content on said networks, manual moderation is infeasible.

Several approaches towards detecting such language have been proposed and recently developed in order to combat the rise of hate speech. With the spread of social networking websites, it has become easier than ever to broadcast one's opinions on whichever topic one may choose. While the quick dissemination of information through such sites can elicit much good, in irresponsible or scheming hands, such power can bring about great division and anguish. One such example of the harm that can come about is the birth of echo chambers on the internet;

misguided or misinformed people can find themselves trapped in a vicious cycle of ingraining more and more radical and polarizing sentiments [1].

Hate speech and its prevalence in online social networks have proven to be an ongoing problem on social media sites such as Facebook and Youtube. While manual user flaggings of comments or posts can help, the process can be abused to silence opinions one disagrees with. With the constant stream of content generation, simply employing an army of moderators will not solve the issue either. Thus, there is a need for an effective and automated system for identifying hate speech.

Unfortunately, sources of hate can undertake actions to morph their text to evade detection thus complicating the problem. The inherent goal of morphing text is to introduce enough noise so that detection models cannot correctly classify phrases and passages as "hateful" while also maintaining the meaning and readability of the original message. Prior work has shown that simple modifications such as the removal of whitespace can decimate the accuracy of state-of-the-art models and that there is a need for some method of defense against these evasion schemes.

We replicate the results of five cutting edge model architectures from four papers as well as four lexical attacks presented in a separate paper. In addition, we propose one new attack that we demonstrate to be more effective than all prior attacks. We show that we can construct a defense that significantly mitigates the effect of each evasion scheme despite alterations to the attack method's parameters. This suggests that evasion schemes are inherently limited by the need for preserved readability, and one can leverage this property to build appropriate countermeasures. Our best defenses can all be categorized as a "pre-processing" stage defense; there is significant difficulty in training hate speech detection models on adversarial data. While

we did include one defense of the latter category, we demonstrate the stark performance difference in the two methods and propose that “pre-processing” stage defenses be built to address any evasion schemes encountered.

The remainder of this paper is structured as follows: in Section II we cover related work and present our motivations. In Section III provide a problem statement and subsequently propose a solution in Section IV. Section V details and discusses our experimental settings and results. In Section VI, we conclude this paper and discuss avenues of future work.

II.Related Work

One method of identifying hate speech is to use a rule-based approach where certain negative words are always flagged to indicate a need for further inspection. Certain words are statistically identified to appear in manually identified hate speech more than others, and they are subsequently added to a ruleset to follow. Classification of hate speech is determined by the construction of a dictionary compiling all “hate” words [2] or through a binary classification of “benign” or “hate” types [3]. Unfortunately, such approaches are somewhat naive and ill-equipped to handle slang and symbolism. It should be noted that some of these rule-based approaches are used in conjunction with other methods to form a more robust solution.

The more generally accepted method of identifying hate speech is the use of machine learning and deep learning algorithms. This approach more readily handles slang and symbolism since the models will be trained upon a dataset that includes such words and phrases. Machine learning and deep learning models built for hate speech detection can fall into one of two categories, word-based and character-based models. Word-based models rely on extracting features from n-grams of different tokenized word combinations while character-based models

do so from n-grams of characters. Word-based models can also utilize rule-based techniques and factor in a word's sentiment or connotation.

Watanabe et al [4] used unigrams along with extracted patterns to develop dictionaries. They used these dictionaries paired with sentiment, semantic and polarity features with a machine learning algorithm, "J48graft" to classify speech into binary and ternary classifications. The researchers made the distinction between hate speech and offensive speech; the ternary classifier used "hateful", "offensive", and "clean" labels.

Rutwandika et al [5] experimented with supervised and unsupervised learning techniques and a variety of different features for hate speech detection. Unlike [4], the researchers focused on binary classification. Amongst the methods , naive bayes, logistic regression, svm, decision tree, and k-means, naive bayes achieved the highest f-1 score when using tf-idf features. [5] also noted the ineffectiveness of K-means clustering in this problem space; it ranked the worst in performance in nearly every scenario.

Grondahl et al. [6] utilized new hate speech detection models from four papers [7][8][9][10] as the basis for testing new evasion schemes. While the five models used from [7][8][9][10] performed well on their own datasets, [6] noted that performance dropped once each model trained upon all datasets. Grondahl et al. noted that hate speech detection models fail to take into account the existence of adversarial examples. Current state of the art hate speech detection models only consider naive examples that originate from entities that do not attempt to avoid detection. However, the real world is filled with adversaries who wish to spread hate speech on a platform while going undetected.

III. Literature Survey

A. Current Methods and Features

i. Methods

One method of identifying hate speech is to use a rule-based approach where certain negative words are always flagged to indicate a need for further inspection. Certain words are statistically identified to appear in manually identified hate speech more than others, and they are subsequently added to a ruleset to follow. Classification of hate speech is determined by the construction of a dictionary compiling all "hate" words or through a binary classification of "benign" or "hate" types. Unfortunately, such approaches are somewhat naive and ill-equipped to handle slang and symbolism. It should be noted that some of these rule-based approaches are used in conjunction with other methods to form a more robust solution.

The more generally accepted method of identifying hate speech is the use of machine learning and deep learning algorithms. This approach more readily handles slang and symbolism since the models will be trained upon a dataset that includes such words and phrases. Machine learning and deep learning models built for hate speech detection can fall into one of two categories, word-based and character-based models. Word-based models rely on extracting features from n-grams of different tokenized word combinations while character-based models do so from n-grams of characters. Word-based models can also utilize rule-based techniques and factor in a word's sentiment or connotation.

ii. Definition

There is no agreed upon standard for hate speech detection, several papers contain significant departures from one another in terms of definitions, classes, features, and architecture used. This complicates the problem of hate speech detection while underlining how it remains an actively worked upon challenge.

In the case of definitions, there is no consensus upon what hate speech actually is. The generic idea consists of language expressing a large degree of hatred of a people, but it becomes tricky differentiating it from offensive speech. As noted before, the European Union draws the line at the "call to action", speech is not considered hateful until it advocates for the suppression or violence against others. For those who utilize this distinction for the categorization of speech, the problem turns into a ternary classification situation with benign, hate, and offensive speech labels [8]. Not everyone abides by this definition however, and in those instances, the line between hate speech and offensive speech blurs. In other cases, hate speech is divided not by the severity or by the existence of a "call to action" but rather the areas for attack such as racism and sexism [7]. Thus, different hate speech detection models do not necessarily compare directly to one another due to the variations in definitions and classes used. It follows that with different classes or labels used, these models use quite different datasets as well. With these variations, the direct evaluation and comparison of different hate speech detection models is difficult.

iii. Features

There is a common thread of features used in addressing hate speech detection. The most commonly used features consist of Term Frequency Inverse Document Frequency (TF-IDF), bag of words vectors, n-grams of characters or words, Global Vectors for Word Representation

(GloVe),. From each of these features, the statistical properties and makeup of hate speech can be determined. Nobata et al. [11] found that these statistical based features are more effective in classifying speech than semantic features such as polarity and sentiment. This may be due to how sentiment based features can be mixed within both benign and hate speech. Benign speech can contain words that register negatively in sentiment while hateful material can contain words that register positively. Contrastingly, statistical based features make class associations based on the appearance of a combination of words or characters within a test class. Despite this issue, sentiment based features are still commonly paired with statistical based features as seen in Gao and Huang [12].

a. TF-IDF

TF-IDF or Term Frequency Inverse Document Frequency is a numerical statistic that reflects the importance of a word or term in a given document. This statistic is primarily based upon the frequency with which a term appears within a given document; if the term appears frequently, it presumably holds more importance than terms that do not. Notably, the second portion of the term, Inverse Document Frequency, reflects that the weight of the frequency of occurrence is inversely proportional to the amount of times the term appears in other documents. In short, a term may appear quite frequently while holding no real significance, and one such example is a common word “the”. This statistic has been used in many hate speech detection models, ones that are primarily lexical based [5][8][9].

b. Bag of Words

Bag of Words is a representation of text using frequencies with an assortment of words. This is a surface level feature that reflects the statistical makeup of a given piece of text. It has

been used in several works, primarily lexical based models but also some neural network based models [5][9].

c. N-grams

In a broad sense, n-grams refer to continuous sequences of n items from a document or given piece of data. Specifically for hate speech detection, these sequences consist of letters or words. N-grams are utilized by many different hate speech detection models, but in the case of neural networks, they are typically used to learn word embeddings [7][9][11]. Character based n-grams are suited for handling typos due to majority of n-grams being preserved even with typos. On the other hand, word based n-grams are better at capturing implicit hate as well as less computationally intensive to generate. Word based n-grams look to be more popular overall than character based n-grams for hate speech detection models.

d. GloVe or Word Embeddings

This type of feature consists of representing terms or more usually words as vectors. These vectors are numerical representations of words or terms in a vector space and are used to denote whether certain words are closely associated with one another or not. Word embeddings are primarily used by neural network hate speech detection models since a neural network is usually required to create them in the first place. Several pre-trained embeddings are available and used by several papers for their neural networks [9][10][12]. Of note is that these embeddings rely heavily on proper tokenization, and any variations in spelling will cause a new vector to be created instead of modifying the original term's vector.

e. Semantic Features

While statistic based and surface level features have been used to great effect, models incorporating these as well as semantic features have shown performance improvements [12]. Semantic features typically revolve around the positive and negative connotations of words or polarity. Some of these rely upon a pre-built lexicon dictionary to judge whether a given word is good or bad. It follows that semantic features are useful in aiding hate speech detection because by nature hate speech should contain more words with negative polarities than benign text. Gao and Huang [12] showcase the usefulness of semantic features by making chiefly utilizing both the polarity and a lexicon dictionary in their hate speech detection model along with other statistical based features. Of note is that semantic features can overlap with one another while coming from different classes. As an example, hate speech can contain many words with a positive polarity while expressing a passionate fervor for the condemnation of another group of people. Meanwhile, benign text that consists of complaints for ordinary problems will contain many words with negative polarities. This "cross contamination" signifies that semantic features are not usually linearly separable by class; to our knowledge, no hate speech detection model uses only semantic features. Semantic features are quite useful in supplementing statistical feature based models but not that useful on their own.

B. Neural Network Hate Speech Detection Models

Neural Network based methods use neural networks to learn abstract feature representations of test hate speech data through several stacked layers. Input features consist of task-specific embeddings learned using FastText, CNNs and LSTMs and other forms of feature encoding. The goal of these neural network based approaches is to learn new abstract feature

representations from simple input text. Some of the popular methods used in this category consist of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks. For the most part, this approach relies upon character or word n-gram one hot encodings.

Wuclzyn, Thain, and Dixon [7] used a multilayer perceptron (MLP) models, a type of artificial neural network, in addition to logistic regression models to handle the binary classification problem. They noted that the MLP models were better at detecting implicit hate speech as opposed to the logistic regression models.

Badjatiya, Gupta, Gupta, and Varma [9] treated the hate speech detection issue as a binary classification problem and used a variety of deep neural networks to attempt to address it. Amongst the various features, tf-idf, bag of words vectors and embeddings learned, and models, FastText, CNNs, and LSTMs, LSTMs using random embeddings and Gradient Boosting Decision Trees (GDBT) performed the best. The authors reasoned that this may be due to the inherent nature of LSTMs. LSTMs are a subset of RNNs and both retain memory of past events through use of an internal state. However, LSTMs do not suffer from an issue that RNNs do that prevents them from propagating information far into the future. The authors noted that the random embeddings version outperformed the pre-trained versions which may be due to the need for backpropagation for the associated GDBTs to learn embeddings for the task [9].

Contrastingly, Zhang, Robinson, and Tepper [10] used a combination approach of CNN and RNN combination architecture to classify hate speech. Specifically, a Gated Recurrent Unit (GRU), a type of RNN, was used in conjunction with a CNN. Recurrent Neural Networks suffer from the vanishing gradient problem which prevents the propagation of information into the distant future. Gated Recurrent Units were developed to address the issue and are often paired

with Convolutional Neural Networks, networks that apply convolution and pooling operations. In this case, the purpose is to have the CNNs extract co-occurring word n-grams as patterns while having the Gated Recurrent Units retain past pattern information to give context. The authors concluded that while both LSTM networks and the combination of CNNs and GRUs can identify hate speech comparably well to one another, the training time for the combination is significantly less than that of the LSTMs [10].

C. Context Aware Hate Speech Detection Models

A new type of approach to hate speech detection involves the leveraging of context as a semantic feature. Hate groups often attempt to avoid detection by the use of words with hidden meaning. In addition, current events give background information that can turn seemingly innocuous text into hate speech once considering the context. There is relatively a small amount of work done in this niche area compared to the generic version of hate speech, but some papers have identified it grondahl et al. as an area for future work.

To our best knowledge, only one significant work has been published in the area of context aware hate speech detection. Gao and Huang [12] define context as any information not included within the original text itself. In addition to typical word and character n-grams, Gao and Huang utilized lexicon derived features such as polarity and emotion as well as context features consisting of usernames and article titles. The authors created a corpus of Fox News comments that were annotated according to strict guidelines; this included context features and is purported to be the first of its kind.

Gao and Huang [12] tested both logistic regression and bi-directional LSTM models while treating hate speech detection as a binary classification problem. They found that in both models' cases, there was an improvement in F-1 score by using the additional context

information. The authors noted the pros of both model approaches. Logistic regression makes good use of character n-grams which helps with capitalizations and misspellings while LSTMs can capture the hidden and subtle meanings of implicit hate speech. To maximize usage of both models strengths, the authors combined the two into an ensemble model which yielded significant performance increases over the use of a single model.

IV. Problem Statement

Grondahl et al. [6] created six different evasion schemes to morph text; these schemes reliably diminished the effective of each detection model while preserving the meaning of the original text. The six attacks proposed fall into one of three categories: word modifications, whitespace manipulation, and benign word insertion. Word modifications are simple changes to the hate speech text, and they include simple typos or deterministic LEETSPEAK where letters are exchanged for numbers. Whitespace manipulation revolves around the removal or insertion of spaces within a given text to throw off word-based models. Finally, benign word insertion is the addition of normal or positive words not normally associated with hate speech, such as "love", in order to fool the model. These attacks require essentially no training for the average person to use, and [6] found that the combination of whitespace removal and benign word insertion zeroed out the F1 scores for many of the state-of-the-art models. The following passages detail the attacks addressed in this paper along with our newly proposed attack.

The typo attack used in [6] consists of a single swapping of letters within the middle of a word. This was done to more effectively trick spell checkers while also maintaining readability and the original meaning of the text. The preservation of readability and meaning must be taken into account for the creation of any lexical attack; fooling detection models can be done easily if

the new text loses all meaning. In this case, the evasion scheme relies on previous cognitive research that determined single character swaps have the least impact on retaining readability.

Whitespace removal converts a given piece of text into one single chunk of characters. This attack greatly hinders the effectiveness of word-based models due to their reliance on proper tokenization of words. Word-based models rely on word-embeddings; if fed improperly tokenized data, these models fail to leverage these embeddings. In this case, such models treat text with removed whitespace as new words with no particular associations thereby allowing adversaries to evade detection. Conversely, character-based models better handle this issue; removing whitespace does cause issues in terms of the separation of words, but the majority n-gram character combinations remain the same.

Benign word insertion attacks, more specifically the word "love", involve the random placement of a positive or neutral word into a text. [6] notes that this attack affected both word-based and character-based models comparably. The effectiveness of this evasion schemes comes from an asymmetric problem regarding the insertion of material. Inserting hateful material into a benign or positive piece of text results in the creation of hateful material while the reverse, inserting positive material into hateful material, does not yield positive material. However, detection models do not take this into account and thus significantly decrease in performance when faced with the evasion scheme.

Our newly proposed attack, character boundary appending (CBA), adds patterns of random letters to the edges of a word. This attack maintains a high amount of readability due to humans' ability to effectively parse out original word as can be seen in the following example:

"Jack and Jill went up the xxxhillxxx"

This attack can be extremely customized with a length of the patterns, pool of letters to draw from, and the balance of the attack. Balance in this case refers to the ratio of letters to append to the left of the word versus that of the right of the word. In the interest of maintaining as much readability as possible, patterns of multiple consonants are used to minimize the chance of a random word being created within the appendings.

In summary, hate speech detection models are not well equipped to handle adversarial examples originating from entities that wish to evade detection. These attacks are effective and easy to execute. We aim to address this security gap of these models and propose several new methods to aid these detection systems against evasion schemes.

V. Proposed Solution

The following sections cover the details of proposed defenses against the aforementioned attacks. We begin with a brief overview of the datasets and the models used. Then, we cover our proposed defense schemes and attack replication process.

A. Models and Datasets

This section describes the models and datasets used in the replication the work conducted by Grondal et al. [6]. The models, named W, T1, T2, and T3, and datasets are sorted by which paper they originally came from.

Wulczyn et al. [7] utilized the W dataset comprising of Wikipedia edit comments; these comments were separated into personal attacks and ordinary speech. These researchers used both multilayer perceptron (MLP) models and logistic regression (LR) with n-gram features. The n-gram features were of both character and word types; the models with the former outperformed

those of that latter which is why [6] chose to use both MLP and LR character models over their word-based counterparts. The MLP and LR models are the only character-based models used in both our experiments and those of [6].

Davidson et al. developed the T1 dataset which labelled speech in one of three classes, hate speech, offensive, and ordinary. The text comprises of tweets gathered from hatebase.org and gathered using searches for common phrases associated with hate speech. [6] conflates the two labels, hate speech and offensive, when using the T1 dataset due to the lack of ternary classification capability of every detection model used. Our use of the T1 dataset was modified to follow that of [6], but for clarity's sake we only will refer to the modified dataset as T1 and make no mention of the unmodified version. Davidson et al. [1] created a LR word-based model to perform ternary classification with great success.

Badjatiya et al. worked with the T2 dataset which also contained three class labels, but conversely to that of the T1 dataset, these labels consisted of "sexism", "racism", and "neither". Similar to the T1 dataset, we replicated the conflation of the sexism and racism labels done by Grondahl et al. [6]. The T2 dataset comprises of tweets as well, and the Badjatiya et al. classified them using a long short-term memory (LSTM) network. LSTM models are particularly useful in text classification due to their ability to retain information from the past better than recurrent neural networks (RNNs) do. RNNs tend to lose information that is too far away due to the vanishing gradient problem.

Zhang et al. utilized both the T1 as well as their own new dataset, T3. In addition, the researchers also combined the hate speech and offensive labels into one class. The T3 dataset contains about 2400 tweets with the hateful portion specifically attacking muslims and refugees. Zhang et al. created a model based on a mix of convolutional neural networks (CNNs) and

RNNs. CNNs are typically useful for extracting features while RNNs retain information from the past. The team used gated recurrent units (GRUs) which are a type of RNN that does not face the vanishing gradient issue that normal RNNs do. The CNN and GRU hybrid model approach achieves comparable results to a LSTM, but they only use up a fraction of the resources a LSTM requires.

B. Proposed Defenses

We propose defenses that fall into one of two categories, pre-processing and retraining. Pre-processing defenses modify incoming data before they reach the model and attempt to recreate the original text. Contrastingly, retraining addresses text morphs through training the model on pre-attacked data. To keep things fair, we design our defenses to be as generic as possible to simultaneously address variations on the attack.

C. Pre-processing Defenses

The first pre-processing defense we propose, word segmentation no redo (WSNR), addresses the whitespace removal attack through separating the one large chunk of words back into separate words. Word segmentation, the separation of words without whitespace, has been thoroughly studied in itself, and many python libraries that support this function are available. Note that the api used to carry out this defense significantly impacts results; given the nature of the data, word segmentation functions may yield unsatisfactory results if they cannot deal with typos.

Our second proposed defense, word balance no redo (WBNR), reverses the typo attack described beforehand. Since the typo attack creates an error in two letters of a word, typical spell

checkers do not properly rectify the mistake [6]. Since we know the effects of the typo attack, we determined that we only need to swap back the letters involved. The algorithm is defined as follows:

1) Match the attacked word to entries inside a dictionary based on the same length of characters
2) Of those matching words, select those with the same outside characters
3) Of those matching words, select those with the same ascii values
4) Of the remaining, select the word that differs with the attacked word by one swap of letters. If no word can be found, return failure status.

Our proposed defense can handle any variations on the typo attack so long as they do not swap more than three letters or modify the outside letters. However, we do not believe that an attack that violates the above two principles would retain a high level of readability.

Thirdly, we present a defense, good grammar no redo, that defends against perhaps the most difficult attack to detect, benign word insertion. The defense first comprises of flagging any words that violate grammar rules. This is done to yield a list of words for polarity checking and is effective versus the benign word insertion attack due to the inherent properties of word insertion. Insertion of a word into a sentence, especially non-adjectives and non-adverbs, most likely cause a break in grammatical sense. In the event that the insertion of the word does not cause a violation of a grammatical rule, the meaning of the text will then change. Since we know that the benign word insertion attack will insert a word of neutral or positive connotation, we can address it by removing the words with such a polarity from the list of words compiled beforehand.

Our fourth proposed defense, vowel search no redo (VSNR), addresses our newly created attack, character boundary appending. The tricky part in addressing this attack is derived from its large amount of customization; an attacker can easily switch between appending characters in a balanced manner to appending them only on one side while not losing readability and meaning. In order to address this attack, we factored in the limitations of text morphings, the need for preservation of readability and meaning. Given these limits, we know that characters appended to a word must not create a new word lest it violate the aforementioned clause. Apart from this, we also know that every word in the english language, barring some exceptions such as “tv” and “dr.”, has a vowel. Thus, we formed our VSNR defense to search for words that can be found in a dictionary based on the position of vowels within an attacked text. Since the characters appended should not form a word either within themselves or in conjunction with the original text, we do not have to worry about the presence of any newly created words. Thus, the longest word found with based on vowel positional searching should be the original word itself.

D. Training Defenses

The first retraining defense we propose attempts to address the whitespace removal attack. Whitespace removal training (WSRT) involves training the models on text data with whitespaces already removed. Any new test data must also have any whitespaces removed from the text, even if the test data has not been attacked. Failure to do so will cause a performance decrease since the models are not trained on clean data and thus do not recognize whitespaces properly. This defense is only applicable to character-based models due to the issues with word tokenization for word-based models. An initial test backed these claims which is why we did not bother applying this defense to non-character-based models.

Our second retraining defense, clean spelling training (CST), addresses the typo attack. This defense is more of a mix of pre-processing and training as opposed to just training. The defense involves flagging all words that are unable to be recognized, mapping them to one word with correct spelling, and compiling this all into a dictionary. Then, the model is trained upon the training data that has been preprocessed to switch any incorrectly spelled words to their one word based upon the dictionary built previously. Any test data will also have any words found in the dictionary replaced with the singular correct spelling.

VI. Performance and Results

We performed four attacks on each of the model-dataset combinations; with seven total combinations, this yielded 28 attacks. In addition, we used two variations of our newly proposed attack along with two combination attacks to end with a total of 56 attacks. For each attack, we applied the appropriate defenses in separate runs.

Due to ethical concerns, [6] decided not to release any of the code associated with the generation of attacks. Thus, we recreated the attacks empirically and selected the settings that yielded the most similar results to those found in [6].

Our experiments were conducted using the sci-kit learn python package and keras, a tensorflow api. We used 5-fold cross validation and applied attacks to the given datasets before they were fed into the pipeline to the models themselves. Any pre-processing defenses were run in the pre-processing stage, and similarly, any training defenses had the model-dataset combinations trained on the pre-attacked data before receiving any test data. As in [6], we used a macro averaged F-1 score as our performance metric of choice. The following sections detail the experimental results based on the attack being addressed.

A. Typos

TABLE I. Macro Averaged F-1 Scores for Model, Dataset Combinations under Typo Attack and Applied Defenses

Model, Dataset	Orig.	Typo	WBNR vs Typo	CST vs Typo
LR Char, W	0.76	0.62	0.76	0.71
LR Word, T1	0.49	0.29	0.48	0.45
MLP Char, W	0.74	0.58	0.73	0.68
CNN + GRU, T1	0.44	0.32	0.44	0.38
CNN + GRU, T2	0.78	0.24	0.77	0.71
CNN + GRU, T3	0.70	0.27	0.69	0.53
LSTM	0.70	0.42	0.70	0.63

While typo attacks can significantly impact the performance of hate speech detection models, apply either the pre-processing or training defense can significantly reduce the effectiveness of the attack. In this case, the WBNR defense outperforms the CST defense greatly. This can be attributed to how the WBNR defense recreates the original input data with great reliability while the CST defense may encounter new typos in the training data that it has not built an entry for.

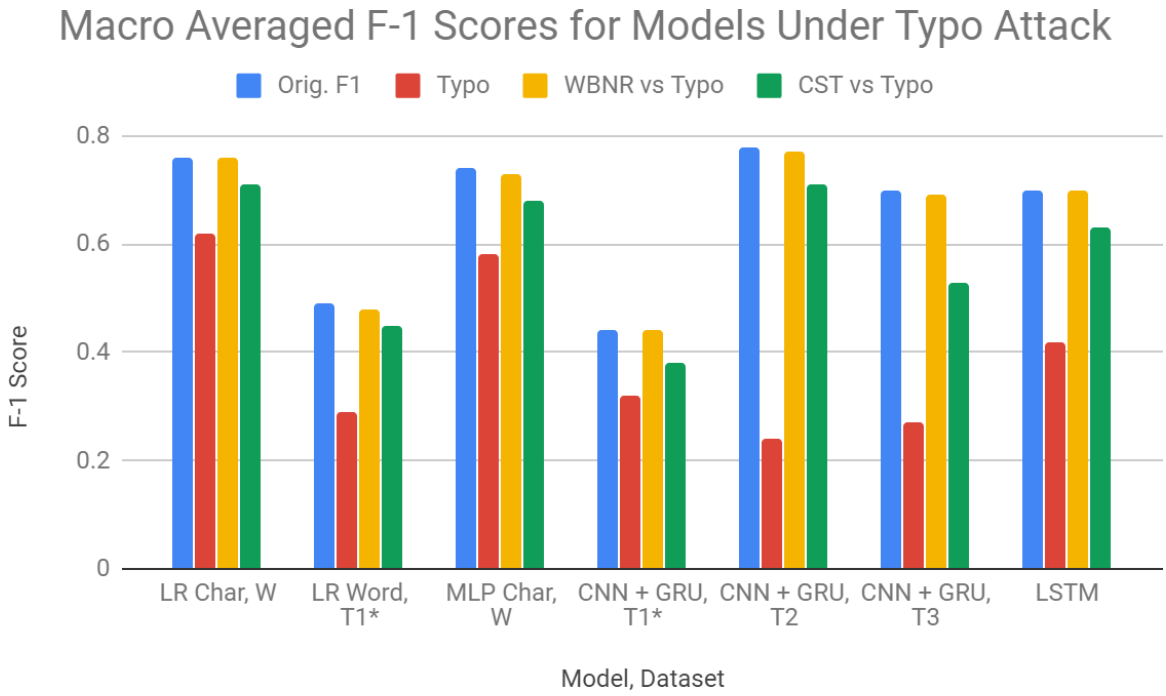

Figure 1. Graph of Macro Averaged F-1 Scores for Model, Dataset Combinations under Typo Attack and Applied Defenses

B. Whitespace Removal

TABLE II. Macro Averaged F-1 Scores for Model, Dataset Combinations under Whitespace Removal Attack and Applied Defenses

Model, Dataset	Orig.	WS Removal	WSNR vs WS Removal	WSRT vs WS Removal
LR Char, W	0.76	0.58	0.62	0.64
LR Word, T1	0.49	0.00	0.44	-
MLP Char, W	0.74	0.56	0.66	0.65
CNN + GRU, T1	0.44	0.00	0.40	-
CNN + GRU, T2	0.78	0.00	0.68	-

CNN + GRU, T3	0.70	0.01	0.64	-
LSTM	0.70	0.00	0.67	-

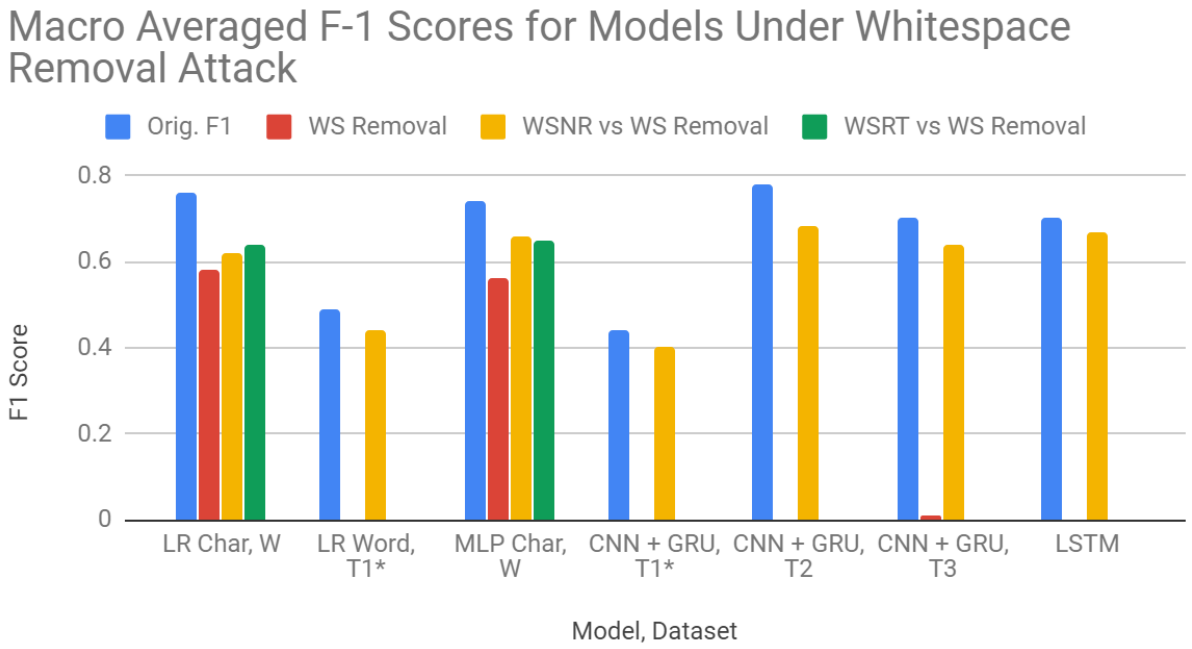

Figure 2. Graph of Macro Averaged F-1 Scores for Model, Dataset Combinations under Whitespace Removal Attack and Applied Defenses

Whitespace removal completely breaks all word-based models while also greatly diminishing the character-based models' performance. [6] suggests that the character-based models lose access to all n-grams concerning whitespaces as the reason why character-based models suffer performance issues. Regarding the word-based models, correct tokenization is key for word-based models to properly utilize their pre-trained embeddings. The removal of whitespace zeroes out an essential part of these models; in the case of the CNN + GRU model, we observed the embedding matrix to only consist of 0's.

Unlike the case of typos, the training defense outperformed that of the pre-processing one. However, the performance increase was not substantial; the word segmentation defense yielded F-1 scores within 2% of the training defense's. Note that we chose to only apply the WSRT defense to the character based models since word-based models will fail when using whitespace removed data. The WSNR defense performs admirably well and holds an advantage over the WSRT defense in that it is applicable to any model regardless of whether it is word-based or character-based. We utilized several different word segmentation apis and found that results differed greatly due to natural typos encountered. The symspell api is able to address spelling errors and yielded the best results which is why we chose it for our final WSNR implementation. We conclude that our defenses against this attack can significantly reduce the its effectiveness although we do note that the pre-processing version could yield better results with a better word segmentation api.

C. Benign Word Insertion

TABLE III. Macro Averaged F-1 Scores for Model, Dataset Combinations under "Love" Insertion Attack and Applied Defenses

Model, Dataset	Orig.	"Love"	GGNR vs "Love"
LR Char, W	0.76	0.55	0.74
LR Word, T1	0.49	0.43	0.47
MLP Char, W	0.74	0.51	0.72
CNN + GRU, T1	0.44	0.02	0.40
CNN + GRU, T2	0.78	0.51	0.73
CNN + GRU, T3	0.70	0.17	0.68
LSTM	0.70	0.14	0.68

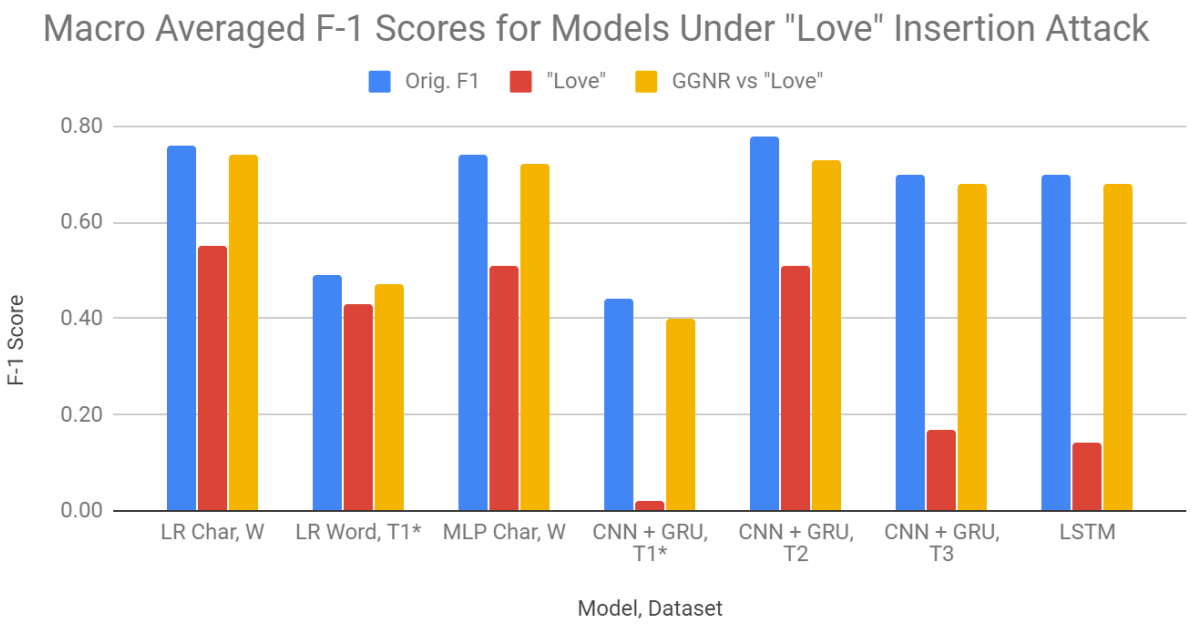

Figure 3. Graph of Macro Averaged F-1 Scores for Model, Dataset Combinations under "Love" Insertion Attack and Applied Defenses

The GGNR defense performed extremely well in addressing the benign word insertion attack. We chose to use "love" as our word of choice due to its nature as not being associated with "hateful" speech. We did not develop a training defense for this attack due to the customizability of the insertion attack. A model would have to be trained on every non-hate word, but doing so would cause all such words to lose proper meaning. Specifically training versus a "known" word such as "love" was deemed to be unfair as it would require us defenders to have too much knowledge of the attack and defeat our goal of making these defenses as generic as possible.

D. Character Boundary Appending

TABLE IV. Macro Averaged F-1 Scores for Model, Dataset Combinations under Different CBA Attacks and Applied Defense

Model, Dataset	Orig. F1	CBA (I, 2)	CBA (I, 2)	CBA (B, All)	VSNR vs CBA (B, ALL)
LR Char, W	0.76	0.53	0.64	0.13	0.74
LR Word, T1*	0.49	0.39	0.39	0.00	0.47
MLP Char, W	0.74	0.58	0.68	0.15	0.71
CNN + GRU, T1*	0.44	0.36	0.36	0.00	0.42
CNN + GRU, T2	0.78	0.58	0.58	0.00	0.77
CNN + GRU, T3	0.70	0.45	0.45	0.00	0.7
LSTM	0.70	0.39	0.39	0.00	0.69

Our proposed character boundary appending attack outperformed all other attacks under the correct customization. The CBA affected both character-based and word-based models and markedly decreased the F-1 scores regardless of settings. The LSTM model took the largest hit in performance with a 30% drop in F-1 score. For word-based models, the CBA performed equally whether it had an imbalanced or balanced setting. We attribute this to the word-based models not being concerned so much by how the attacked words were structured but rather the existence of an unknown word. Contrastingly, character-based models performed worse with the balanced version of the CBA attack. We suggest this may result from a how the n-grams concerning the affected word's boundaries are no longer usable in the case of a balanced attack. A balanced attack completely surrounds the word with characters while an imbalanced attack has the

potential to append only to one side. When applied to all words, the attack zeroes out all word-based models and reduces the F-1 score of character models by more than 50%. This version of the attack floods the input text with too many characters for the character-based models to properly build classifiers. The random nature of the appendings force any word-based models to be unable to use any embeddings, because the likelihood of the same word being appended in the same manner is quite small.

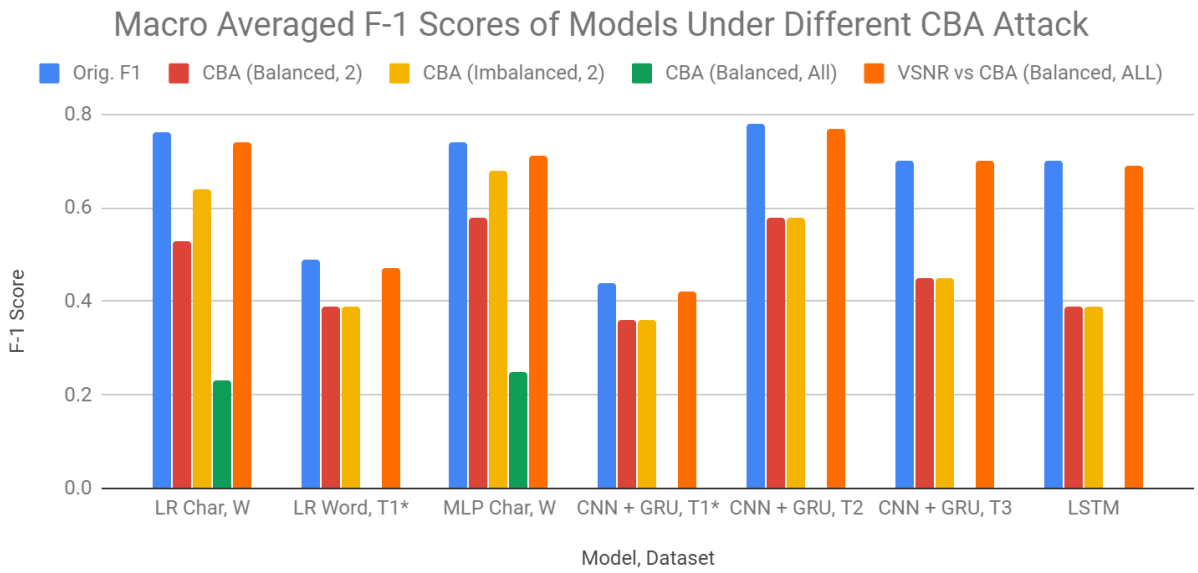

Figure 4. Graph of Macro Averaged F-1 Scores for Model, Dataset Combinations under Different CBA Attacks and Applied Defense

The defense we built for this CBA attack performed very well and nearly completely reversed the effects of the CBA attack. VSNR worked regardless of the settings for CBA, and we argue that this highlights the importance of generality in a defense. This defense reduced the effectiveness of CBA to a maximum of 3% in the case of the multilayer perceptron model. Similarly to the benign word insertion defense, we did not create a training defense for this attack due to its random nature.

E. Combined attacks

TABLE V. Macro Averaged F-1 Scores for Model, Dataset Combinations under Combination Attacks

Model, Dataset	Orig. F1	Whitespace Removal + Love	Whitespace Removal + CBA (B, All)
LR Char, W	0.76	0.51	0.04
LR Word, T1*	0.49	0.00	0.00
MLP Char, W	0.74	0.49	0.02
CNN + GRU, T1*	0.44	0.00	0.00
CNN + GRU, T2	0.78	0.00	0.00
CNN + GRU, T3	0.7	0.01	0.00
LSTM	0.7	0.00	0.00

We combined two of our strongest attacks at our disposal, CBA and whitespace removal, and compare it to the strongest combined attack from [6]. The pairing of whitespace removal and our new attack nearly zeroes out the F-1 scores in both word-based and character-based models thereby outperforming the “love” insertion and whitespace removal combination attack.

Use of a combination of attacks allows us to simultaneously affect two properties that hate speech detection models rely upon, word tokenization and statistical makeup. In the case of word tokenization, removing whitespaces completely breaks word-based models as seen in Table 2. The evasion scheme causes word-based models to lose access to previously generated word embeddings since each new affected text will be treated as an unknown word. CBA further exacerbates this issue for word-based models by appending random patterns to the outsides of words. Furthermore, CBA significantly alters the statistical makeup of affected text. By doing so,

the n-grams used by character-based models build false associations to a class thereby introducing larger errors. In addition, the removal of whitespaces is suggested by [6] to have further adverse effects on these n-grams. N-grams including whitespaces are used to signify the starts and ends of words, and the removal of whitespace significantly impedes a model's ability to determine these boundaries that are large factor.

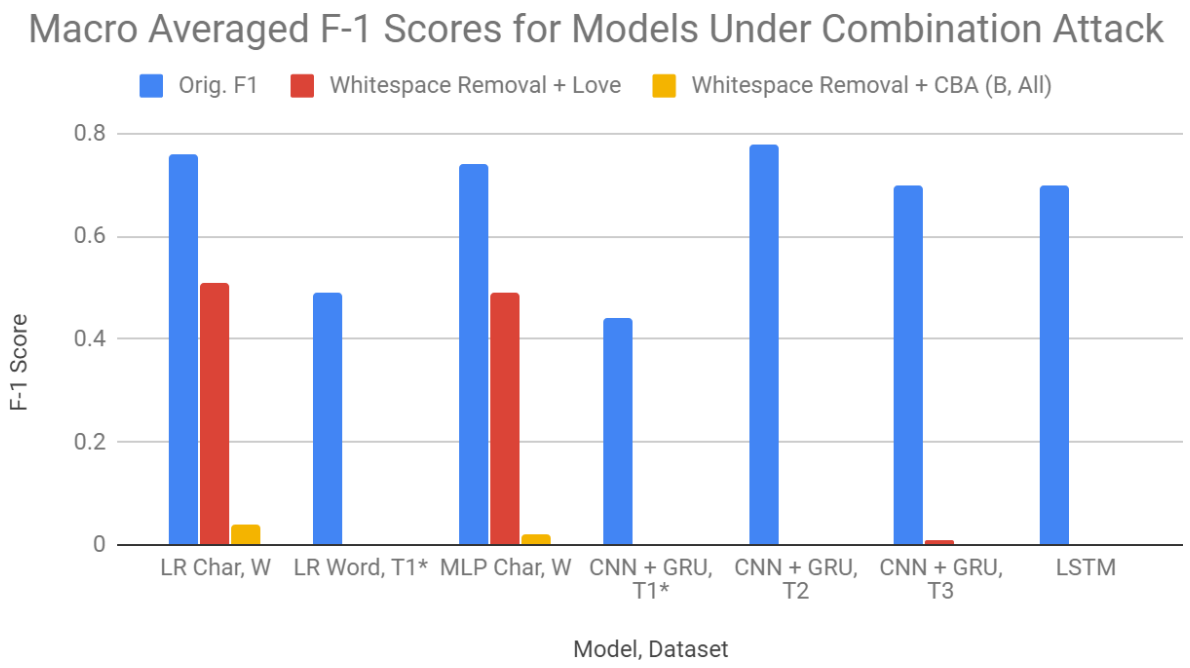

Figure 5. Graph of Macro Averaged F-1 Scores for Model, Dataset Combinations under Combination Attack

Classification of Evasion Schemes

Although we may have an effective defense for each attack, properly applying them is another problem. Some defenses such as GGNR may cause a false positive if applied to neutral text which highlights the need for proper classification before applying the defense. Thus, there

is a need to classify which evasion scheme has been applied to a text before using a defense. To our knowledge, this problem has not been addressed in any other published work.

TABLE IV. Macro Averaged Precision, Recall, F-1 Scores of Model, Feature Type Combinations For Lexical Evasion Scheme Classification

Model, Dataset	**Macro Averaged Metric**		
	Precision	***Recall***	***F-1***
Random Forest, Engineered	0.4012	0.3953	0.3982
Random Forest, Char TF-IDF	0.9827	0.9826	0.9825
AdaBoost, Engineered	0.4859	0.4038	0.4411
AdaBoost, Char TF-IDF	0.7092	0.7427	0.7014
Naive Bayes, Char TF-IDF	0.9557	0.9543	0.9543
SGD, Char TF-IDF	0.9799	0.9797	0.9797
SVM, Char TF-IDF	0.9847	0.9846	0.9846
KNN, Char TF-IDF	0.8705	0.8615	0.8637
Multilayer Perceptron, Char TF-IDF	0.9825	0.9824	0.9824
Logistic Regression, Engineered	0.4206	0.4239	0.4222
Logistic Regression, Engineered RFE	0.4261	0.4352	0.4306
Logistic Regression, Char TF-IDF	0.9845	0.9844	0.9844
LGBM, Char TF-IDF	0.9893	0.9892	0.9891

Table 4. displays our results from testing every model. We applied 5-fold cross validation when testing and used one of three sets of features for models to train upon. The three feature

sets consist of Char tf-idf, Engineered, and Engineered RFE. Engineered signifies all six features were used while Engineered RFE denotes that only character count, average word length, and number of spaces were used as features.

Overall, our engineered features performed poorly with most achieving a F-1 score of 0.42. This may be due to a lack of ability for our features to capture the statistical makeup of the text itself. Meanwhile, the models that used Char tf-idf as a feature yielded extremely strong results with logistic regression managing to reach a 0.98 F-1 score. We attribute this high performance to the ability of Char tf-idf to capture the important character n-grams that are frequently modified in significant ways when an evasion scheme is applied.

Our best performing models consist of Random Forest, Support Vector Machine, and Light Gradient Boosting Machines (LGBM), a new variant of gradient boosted decision trees. These models achieved strong results which we consider to be a result of these models' ability to leverage changes in statistical information and makeup when comparing different attacks.

VII. Conclusion and Future Work

Our results strongly suggest that evasion schemes employed on lexical data can be reliably defeated and effects minimized. Of each attack that we addressed, none were resilient enough to break a generically crafted defense. In crafting each defense, we leveraged the important requirements of any evasion scheme, the need for preserving meaning and readability, to defeat them. We strongly push for any future defenses to factor in this property of evasion schemes.

Furthermore, we demonstrated the viability of using pre-processing defenses to reverse the effects of attacks. While training defenses were also effective at mitigating attacks, they only

outperformed their pre-processing counter in one instance in an insubstantial manner. Pre-processing defenses require minimal computational power and time in comparison to training. However, training is simpler to implement since it only involves training on pre-attacked data while pre-processing defenses required a level of ingenuity to create.

The hardest attacks to defend against were those that affected the tokenization of words. Both the CBA and the whitespace removal attacks fall into this category; they yielded some of the strongest results. This suggests that tokenization is an important factor in the proper operation of hate speech detection models regardless of whether the model is word or character-based.

Our newly proposed attack, CBA, outperformed every attack created by [6]. We attribute this to how attack can affect properties that both character-based and word-based models utilize, statistical makeup and tokenization. Due to the importance of both these properties, we suggest that future defenses aim to preserve these two properties .

We suggest several directions for future work. One direction involves the classification of evasion schemes. In both this work and in [6], the issue of classifying attacked data was never addressed. Although we may have an effective defense for each attack, properly applying them is another problem. Some defenses such as GGNR may cause a false positive if applied to neutral text which highlights the need for proper classification before applying the defense. In other cases, there will be a waste of computational resources if defenses are applied on benign data.

Another direction for future work is the development of more robust or specialized software for the defenses that required apis. In the case of GGNR and WSNR, both relied on external apis that would solve a subproblem for them. Through our experiments, we found that we replacing the APIs with either more specialized functions or robustness in general yielded

significant performance boosts to our defenses. The detection of words violating grammar rules and word segmentation are significant problems in themselves and thus warrant further research.

References

[1] Kumar, S., and Shah, N. False information on web and social media: A survey. CoRR abs/1804.08559 (2018)

[2] A. H. Razavi, D. Inkpen, S. Uritsky, and S. Matwin, ‘‘Offensive language detection using multi-level classification,’’ Advances in Artificial Intelligence, vol. 6085. Ottawa, ON, Canada: Springer, Jun. 2010,
pp. 16–27.

[3] W. Warner and J. Hirschberg, ‘‘Detecting hate speech on the world wide Web,’’ in Proc. 2nd Workshop Lang. Social Media, Jun. 2012, pp. 19–26.

[4] Watanabe, H., Bouazizi, M., & Ohtsuki, T. (2018). Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection. *IEEE Access,6,* 13825-13835. doi:10.1109/access.2018.2806394

[5] Ruwandika, N., & Weerasinghe, A. (2018). Identification of Hate Speech in Social Media. *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*. doi:10.1109/icter.2018.861551

[6] Grondahl, Tommi, Pajola, Luca, Juuti, Mika, Conti, Mauro & Asokan, N (2018). All You Need is" Love": Evading Hate-speech Detection. *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security.* 2-12. doi:10.1145/3270101.3270103

[7] Wulczyn, E., Thain, N., & Dixon, L. (2017). Ex Machina: Personal Attacks Seen at Scale. Proceedings of the 26th International Conference on World Wide Web - WWW 17. doi:10.1145/3038912.3052591

[8] Davidson, T., Warmslay, D., Macy, M., and Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. In Proceedings of the 11th Conference on Web and Social Media. 512-515.

[9] Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep Learning for Hate Speech Detection in Tweets. Proceedings of the 26th International Conference on World Wide Web Companion - WWW 17 Companion. doi:10.1145/3041021.3054223

[10] Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. The Semantic Web Lecture Notes in Computer Science,745-760. doi:10.1007/978-3-319-93417-4_48

[11] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. Abusive language detection in online user content. In WWW, 2016.

[12] Gao, L., & Huang, R. (2017). Detecting Online Hate Speech Using Context Aware Models. RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning. doi:10.26615/978-954-452-049-6_036