Automating Online Hate Speech Detection: A Survey of Deep Learning Approaches

Ashe Magalhaes

Master of Science

School of Informatics
University of Edinburgh

2019

Abstract

The problem of automating the detection of online hate speech is made more critical by the moral and legal consequences of its propagation in a social media network. This research surveys the current landscape of online hate speech detection, deep learning model architectures, embedding choices, and the inclusion of user behavior metrics, in order to improve performance on detecting the hateful class in a Twitter dataset. We find that Google's pretrained BERT embeddings outperform the inclusion of user behavior metrics across many model choices and that a single-layer multiple input Convolutional Neural Network with many filters performs best. The contribution of this paper is a range of experiments that inform machine learning researchers on promising future paths and policymakers on the capabilities and limitations of automated systems on the task of detecting hate speech.

Keywords: *Hate Speech Detection, Deep Learning, Natural Language Processing, Twitter*

Acknowledgements

I would like to thank my advisor, Walid Magdy, for his guidance, flexibility, and feedback. It was consistently fun to brainstorm with Walid and this brought a valuable level of excitement to the lengthy process of writing this dissertation. Thank you to Antreas Antoniou, who kept me engaged with this topic and challenged me to grow into a more robust and creative machine learning engineer. Thank you to Abeer AL-Dayel, who offered valuable insight around model performances. Finally, I would like to thank my family, mentors, and friends for their support and encouragement–you are all instrumental in my ability to work towards lofty goals.

Statement of Originality

I confirm that I have read and understood the University's plagiarism guidelines. All work submitted is my own unless referenced, in both the text and the bibliography.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

In recent years social media companies like Facebook, Twitter, and Alphabet Inc.'s YouTube have been widely criticized for failing to detect and remove hate speech from their platforms [1]. Terrorist groups like al-Qaeda, ISIS, the Taliban, and Boko Haram have repeatedly been unhindered in their releasing of hateful content, used for the purposes of recruiting and radicalizing followers [2]. In March 2019, a white supremacist posted racist manifestos–which spread through Twitter–and live-streamed the shooting of over 51 people on Facebook and YouTube. While both companies leverage machine learning algorithms to automatically detect and remove such content, they failed to take down the content quickly enough. Facebook's head of public policy defended the platform's slow response time:

> "The video was a new type that our machine learning system hadn't seen before. It was a first person shooter with a GoPro on his head...This is unfortunately an adversarial space. Those sharing the video were deliberately splicing and cutting it and using filters to subvert automation. There is still progress to be made with machine learning [3]."

Progress needs to be made sooner rather than later. Hateful content on social media contributes to real-world violence [4], recruitment to and propaganda for terrorist individuals/groups [5], makes other users feel less safe and secure on social platforms [6], and triggers increased levels of toxicity in the network [7].

Machine learning is a powerful tool for natural language processing (NLP) tasks such as hate speech detection. Statistical learning methods learn weights for human-designed representations of feature data. In contrast, deep learning, a subfield of ML,

shifts the burden of time-intensive feature design from linguistic experts to the learning system. Supervised deep learning methods have achieved high levels of performance in NLP tasks such as sentiment analysis, language generation, and information retrieval [8]. For the hate speech detection task, we have a corpus of posts, tweets, or videos labeled as hate speech or not, with automated systems learning the nuanced interpretation required to quickly detect hateful content at scale.

Governments are enforcing the need to find reasonable solutions, fast. The Sri Lanken government temporarily banned social media networks three separate times in the wake of the Easter 2019 suicide bombings that killed hundreds of people, in order to prevent "social unrest via hate messages" [9]. Germany and the UK have strict legislation against hate speech [10]. French leader Emmanuel Macron has made fighting online hate speech a priority, taking meetings with Mark Zuckerberg and other high-level Facebook representatives; as a result Facebook has agreed to hand over data on French users suspected of spewing hate speech online. This is an international first, and according to a counsel at law firm Linklaters, "a strong signal in terms of regulation" [11]. EU countries have threatened to fine social networks up to 50 million euro per year if they continue to fail to act fast enough [12].

This paper aims to navigate the challenge of creating systems that distinguish hateful content from abusive, normal, and spam content when there are very few hateful samples to learn from. There are 500 million tweets produced daily [13], and the vast majority of those tweets are not considered hate speech [14]. We echo the concerns of Facebook's head of public policy by asking:

How can we improve the performance of automated systems on identifying hate speech when they must learn from very few hateful samples?

Of particular relevance to this paper is modeling user behavior on Twitter. User timelines and user interactions provide information on each user's surrounding community and offer a more detailed picture of identifying 'hateful users', or those who are prone to disseminating hateful content. Hateful users have been shown to tweet more often and favorite more tweets than 'normal' users [15]. The median hateful user tends to have higher network centrality than normal users according to metrics like betweeness and out-degree. Augmenting inputs to deep learning classifiers with metrics of user behavior could help moderation teams quickly identify hateful content through profiles.

Social media has contributed to a more open and connected world [16]. It has promoted Western liberal values through its effects on protest mobilization [17], commu-

nity building [18], and accountability in governments and institutions [19]. It is critical that we do not lose the benefits of these platforms as they operate under increased regulation and scrutiny. So, while automated systems can and should enforce uniform policies for user behavior that fosters welcoming online environments, this should not come at the cost of dissenting and fringe views. This research surveys the capabilities and limitations of state-of-the-art (SOTA) deep learning classifiers. We aim to inform policy and decision makers, who must reconcile the benefits of social media platforms with the harms they threaten when hate speech is allowed to propagate.

1.2 Outline

We begin by exploring the history of online hate speech as well as the challenges of collecting and annotating a hate speech dataset from social media platforms. We then describe related research and the model architectures which have shaped the current landscape of automated hate speech detection. We explain how this research informs our methodology and go on to describe our data, how it is collected and processed, the details of our model architectures, and metrics for evaluation. **We hypothesize that the inclusion of user behavior features in feature embeddings to deep learning model architectures will improve performance of the hate speech detection task.** Finally, we interpret the results from both a technical and policy perspective in order to a) enumerate and qualify the contributions of our experiments in relation to the existing technical body of related research and b) advise policymakers on how such results may be acted on. We find that pairing sophisticated pretrained embeddings with user behavior metrics in a multiple input CNN architecture achieves competitive performance given the scarcity of hateful samples, but does not perform at a level acceptable for complete automation of the task.

While this paper describes technical experiments, its intention is to offer an accessible discussion on this topic to readers from all disciplines. A glossary is offered in Appendix B to define technical jargon. The hope is that computer scientists, linguists, policymakers, lawyers, and others can form interdisciplinary coalitions that are better equipped to tackle the complex and important task of hate speech detection.

Results from this research are compiled in a workshop paper, targeted at Fall 2019 conference submissions. See Appendix C for the full draft.

Chapter 2

Background

2.1 History of Automating Hate Speech Detection

Early work on online hate speech detection focused on the necessity of preserving free speech despite hate speech [20, 21]. The question of whether online forms of expression translate into negative offline actions has been hotly contested. Stormfront, considered the first "hate website", was launched in 1995 by a Ku Klux Klan leader as a forum for discussing ideas related to white separatism and white nationalism. Members of the site claimed to be covered under their American First Amendment rights for expression, arguing that they did not act on the forum's sentiment. A subsequent study conducted in 2014 found users of the site indirectly responsible for "the murders of nearly 100 people in the preceding five years" [22].

In 2014 the UN Human Rights Council observed the proliferation of hate speech online with a focus on ethnicity, nationality, and religion [23]. Subsequent research demonstrated that online hate speech incites real-world violence against refugees in Germany [4] and that in online environments toxicity from one source is likely to foment both reciprocal and sympathetic toxicity from adjacent users [7]. Public criticisms against Facebook, Twitter, and YouTube's slow responses to combating the spread of hate speech have impelled them to hire thousands of people to work on the problem [24]. In the meantime, strict online policies against offensive language or sentiment that falls short of hate speech has resulted in the rapid rise of new platforms that cater to fringe online communities. The social network Gab was created as an alternative to Twitter, welcoming users banned or suspended from other platforms. Researchers have found that its posts contain hate words at a rate 2.4 times higher than Twitter [25].

In spite of the growing phenomenon of hate speech, prior to 2016, NLP research on hate speech was limited [6, 26]. Since then, statistical machine learning techniques like logistic regression and naive Bayes classifiers have been applied to this area with success; logistic regression in particular is widely considered a baseline model for experiments today [26, 27, 28]. Recently, the aforementioned pressures, coupled with concurrent advances in NLP and deep learning, has sparked a resurgence in interest in the field. The success of deep neural networks on NLP tasks like sentiment analysis [29], named entity recognition [30], and part-of-speech tagging [31] have made it a natural choice to apply to the task of classifying tweets or posts as hateful.

2.2 Hate Speech Datasets

It is a nontrivial task to create datasets for training machine learning classifiers on hate speech detection. The common practice is to deploy a collection of tweets or posts to a crowdsourcing site, where a number of human annotators identify an appropriate classification (i.e. hateful, normal, abusive, spam). This is problematic for a few reasons and highlights the ways in which human-annotated data reflects social biases. First, the subjective nature of identifying hate speech makes it difficult to extract ground truth labels for tweets or posts [14]. There are many definitions of hate speech, which vary across different languages, cultures, and governments. Many of the definitions involve the hateful targeting of members of a group, like this one:

> "Language used to express hatred towards a targeted individual or group, or is intended to be derogatory, to humiliate, or to insult the members of the group, on the basis of attributes such as race, religion, ethnic origin, sexual orientation, disability, or gender." [32]

Similarly, the UN's definition is:

> "The term hate speech is understood as any kind of communication in speech, writing or behaviour, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, colour, descent, gender or other identity factor. This is often rooted in, and generates intolerance and hatred and, in certain contexts, can be demeaning and divisive." [33]

Second, because there is no universally agreed upon definition of hate speech [18], it is often difficult for human annotators to separate hate speech from offensive language. Given the legal consequences of hate speech, it is important that annotators and

machine learning classifiers distinguish between hateful language and offensive language [14, 32]. It has been found that showing users a definition of hate speech does not improve annotation reliability, hinting that the presence of hate speech is not a binary decision and annotators likely need more context [28], such as sample post-label pairings.

Third, it is likely that annotators skim through tweets or posts too quickly and fail to pick up on the context of the language. Past research demonstrates that annotators will incorrectly label normal text as hate speech if it contains hate or curse words [32]. Annotators are likely to miss out on hate speech without a slur as well as interpret racist and homophobic slurs as hateful but sexist slurs as 'just offensive' [26, 32].

Creating robust hate speech datasets remains an active area of research with persistent challenges. While this research focuses on the implementation of deep learning model architectures, it is clear that progress made in reconciling the social biases in the collection of labeled hate speech data will reduce algorithmic bias.

2.3 State of the Art Approaches

The application of deep neural networks to the task of hate speech detection is fairly recent. While we define SOTA approaches as those specifically cited in the hate speech detection literature, other advances in NLP (i.e. Open AI's recent transformer language model [34]) offer novel contributions to the task. We categorize implementations based on how data is processed, model architecture choices, and the feature embedding types used to convert text into inputs suitable for machine learning. Of particular relevance to this work is research that identifies hateful or abusive language on Twitter.

2.3.1 Data Processing

Hate speech detection research focuses less on data "cleaning" (i.e. removing stop words, punctuation, capitalization, etc. from tweets or posts) than other natural language tasks, likely because these language features capture more information than in more formal text. For example, some users respond to platform restrictions on speech by creating posts that replace letters with symbols (e.g. a$$) in order to convey their otherwise 'illegal' message undetected [35]. A standard procedure is for researchers to lowercase and tokenize tweets, as well as remove URLs. Researchers may also remove emojis, although keeping emojis will likely boost NLP-task performances [36].

Another form of data processing is bootstrapping; this method identifies hateful user accounts from annotated tweets and obtains all tweets from their timeline, labeling them hateful as well, in order to increase the number of samples and better balance the dataset in cases where its imbalanced [6].

2.3.2 Embedding Choices

A tweet or post must be made into a feature embedding in order to be fed to our model as a vector of real numbers. An embedding is a representation of model inputs that captures some of the semantics of the input in a relatively low-dimensional space. Pooling layers or trimming and padding of tweets [37] can be used to convert tweets into a fixed length for standardized inputs to our neural networks. N-grams, at both the word-level and character-level, are widely used for converting a sequence of text into a sequence of embeddings [38, 39]. Character n-grams have been used for natural language processing tasks such as authorship identification [40], native language identification [41] and machine translation [42]. Word-level and character-level embeddings have been concatenated and made into feature embedding with success, as demonstrated by [38]. Word2Vec and GloVe (see appendix) embeddings are staples of NLP research and can be compared to randomly initialized embeddings as a baseline. While GloVe is said to better account for global contextual information, in practice researchers have had success with both, as they each vary in efficacy on different datasets. Word-based frequency vectorization methods, such as term frequency-inverse document frequency (TF-IDF), are also commonly used embedding choices and offer the advantage of being language agnostic [37]. Pretrained embeddings, in contrast, are dependent on the language of the corpus they are trained on.

2.3.3 Model Architectures

The baseline machine learning model choice, which tends to perform well, is logistic regression [32, 43]. Another baseline model, support vector machines (SVMs)–with linear kernels–perform similarly to logistic regression in practice [44]. Linear models tend not to capture the sparse features in document classification tasks, undermining the detection of relations between words or symbols in sentences. CNNs and RNNs (particularly LSTMs) have achieved SOTA performance due to their ability to capture long-term dependencies in a sentence, tweet, or post through their ability to maintain information in memory for a period of time [27, 37].

2.3.4 Evaluation Metrics

Because of the scarcity of hateful content relative to benign content [14], it is critical we evaluate models based on their classification of actual hateful or abusive language. Otherwise, a model could achieve a high classification accuracy by labeling everything as non-hateful, repeatedly failing our task. F-score, precision, and recall are commonly used as evaluation metrics, because they better capture the class contributions in an imbalanced dataset. Precision is the ratio of the number of tweets correctly classified to a given class divided by the total number of tweets classified to that class; recall is the ratio of tweets correctly classified to a given class divided by the number of tweets belonging to that class. The f-score is the harmonic mean of both. We express them below with true positive (TP) and false positive (FP) rate:

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.3}$$

In our task, precision answers: of all the tweets labeled as hateful, how many were actually hateful? Recall answers: of all the hateful tweets, how many did we label? F-score is the weighted average of both precision and recall, providing a measure of if a system can achieve high precision and recall.

Due to the difficulty in acquiring hate speech datasets and relatively small dataset sizes, 10-fold cross validation is a widely used metric of system performance [27, 37]. This is because it allows the model to learn from more observations, as all observations are used for training and validation. Another interesting method is ensemble classification, in which multiple classifiers predict the label of an input and the label with majority consensus is chosen. If all classifiers disagree, the classifier with the strongest confidence in a prediction is chosen [37].

2.4 Metrics of User Behavior

Hateful users tend to make themselves central to the Twitter social network: they target more popular users and they have more statuses and followees per day, although

they have younger accounts [6, 15]. While this may increase the user's visibility, the inclusion of hateful or antagonistic content in a tweet reduces the rate of retweet by a factor of 45 [45]. Despite this, hateful users are central to Twitter reply networks as determined by features such as betweenness and eigenvector centrality [15]. Furthermore, the modeling of reply trees–rooted at a tweet and joined by replies–reveals that the in-degree of the tweet, or the number of retweets, plays an important role in the overall shape of the reply tree and a component of the Twitter reply network as a whole [46].

Latent Dirichlet Allocation (LDA) [47] is a method of topic modeling that extracts the main topics in text. It has been used to improve the performance of linear machine learning classifiers on the task of identifying abusive social media content [48]. While LDA-derived topics have been used as annotations [49], to the best of our knowledge it has not been included in feature embeddings in order to improve the performance of deep learning classifiers on the task.

2.5 Challenges

This section highlights relevant background and previous work to our task of hate speech detection; it is also worth summarizing the challenges cited by other researchers in order to inform our methodology. First, extensive data processing may remove valuable information from social media content. Given the limited number of hate speech datasets available, it is important to consider the social biases and fallacies that are inherent to crowdsourced labels. Given the scarcity of hateful content relative to benign content, evaluation metrics that account for imbalanced classes should be used. Because of the legal and moral consequences of identifying a post or user as hateful, it is critical that our classifier distinguish between hateful and abusive language. Finally, the distinction between hateful content and hateful users is worth defining [15], as the task of identifying hateful users is easier but not equivalent to the task of identifying hateful content.

Chapter 3

Methodology

Objective. The overall goal of this research is to conduct a survey of deep learning model architectures, embedding choices, and feature inputs in order to automate the identification of hateful tweets from a dataset with few hateful samples. Unlike previous work, we experiment with user behavior metrics in multiple input model architectures to provide our deep learning classifiers with more context to identify hate speech with.

3.1 Data Collection

The dataset for this research comes from Founta et al.'s work that describes the process of large scale crowdsourcing for annotations of hateful, normal, abusive, and spam tweets [14]. Founta et al. provide status ids for others to obtain tweets through the Twitter API. After learning that a number of tweets were no longer available on Twitter, they provided 100k tweets' status ids and their associated majority labels. Upon using the Twitter API, we find that only 64% of those tweets are still available online; we assume the remainder have been taken down by Twitter or the users. The class distribution is as follows: **4% hateful, 20% abusive, 62% normal, and 14% spam**; there are **64,149 tweets total**. Half the tweets achieved annotation majority with an agreement of more than 3 out of 5 votes, which should give us a degree of confidence in the annotations. Figure 3.1 illustrates a word cloud of each class.

The average retweet count of hateful content is 15% higher than the average retweet count of normal content; the average favorite count of hateful content is 76% higher than the average favorite count of normal content. Finally, on average hateful content is a reply to existing tweets 31% more often than normal content is.

Figure 3.1: Word clouds of tweets from the hateful, abusive, normal, and spam classes.

For each tweet, if it is in reply to a tweet we use Twitter's API to collect the context tweet. 19% of the total tweets are a reply. Of those, 14% are no longer available on Twitter. For each tweet, we crawl the user timeline of the author and collect their 200 most recent tweets, giving us around 12.8 million additional tweets for LDA topic modeling.

3.2 Data Processing

We clean the tweets by tokenizing, lowercasing, and removing punctuation. We keep URLs because they provide important context; tweets with URLs are more likely to get retweeted [50]. We experiment with three types of text embeddings:

1. **TF-IDF embeddings.** TF-IDF fits the training set into a weighted vector by normalized frequency of the 10,000 most common words in our vocabulary. Our validation and test set are transformed using the learned weights.

2. **Pretrained Twitter embeddings.** These embeddings are from a Word2Vec model trained on 400 million raw English tweets, with an embedding dimension of 400 [51].

3. **Pretrained BERT embeddings.** Google's BERT, or Bidirectional Encoder Representations from Transforms, is a novel method of pre-training language representations which obtains SOTA results on a range of NLP tasks described in [52].

	Instigator Tweet	Annotated (Reply) Tweet
Hateful	Want some cheese with that whine?	wow nice retort, putin lover. anything minorities say = whine. kill all f*gs, right?
Abusive	Bombshell report over the weekend shows a 'very high up' Obama official unmasking Trump associates for political pu `https://t.co/UwNWfa27ea`	You bunch of ass clowns are as bad as the illegitimate Pres we got stuck with. Leaders in bs news. #worstpresever
Normal	watching the Rays game in NY is frustrating AF. I have 2 listen 2 the announcers fave the Yankees & it goes against everything I believe in	Whoa. Chill. I may have to put this friendship on hold if you bad mouth my Yankees
Spam	This great video from @ will make you want to hit a bike trail. `https://t.co/VvK0KAJh4T`	Thanks for sharing @! Anyone interested in a MTB road trip to NI, plan your trip at `https://t.co/mKBUPSDWZc` #RideNorth

Table 3.1: Sample tweet reply pairings for hateful, abusive, normal, and spam classes.

Specifically, we use the BERT-12-768-12 model with an embedding dimension of 768.

For our word embeddings, we trim or pad each tweet to 17 words, the average size of a tweet in our dataset. When padding, we add randomly generated embeddings so that each tweet embedding is a n-gram, where n=17. To account for the imbalance of classes in our dataset, for our training experiments we draw samples through a multinomial distribution that accounts for each class weight. Our balanced batches allow the classifier to learn class representations more consistently each epoch. We do not do this for validation and testing because a) we run the risk of tweet samples that are seen multiple times or never in our evaluation and b) it is not a realistic representation of tweets in the wild. Table 3.1 displays a random sample of each of our categories of tweets, with personally identifiable information removed.

3.3 Environment Configurations

Each experiment type is run with three or more different seeds, and the mean and standard deviation of results are reported. This adds robustness to our experiments and higher confidence in our results. In order to ensure reproducibility, we set the random number values and random number generators of our data providers, PyTorch (deep learning platform) backends, and NumPy (scientific computing package) to our experiment seed. This is so that computations are deterministic and replicable.

All experiments use the cross entropy loss function which takes in the classifier's prediction score for each class, applies a softmax layer, and measures its divergence from the true label. For our deep learning experiments, we use the Adam optimizer, an adaptive learning rate optimization algorithm designed for deep neural networks; for our logistic regression experiment we use limited-memory BFGS, a popular optimizer for linear machine learning models. Finally, we use a scheduled learning rate that relies on a cosine annealing schedule to more quickly reach a learning rate that reduces noise in our parameter updates [53].

3.4 Model Architectures

Logistic Regression Our baseline model is multi-class logistic regression which uses the L2 norm penalty as a form of regularization. We do not expect it to perform well on the hateful and abusive classes because this model assumes the data is linearly separable.

Multilayer Perceptron. Known as a "vanilla neural network", this model consists of fully connected layers with the non-linear activation function leaky ReLu applied. We use Leaky ReLus here and in other models because it allows a small, positive gradient when the unit is not active and tends to perform better than nonleaky ReLu in practice.

Convolutional Neural Network (CNN). While this model is traditionally used for image inputs, it has performed quite well on NLP tasks and has had success when applied to hate speech detection [38]. The convolutional layer computes the output of neurons (i.e. mathematical operations) that are connected to local regions in the input. It consists of a set of learnable filters that produce an activation map used to ultimately compute the class scores. For a more thorough explanation of CNNs, see [53]. Our CNN consists of a series of convolutional layers, batch normalization, leaky ReLu,

and dropout. The output is processed through a max pooling layer and then a fully connected layer which computes class probabilities. Our CNN contains a context list that gives every layer access to all the previous layers, allowing for more connectivity and greater chance of the network learning from long-term dependencies in the tweet.

DenseNet. CNNs are more accurate and efficient if they contain shorter connections between layers close to the input and output [54]. DenseNet leverages this observation by connecting every layer to every other layer: for each layer, the feature maps of all preceding layers are used as inputs into all subsequent layers. Our DenseNet implementation has achieved SOTA results for image datasets; we experiment with its application to NLP.

LSTM. LSTMs (and RNNs more broadly) improve on traditional deep neural networks by propagating information in both directions. This loop in the architecture acts as a memory state by which the network makes adjustments in the information flow, allowing the model to selectively remember or forget. This model has performed well on NLP tasks in practice because of its ability to remember dependencies and between sentences; it is applied to the task of hate speech detection by [27] [44] [38]. Our implementation passes our input through a series of LSTM layers to a final fully connected layer.

Chapter 4

Experiment Design

The experiments consist of three phases that demonstrate the effect of user behavior metrics on a combination of models and embedding types. Each of our 4 model choices are run with our 3 embedding types, for 4 rounds of experiments, with 3 different seeds, for a total of **144+ experiments before tuning**.

4.1 Phase 1: Tweet Embeddings

We compare our deep learning models to a baseline logistic regression model to begin with an idea around how much of an improvement they can offer. Here the feature embedding to our deep learning models are simply the tweet embeddings. Table 4.1 enumerates regularization choices, number of layers, and model-specific hyperparameter values. Regularization reduces the number of parameters and amount of computation in the network, which helps to control overfitting. The number of layers and hyperparameter choices are set with values that are commonly used in the literature and with the expectation that tuning will be conducted down the line on the best performing models.

4.2 Phase 2: Tweet + Reply Embeddings

The goal of this round of experiments is to apply some type of context to our embeddings.

MODEL	REGULARIZATION	LAYERS
LR	L2-NORM PENALTY	-
MLP	MP, λ=1E-4	3 FC-LERELU-FC
CNN	MP, BN, δ=0.5, λ=1E-4	3 CONV-BN-LERELU-FC
DENSENET	AP, BBN, δ=0.5, λ=1E-4	4 DENSE-BBN-RELU-CONV-FC
LSTM	δ=0.5, λ=1E-4	3 LSTM-FC

Table 4.1: Baseline model architectures. Terms include Weight decay (λ), Max Pooling (MP), Average Pooling (AP), Dropout (δ), Batch Normalization (BN), and Bottleneck Layer, which includes BN (BBN). Model-specific hyperparameters include CNN filters=8, DENSENET growth-rate=12, LSTM hidden-layers=5.

4.2.1 Reply Pairings

First, because of the statistics collected around the behavior of hateful users and retweeting, we define pairs of tweets: the original tweet and a context tweet. The context tweet is for the case that the tweet was a response to something else. If the tweet is not a response to something else, we add null embeddings to indicate that there is no context for the tweet. Although only 19% of our tweets in the dataset are replies, the extra information around context (when it is there and not there) may help our network learn in some way. See Table 3.1 for an example of reply pairings.

4.2.2 Reply Network Metrics as Embedding Coefficients

Next, because tweet in-degree has been shown to be significant ([46], see Background) we focus on in-degree in terms of number of times someone has retweeted a given tweet and number of times someone has favorited a given tweet. Our maximum **retweet value is 154,565** and our **maximum favorite value is 27,741**. Because these numbers are so large, we log each (and use 0 when the count is 0), so as to not skew the learned weights. We concatenate the logged retweet and favorite counts to our tweet and reply embeddings. We are interested to see if the network can better learn from the retweet and favorite numbers as a measure of context.

4.2.3 Multiple Input Model Architecture

Here and for the remaining experiments, we shift to building a neural network with multiple inputs in order to have the network learn from the annotated tweet in addition to other types of embeddings. We have the annotated tweet embedding and the context tweet embedding as separate inputs; they are processed by different parts of the model architecture. The learned features for each are concatenated and fed into a final fully connected layer. See Figure 5.2 for a visualization of a multiple input network architecture.

4.3 Phase 3: Tweet + User Topic Embeddings

This phase aims to add context in a more sophisticated way.

4.3.1 LDA Topic Modeling Embeddings

For each tweet in our dataset, we crawl the author's user timeline and collect 200 tweets. We then conduct topic modeling through the LDA approach (discussed in Background). We use LDA, as opposed to other topic modeling techniques, because LDA represents documents (or tweets) as random mixtures over topics in the corpus, which reflects what we expect from tweets on a user's timeline [47]. For each of our users, we find the dominant topic in their timeline tweets and create embeddings of a sequence of the top 10 words related to the topic. We choose the same embedding that is used to embed our tweets for a given experiment. We also concatenate the coherence and perplexity scores of the user's timeline to each embedded topic word to add a global measure of topic modeling. Perplexity is a measure of how well a probability model can predict a topic (lower is better) and coherence captures how well topics can be defined (higher is better).

Our multiple input model architecture processes the topic embeddings and tweet embeddings. Again, we concatenate the learned features and use this as input to the final fully connected layer.

4.4 Hyperparameter Tuning & Evaluation

We tune our best performing models–by model type and embedding type–by experimenting with learning rate, regularization, number of layers, and the model specific

parameter. We define best performing model as the model with the greatest validation set f-score on the hateful classes. We evaluate our results by overall metrics of f-score, precision, and recall as well as f-score of the hateful and abusive classes. This is because we are concerned with our classifier's ability to correctly identify hateful and abusive samples. To achieve stability in the results, we evaluate the mean and standard deviation of 3 experiments each.

4.5 Limitations

In order to keep our research in the scope of our central questions, we choose experiments that demonstrate the effect of user behavior in feature embeddings on model performance. We acknowledge that there are alternative paths within our methodology that may result in higher performance and are worth exploring in future work. We list a few here. First, we choose to trim or pad tweets instead of processing tweets in batch sizes by tweet length or applying a pooling layer. Second, we do not experiment with character embeddings. Third, we do not add attention to our LSTM architecture despite its SOTA performance in other related NLP tasks. Instead, we depend on our experiments with feature embeddings to add a form of attention. Fourth, we do not conduct cross-validation or ensemble classification, even though it may augment training, because of time considerations. Finally, we collect only 200 tweets from user timelines instead of their full timelines due to rate-limiting restrictions. This research would benefit from collecting all tweets in the timeline in addition to crawling the timelines of all followers of all users in our dataset.

Additionally, aside from rate-limiting, we are limited by the data returned by Twitter's API. The reply tree study conducted by [46] in 2016 is no longer easily replicable, as Twitter only provides a subset of data. For example, when trying to collect replies from a tweet with 78 replies, only 15 were available through the API. According to Twitter:

> Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface.

Chapter 5

Results

5.1 Phase 1 Results

Table 5.1 illustrates Phase 1 results across our evaluation metrics. Logistic regression performs surprisingly well, which is in line with the results from [38]. The BERT embeddings perform the best across all models, with CNN-BERT being the best performing combination for both f-score on the hateful class and f-score overall. The MLP does not perform well at all. We hypothesize that this is because of the scarcity of hateful tweets in addition to the fact that our feature embeddings only consist of tweet embedddings; as a result the MLP and LSTM models do not have enough signals or cues to interpret the input. In contrast, we hypothesize that the CNN and DenseNet models perform well because, similar to how they handle images, they are interpreting patterns from an otherwise noisy input.

5.2 Phase 2 Results

Table 5.2 illustrates Phase 2 results across our evaluation metrics. While different model-embedding combinations see different performance boosts/losses from this round of experiments, we achieve our highest score with the CNN-BERT with network metrics as embedding coefficients, but only by a slight margin. We see the most benefit to TF-IDF embeddings for CNN and DenseNet, where the overall f-score is improved with both types of reply metrics. For pretrained Twitter and BERT embeddings, using reply metrics as embedding coefficients by passing the context tweets through a tweet-level version of our model did not help the final model in learning. MLP and LSTM were likely not affected much because similar to our hypothesis from our baselines,

Table 5.1: Phase 1 Experiment Results

MODEL	F_H	F_A	F	P	R
LR-TFIDF	0.13 ± 0.05	0.28 ± 0.12	0.52 ± 0.05	0.53 ± 0.07	0.57 ± 0.06
LR-TWIT	0.15 ± 0.01	0.63 ± 0.0	0.65 ± 0.01	0.65 ± 0.01	0.64 ± 0.01
LR-BERT	0.2 ± 0.0	0.74 ± 0.0	0.71 ± 0.0	0.71 ± 0.0	0.72 ± 0.0
MLP-TFIDF	0.0 ± 0.0	0.22 ± 0.16	0.05 ± 0.02	0.03 ± 0.01	0.18 ± 0.03
MLP-TWIT	0.08 ± 0.04	0.42 ± 0.01	0.32 ± 0.13	0.57 ± 0.03	0.32 ± 0.07
MLP-BERT	0.05 ± 0.04	0.24 ± 0.16	0.19 ± 0.18	0.33 ± 0.22	0.23 ± 0.16
CNN-TFIDF	0.1 ± 0.01	0.22 ± 0.09	0.38 ± 0.06	0.49 ± 0.06	0.33 ± 0.06
CNN-TWIT	0.27 ± 0.04	0.78 ± 0.02	0.68 ± 0.01	0.75 ± 0.01	0.65 ± 0.01
CNN-BERT	0.29 ± 0.03	0.75 ± 0.02	0.71 ± 0.03	0.76 ± 0.02	0.69 ± 0.03
LSTM-TFIDF	0.18 ± 0.01	0.62 ± 0.06	0.52 ± 0.03	0.67 ± 0.02	0.49 ± 0.03
LSTM-TWIT	0.22 ± 0.01	0.67 ± 0.02	0.62 ± 0.02	0.72 ± 0.01	0.58 ± 0.02
LSTM-BERT	0.25 ± 0.02	0.74 ± 0.01	0.67 ± 0.01	0.76 ± 0.0	0.64 ± 0.01
DENSE-TFIDF	0.08 ± 0.0	0.14 ± 0.1	0.27 ± 0.19	0.29 ± 0.21	0.26 ± 0.15
DENSE-TWIT	0.22 ± 0.01	0.74 ± 0.01	0.67 ± 0.01	0.7 ± 0.0	0.66 ± 0.01
DENSE-BERT	0.26 ± 0.01	0.73 ± 0.0	0.68 ± 0.01	0.72 ± 0.0	0.67 ± 0.01

F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall. The red highlight indicates best performing model and the blue highlight indicates the TF-IDF embeddings that see the most improvement with the inclusion of user behavior metrics.

Table 5.2: Phase 2: Reply Pairings (Top) & Network Metrics (Bottom) Experiment Results

MODEL	F_H	F_A	F	P	R
MLP-TFIDF	0.06 ± 0.04	0.11 ± 0.16	0.03 ± 0.03	0.02 ± 0.02	0.1 ± 0.07
MLP-TWIT	0.12 ± 0.01	0.36 ± 0.06	0.26 ± 0.14	0.44 ± 0.2	0.28 ± 0.09
MLP-BERT	0.07 ± 0.02	0.38 ± 0.01	0.4 ± 0.0	0.52 ± 0.02	0.38 ± 0.02
CNN-TFIDF	0.06 ± 0.04	0.23 ± 0.08	0.26 ± 0.14	0.38 ± 0.1	0.28 ± 0.05
CNN-TWIT	0.25 ± 0.01	0.77 ± 0.03	0.66 ± 0.03	0.75 ± 0.01	0.64 ± 0.03
CNN-BERT	0.29 ± 0.01	0.76 ± 0.02	0.7 ± 0.02	0.75 ± 0.01	0.68 ± 0.01
LSTM-TFIDF	0.19 ± 0.0	0.69 ± 0.0	0.56 ± 0.0	0.7 ± 0.0	0.53 ± 0.0
LSTM-TWIT	0.13 ± 0.0	0.41 ± 0.0	0.5 ± 0.0	0.61 ± 0.0	0.46 ± 0.0
LSTM-BERT	0.22 ± 0.0	0.7 ± 0.0	0.65 ± 0.0	0.74 ± 0.0	0.61 ± 0.0
DENSE-TFIDF	0.11 ± 0.01	0.3 ± 0.16	0.48 ± 0.07	0.5 ± 0.06	0.48 ± 0.07
DENSE-TWIT	0.09 ± 0.01	0.31 ± 0.07	0.46 ± 0.03	0.55 ± 0.02	0.43 ± 0.04
DENSE-BERT	0.19 ± 0.01	0.69 ± 0.03	0.64 ± 0.02	0.7 ± 0.0	0.62 ± 0.02
MLP-TFIDF	0.0 ± 0.0	0.22 ± 0.16	0.2 ± 0.19	0.15 ± 0.16	0.34 ± 0.19
MLP-TWIT	0.06 ± 0.02	0.37 ± 0.02	0.14 ± 0.02	0.38 ± 0.21	0.23 ± 0.01
MLP-BERT	0.05 ± 0.0	0.35 ± 0.01	0.13 ± 0.03	0.29 ± 0.23	0.21 ± 0.02
CNN-TFIDF	0.1 ± 0.01	0.25 ± 0.07	0.41 ± 0.08	0.5 ± 0.05	0.39 ± 0.12
CNN-TWIT	0.26 ± 0.03	0.77 ± 0.01	0.66 ± 0.01	0.75 ± 0.01	0.63 ± 0.01
CNN-BERT	0.3 ± 0.01	0.76 ± 0.0	0.69 ± 0.01	0.76 ± 0.0	0.67 ± 0.01
LSTM-TFIDF	0.17 ± 0.01	0.64 ± 0.04	0.52 ± 0.03	0.67 ± 0.02	0.49 ± 0.03
LSTM-TWIT	0.21 ± 0.01	0.65 ± 0.04	0.6 ± 0.03	0.71 ± 0.02	0.56 ± 0.02
LSTM-BERT	0.25 ± 0.0	0.73 ± 0.01	0.66 ± 0.01	0.75 ± 0.01	0.63 ± 0.01
DENSE-TFIDF	0.07 ± 0.02	0.29 ± 0.05	0.49 ± 0.06	0.53 ± 0.07	0.5 ± 0.1
DENSE-TWIT	0.22 ± 0.02	0.75 ± 0.01	0.67 ± 0.0	0.71 ± 0.02	0.65 ± 0.0
DENSE-BERT	0.25 ± 0.0	0.73 ± 0.01	0.68 ± 0.0	0.71 ± 0.0	0.67 ± 0.01

F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall. The red highlight indicates best performing model and the blue highlight indicates the TF-IDF embeddings that see the most improvement with the inclusion of user behavior metrics.

Table 5.3: Phase 3: User Behavior Metrics Experiment Results

MODEL	F_H	F_A	F	P	R
MLP-TFIDF	0.14 ± 0.02	0.47 ± 0.02	0.39 ± 0.04	0.58 ± 0.01	0.39 ± 0.03
MLP-TWIT	0.08 ± 0.04	0.36 ± 0.1	0.31 ± 0.16	0.41 ± 0.24	0.32 ± 0.11
MLP-BERT	0.06 ± 0.01	0.39 ± 0.02	0.35 ± 0.09	0.53 ± 0.03	0.34 ± 0.07
CNN-TFIDF	0.24 ± 0.01	0.82 ± 0.01	0.64 ± 0.02	0.77 ± 0.01	0.61 ± 0.02
CNN-TWIT	0.19 ± 0.07	0.71 ± 0.08	0.62 ± 0.06	0.71 ± 0.04	0.61 ± 0.05
CNN-BERT	0.3 ± 0.02	0.76 ± 0.01	0.68 ± 0.03	0.76 ± 0.01	0.65 ± 0.03
LSTM-TFIDF	0.2 ± 0.0	0.72 ± 0.01	0.59 ± 0.01	0.73 ± 0.0	0.55 ± 0.01
LSTM-TWIT	0.21 ± 0.01	0.66 ± 0.02	0.61 ± 0.02	0.72 ± 0.01	0.57 ± 0.02
LSTM-BERT	0.25 ± 0.01	0.73 ± 0.01	0.65 ± 0.01	0.75 ± 0.01	0.62 ± 0.01
DENSE-TFIDF	0.21 ± 0.0	0.76 ± 0.01	0.66 ± 0.01	0.72 ± 0.02	0.64 ± 0.01
DENSE-TWIT	0.19 ± 0.01	0.74 ± 0.02	0.69 ± 0.01	0.71 ± 0.0	0.67 ± 0.01
DENSE-BERT	0.24 ± 0.0	0.73 ± 0.0	0.68 ± 0.02	0.72 ± 0.0	0.66 ± 0.02

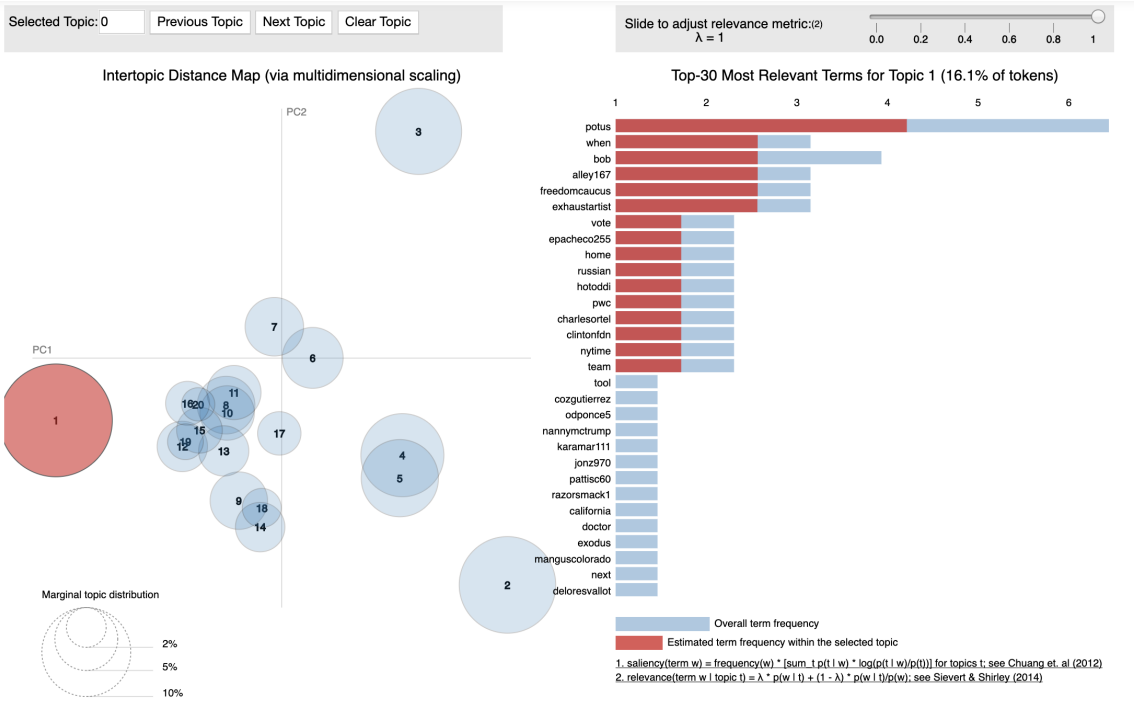
F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall. The red highlight indicates best performing model and the blue highlight indicates the TF-IDF embeddings that see the most improvement with the inclusion of user behavior metrics.

the sparse amount of hateful samples coupled with the lack of cues or signal around the content of our tweets and reply tweets inhibits the model from learning. This tells us that it is not enough to feed our models more context: we need to find a better way to define that context within our feature embeddings.

5.3 Phase 3 Results

Figure 5.1 illustrates the topic keywords from our LDA model of user timeline tweets from a user that authored a sample hateful tweet. We see that our largest topic, with **16.1% of tweet tokens**, includes terms like 'potus', 'russian', 'nytime', and 'vote', implying that this user frequently tweets about US political issues. Another tweet annotated as hateful listed the dominant topic's top words as 'film', 'good', 'award', 'location', 'series', 'international'; upon further inspection of the tweet and the ac-

Figure 5.1: Topic LDA Modeling of User's Timeline from Sample Hateful Tweet

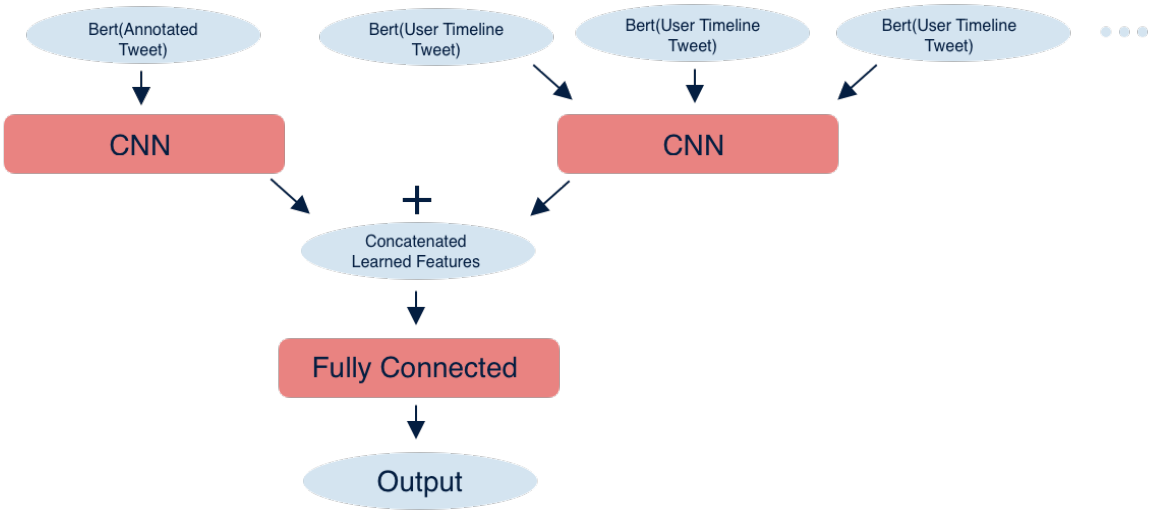


Sample hateful tweet is: "Muslims immigrants are living like they're used to where their from, squalor, filth and violence are all they know." This illustrates output from the Gensim library for LDA Topic modeling.

count, we found that the tweet was quoting a film and was mislabelled as hateful. In this case, topic modeling highlighted the difference between hateful users and hateful tweets by providing much needed context.

Table 5.3 illustrates phase 3 results across our evaluation metrics. We see that we achieve our best performing model on the hateful class, CNN-BERT, by using topic modeling features, but with only a slight improvement from our previous experiment phases. Figure 5.2 illustrates the model architecture of this best performing model. Notably, adding the user behavior metrics to the TF-IDF embedded models improves performance for the CNN and DenseNet classifiers, more than in phase 2 of our experiments. This tell us that the pretrained Twitter and BERT embeddings add a great deal of semantic value and cannot be much improved upon by user behavior metrics, such as topics from the user timeline. In contrast, TF-IDF embeddings do not offer this, and dominant topic words compensate for that.

Figure 5.2: Best Performing Model Architecture

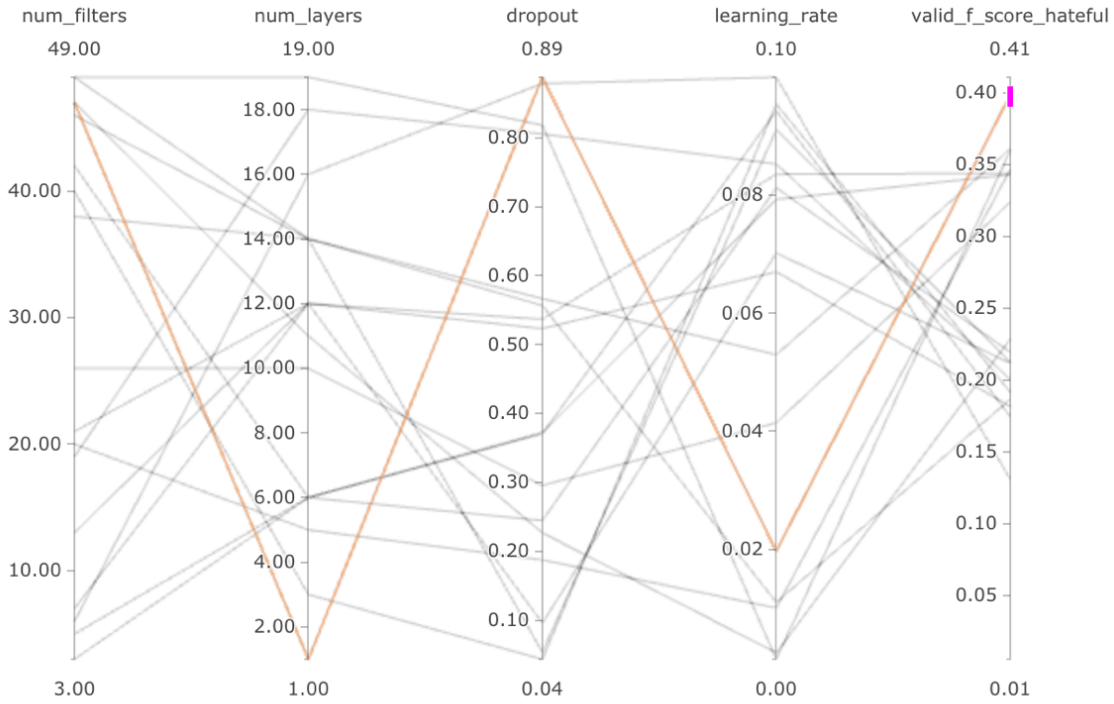


CNN-BERT with User Behavior Metrics. Illustrates a multiple input model architecture where the BERT embeddings of annotated tweets and user timeline tweets are passed through distinct CNN models. The learned features are concatenated and passed as input to the final fully connected layer, which maps to the output.

5.4 Hyperparameter Tuning

We take our best performing model and embedding combination, CNN-BERT with user behavior metrics, and tune it by experimenting with dropout, number of layers, number of filters, and learning rate. Because the CNN-BERT model converged on average at **epoch 44**, we do not experiment with number of epochs. Figure 5.3 illustrates the 17 rounds of tuning experiments, with runtimes ranging from 5 hours and 20 minutes to 4 minutes (see Appendix A for machine configurations). We use the Bayes algorithm with the adaptive Parzen-Rosenblatt estimator (facilitated by the Comet.ML library's optimization framework [55]), which balances exploring unknown space with exploiting the known hyperparameter values that yield the best results. The following values perform best of the values explored: **num-filters=47, num-layers=1, dropout=0.8877, learning-rate=0.0196**. This leaves us with a single layer CNN with many filters and with a high level of regularization.

Figure 5.3: Hyperparameter Tuning Parallel Coorodinates Chart



This chart illustrates the parameter choices and validation f-score on the hateful class across 17 experiments. Hyperparameter tuning was facilitated by the Comet.ML library's optimizer and visualization tools [55].

5.5 Tuned Model Results Across Experiments

Table 5.4 and Figure 5.5 illustrate the tuned model performances. See Appendix A.1 for additional results. All the tuned models demonstrate higher f-score on the minority class, with the tuned phase 2 reply-pairings model performing best at $\mathbf{0.33} \pm \mathbf{0.01}$. This model also performs best on all classes, with an overall f-score of $\mathbf{0.72} \pm \mathbf{0.01}$. It achieves the highest recall but not the highest precision of the experiments conducted so far. We hypothesize that the tuned phase 2 reply-pairings model performs better than the other models because the single layer CNN with many filters is overfitting with the BERT and LDA topic embeddings as well as with the reply network coefficients. With only the tweet and reply-pairing BERT embeddings, the network seems to have a more generalizeable notion of context.

In order to better understand our precision and recall scores, we analyze a confusion matrix in table 5.4. We see that the hateful class was correctly classified **40%** of the

time and was most commonly confused with the abusive class. The abusive class was correctly classified **82%** of the time, which is higher than the model's ability to detect normal tweets or spam tweets. In the below samples, the model predicts a tweet is hateful when it is labeled as hateful, abusive, and normal, respectively. Personally identifiable information is removed.

> *RHate when folk ask me questions that they already know the answers to . B*TCH what you fishing for ?* (Hateful Label)

> *That was the most patronising thing I've ever heard about young people and propaganda. What about old people and the mail?!* (Abusive Label)

> *TEACHERS too! Another "trusted" member of society, doing the nasty with teen.* (Normal Label)

There are a few problems with this. First, the tweet that is labelled as hateful does not appear to be hate speech from the definitions explored in our background. Second, the tweet that is labelled as normal could be considered abusive towards teachers. All of these tweets have negative sentiment, so it does appear our model has picked up on that. This deeper dive into the actual content that is misclassified hints at a problem with the annotation quality of our dataset.

5.6 Results Summary

The models that used Google's pretrained BERT embeddings performed better than TF-IDF and Twitter pretrained embeddings across most models in our three phases of experiments. CNN-BERT outperformed the logistic regression, MLP, LSTM, and DenseNet models for all three phases of experiments. Before tuning, our best performing model is the CNN multiple input model architecture with tweet and LDA topic words from user timeline tweets, all embedded with Google's pretrained BERT embeddings. After tuning, our best performing model is the CNN multiple input model architecture with tweet and reply pairings embedded using Google's pretrained BERT embeddings. We hypothesize that this is because the parameter choices of a single layer, 47 filter CNN with high dropout still leads to overfitting when additional context is added to the embeddings. This may be because the pretrained BERT embeddings add a lot of semantic information to begin with, which is demonstrated in our results where BERT embeddings are not much improved upon between experiment phases.

Table 5.4: Tuned CNN-BERT Model Performance

MODEL	F_H	F_A	F	P	R
PHASE-1	0.32 ± 0.02	0.77 ± 0.01	0.71 ± 0.01	0.76 ± 0.01	0.69 ± 0.01
PHASE-2-1	0.33 ± 0.01	0.78 ± 0.01	0.72 ± 0.01	0.76 ± 0.0	0.7 ± 0.01
PHASE-2-2	0.32 ± 0.0	0.77 ± 0.01	0.7 ± 0.01	0.76 ± 0.0	0.69 ± 0.01
PHASE-3	0.32 ± 0.01	0.78 ± 0.01	0.7 ± 0.02	0.77 ± 0.01	0.68 ± 0.02

F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall.

Ultimately, we are able to improve on our logistic regression baseline performance on the hateful class by **.13** on a dataset with scarce hateful labels. If we were to randomly annotate a tweet as hateful with 4% probability, we'd achieve around 4% accuracy on the hateful class. Thus, we interpret the f-score of .33 as relatively high.

5.7 Experiment Difficulties

Below are a few experiment difficulties.

Nondeterministic evaluation error in university cluster's ability to load a saved model. This bug was identified when the test metrics had a standard deviation of .3+, which implied a bug in the metric computation, in sampling, in the experiment builder or in the dataset. After verifying the performance of each of those sections, the bug was isolated to the model load functionality. When the model was evaluated within training on the test set, the metrics were normal, but when the best performing model was loaded in using PyTorch, the abnormal metrics occurred in one or two out three experiment seeds. After verifying all parts of the models state dictionary and parameter weights, we guessed it was a hardware stack error, and deployed the experiment on the Google Cloud Platform (GCP) instead of the Machine Learning Practical cluster. The experiment ran a) significantly faster and b) with normal test metrics.

BERT embeddings. In order to expedite experiments, we do not load BERT embeddings from the Python library every time, because that is expensive in terms of time and computation. Instead we collect the BERT embeddings of our tweets (including our reply and user timeline tweets) 1,000 at a time, process the embeddings with

trimming or padding, and store a dictionary mappings tweet ids to embeddings.

Twitter data that has been taken down. Only 64% of the tweets in the original dataset could be recovered. For phase 2, about 16% of the instigator tweets could not be recovered. While this signals that potentially hateful or abusive content is being taken down rapidly (within a 9 month time window), it limits our ability to conduct research. A recommendation is for future researchers to provide the social media content they collect directly instead of references to it, although the retention of user data may violate some form of General Data Protection Regulation.

Figure 5.4: Confusion Matrix of Final Model.

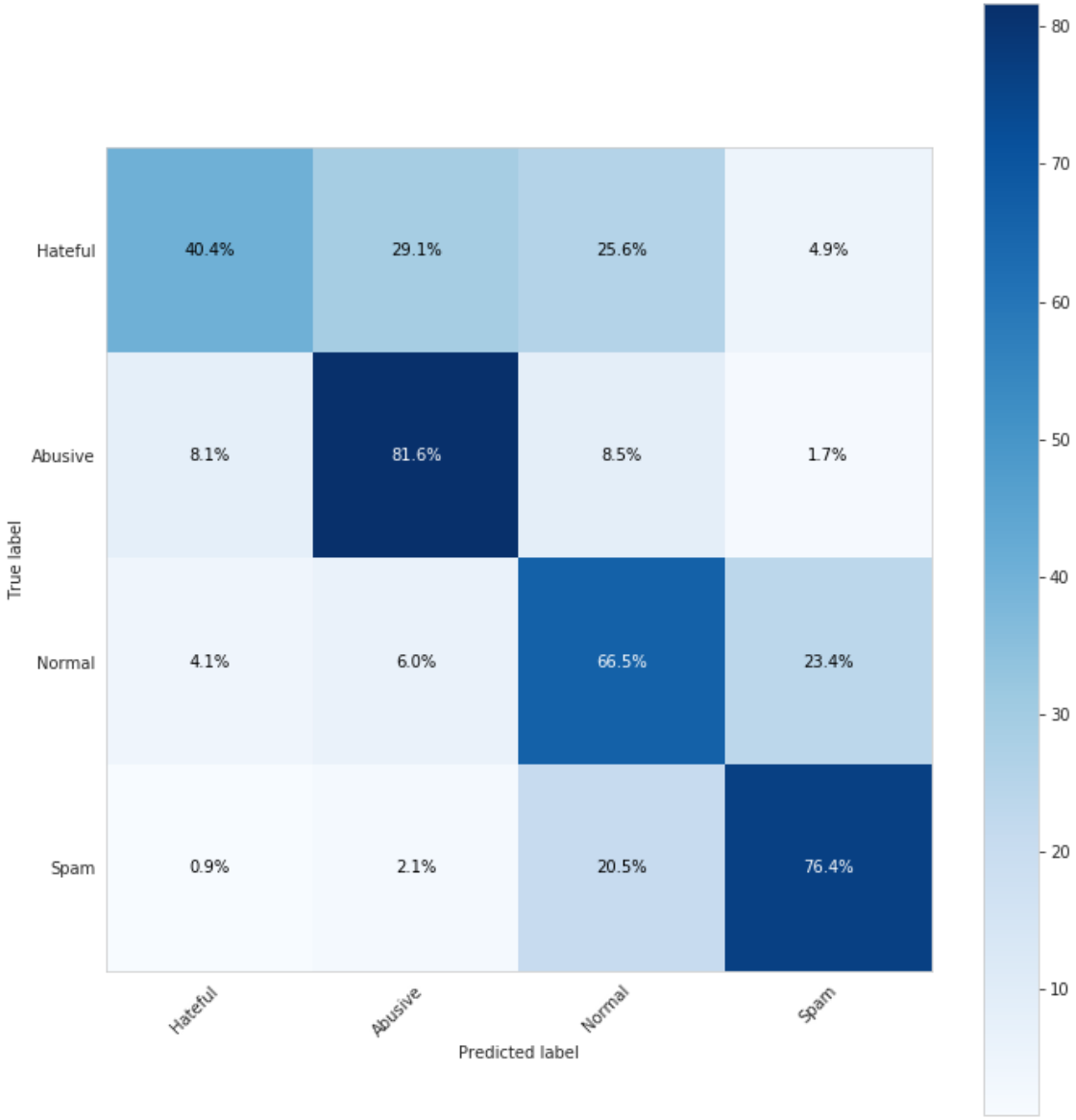
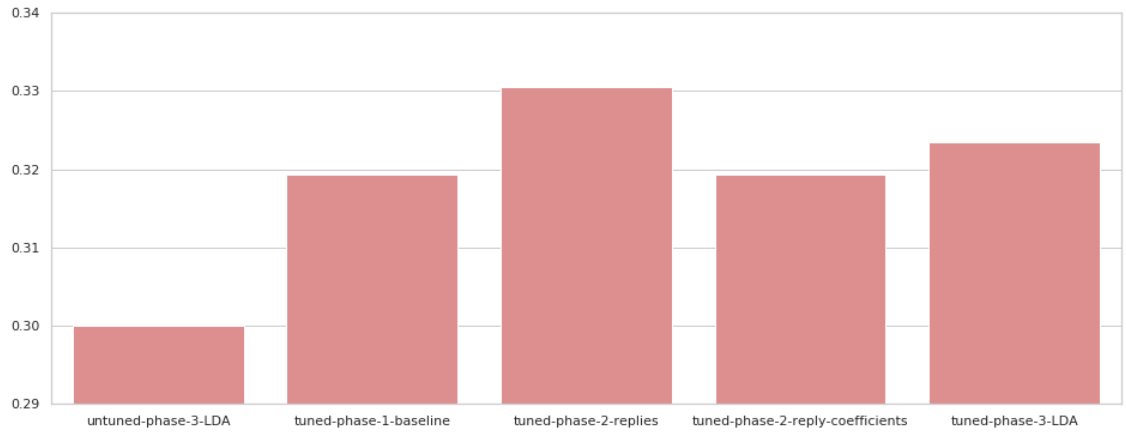


Figure 5.5: Tuned CNN-BERT Experiment Results



All the tuned models demonstrate higher f-score on the minority class, with the tuned phase 2 model performing best at 0.33.

Chapter 6

Conclusion

6.1 Technical Contributions

Our final model successfully picked up on negative tweet sentiment and identified the abusive class at the highest rate, of **82%** accuracy and **0.78** f-score. The model offers a significant improvement on detecting hate speech, as we are able to improve on our logistic regression baseline performance on the hateful class by **.13** f-score on a dataset with scarce hateful labels. If we were to randomly annotate a tweet as hateful with 4% probability, we'd achieve around 4% accuracy on the hateful class. Thus, we interpret the final f-score on the hateful class of **.33** as relatively high.

The models that used Google's pretrained BERT embeddings performed better than TF-IDF and Twitter pretrained embeddings across most models in our three phases of experiments. CNN-BERT outperformed the logistic regression, MLP, LSTM, and DenseNet models for all three phases of experiments. Before tuning, our best performing model is the CNN multiple input model architecture with tweet and user topic BERT embeddings. After tuning, our best performing model is the CNN multiple input model architecture with tweet and reply BERT embeddings. We hypothesize that this is because the parameter choices of a single layer, 47 filter CNN with high dropout will overfit with BERT and a large measure of user context. **The pretrained BERT embeddings add enough semantic information to give us our most competitive models and adding additional metrics of context through aspects of the social network does not improve performance.**

The task of automating the detection of hate speech on social media platforms remains a challenge, in part due to the difficulty in obtaining high-quality, large-scale annotated datasets and the scarce hateful samples available for machine learning mod-

els to learn from. Our experiments reflect this and suggest that improving the quality and consistency of annotations in our dataset is likely to result in more accurate automated systems. A promising area of future research is to use boosting methods, like oversampling or undersampling, in order to adjust the class distribution of our dataset.

6.2 Policy Recommendations

We now outline the key takeaways for policy and decision makers. First, if the task is completely automated it seems we risk removing posts that are abusive, normal, or spam which would look like a form of censorship on social media platforms. In this environment, the promotion of free speech may warrant keeping a human in the loop and having hateful content online for longer amounts of time. When content is published on the internet, especially controversial hate speech, it will likely be copied very quickly. Therefore, the speed at which a social media platform takes down the content does not guarantee that the content will not be made available elsewhere. It may be preferable to keep the human in the loop, especially if the detection of hate speech carries legal penalties, as is the case for certain European Union countries.

Second, topic modeling and other forms of user behavior metrics can be incorporated into a contagion tool. A contagion tool monitors trends and patterns and how they might spread through a network or change in response to triggering events. Topic modeling can provide helpful backgrounds on users and support the automated system's decision. While the inclusion of this in the feature embedding may not add much information to the model, it can add a level of interpretability to the decision by summarizing how coherent a user's timeline is around a topic and the words that make up a dominant topic. For example, we found that the author of a hate speech tweet in our dataset had a timeline defined by the topics of US politics and Russia. With this type of tool, researchers could track if hate speech from this user's follower Twitter network increases as we approach the US 2020 elections.

Third, given the massive amounts of content produced on social media platforms daily, it is not reasonable today to expect that a company can immediately detect and remove online hate speech. While increased scrutiny has furthered our capabilities, a "good" automated hate speech detector may mean detecting detecting 30% of hate speech and passing merely abusive content to human moderators.

The technical limitations described indicate that this problem will not be solved through automation alone; policy-makers must cooperate with social media companies

in order to guide the handling of classified hateful and abusive content. This research is meant to inform interdisciplinary coalitions on the current landscape of hate speech detection. Ultimately, social media has contributed to a more open and connected world. It is critical that we mitigate the negative consequences of hate speech while preserving the benefits of online discussion.

Bibliography

[1] D. Cave, "Countries want to ban 'weaponized' social media. what would that look like?" Mar 2019. [Online]. Available: https://www.nytimes.com/2019/03/31/world/australia/countries-controlling-social-media.html

[2] S. Alava, D. Frau-Meigs, and G. Hassan, *Youth and violent extremism on social media: mapping the research*. UNESCO Publishing, 2017.

[3] J. Wakefield, "Hate speech: Facebook, twitter and youtube told off by mps," Apr 2019. [Online]. Available: https://www.bbc.co.uk/news/technology-48037482

[4] K. Müller and C. Schwarz, "Fanning the flames of hate: Social media and hate crime," *Available at SSRN 3082972*, 2018.

[5] I. Awan, "Cyber-extremism: Isis and the power of social media," *Society*, vol. 54, no. 2, pp. 138–149, 2017.

[6] M. ElSherief, S. Nilizadeh, D. Nguyen, G. Vigna, and E. Belding, "Peer to peer hate: Hate speech instigators and their targets," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[7] J. Cheng, M. Bernstein, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Anyone can become a troll: Causes of trolling behavior in online discussions," in *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*. ACM, 2017, pp. 1217–1230.

[8] L. Deng and Y. Liu, *Deep Learning in Natural Language Processing*. Springer, 2018.

[9] T. Marcin, "Facebook, youtube, whatsapp banned again in sri lanka after violence against muslims," May 2019. [Online]. Available: https://news.vice.com/en_us/article/597dm5/facebook-youtube-whatsapp-banned-again-in-sri-lanka-after-violence-against-muslims

[10] E. Stein, "History against free speech: The new german law against the" auschwitz": And other:" lies"," *Michigan Law Review*, vol. 85, no. 2, pp. 277–324, 1986.

[11] M. Rosemain, "Exclusive: In a world first, facebook to give data on hate speech..." Jun 2019. [Online]. Available: https://www.reuters.com/article/us-france-tech-exclusive/exclusive-in-a-world-first-facebook-to-give-data-on-hate-speech-suspects-to-french-courts-idU

[12] E. Thomasson, "German cabinet agrees to fine social media over hate speech," Apr 2017. [Online]. Available: https://uk.reuters.com/article/uk-germany-hatecrime-facebook/german-cabinet-agrees-to-fine-social-media-over-hate-speech-idUKKBN1771FK

[13] "Twitter usage statistics." [Online]. Available: https://www.internetlivestats.com/twitter-statistics/

[14] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, "Large scale crowdsourcing and characterization of twitter abusive behavior," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[15] M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. Almeida, and W. Meira Jr, "" like sheep among wolves": Characterizing hateful users on twitter," *arXiv preprint arXiv:1801.00317*, 2017.

[16] H. Rainie, J. Q. Anderson, and J. Albright, *The future of free speech, trolls, anonymity and fake news online*. Pew Research Center Washington, DC, 2017.

[17] A. Breuer, T. Landman, and D. Farquhar, "Social media and protest mobilization: Evidence from the tunisian revolution," *Democratization*, vol. 22, no. 4, pp. 764–792, 2015.

[18] S. J. Jackson, M. Bailey, and B. Foucault Welles, "# girlslikeus: Trans advocacy and community building online," *New Media & Society*, vol. 20, no. 5, pp. 1868–1888, 2018.

[19] R. Enikolopov, M. Petrova, and K. Sonin, "Social media and corruption," *American Economic Journal: Applied Economics*, vol. 10, no. 1, pp. 150–74, 2018.

[20] R. Weintraub-Reiter, "Hate speech over the internet: A traditional constitutional analysis or a new cyber constitution," *BU Pub. Int. LJ*, vol. 8, p. 145, 1998.

[21] L. Leets and H. Giles, "Words as weaponswhen do they wound? investigations of harmful speech," *Human Communication Research*, vol. 24, no. 2, pp. 260–301, 1997.

[22] R. Cohen-Almagor, "Taking north american white supremacist groups seriously: The scope and the challenge of hate speech on the internet," *International journal of crime, justice, and social democracy*, vol. 7, no. 2, pp. 38–57, 2018.

[23] E. Keen, "Mapping study on projects against hate speech online," *Council of Europe*, 2014.

[24] I. Lunden and I. Lunden, "Facebook to add 3,000 to team reviewing posts with hate speech, crimes, and other harming posts," May 2017. [Online]. Available: https://techcrunch.com/2017/05/03/facebook-to-hire-3000-to-review-posts-with-hate-speech-crimes-and-other-harming-posts/

[25] S. Zannettou, B. Bradlyn, E. De Cristofaro, M. Sirivianos, G. Stringhini, H. Kwak, and J. Blackburn, "What is gab? a bastion of free speech or an alt-right echo chamber?" *arXiv preprint arXiv:1802.05287*, 2018.

[26] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

[27] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 759–760.

[28] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, 2017, pp. 1–10.

[29] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Coooolll: A deep learning system for twitter sentiment classification," in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 2014, pp. 208–212.

[30] M. Habibi, L. Weber, M. Neves, D. L. Wiegandt, and U. Leser, "Deep learning with word embeddings improves biomedical named entity recognition," *Bioinformatics*, vol. 33, no. 14, pp. i37–i48, 2017.

[31] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1818–1826.

[32] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Eleventh international aaai conference on web and social media*, 2017.

[33] [Online]. Available: https://www.un.org/en/genocideprevention/documents/UNStrategyandPlanofActiononHateSpeech18JuneSYNOPSIS.pdf

[34] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[35] Y. Mehdad and J. Tetreault, "Do characters abuse more than words?" in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016, pp. 299–303.

[36] I. Rahwan, "What can we learn from emojis?" [Online]. Available: https://www.media.mit.edu/posts/what-can-we-learn-from-emojis/

[37] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Detecting offensive language in tweets using deep learning," *arXiv preprint arXiv:1801.04433*, 2018.

[38] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proceedings of the first workshop on abusive language online*, 2017, pp. 85–90.

[39] S. O. Sood, E. F. Churchill, and J. Antin, "Automatic identification of personal insults on social news sites," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 2, pp. 270–285, 2012.

[40] U. Sapkota, S. Bethard, M. Montes, and T. Solorio, "Not all character n-grams are created equal: A study in authorship attribution," in *Proceedings of the 2015*

conference of the North American chapter of the association for computational linguistics: Human language technologies, 2015, pp. 93–102.

[41] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2016, pp. 145–153.

[42] P. Nakov and J. Tiedemann, "Combining word-level and character-level models for machine translation between closely-related languages," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 301–305.

[43] G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose, "Detecting offensive tweets via topical feature discovery over a large scale twitter corpus," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1980–1984.

[44] F. Del Vigna12, A. Cimino23, F. DellOrletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on facebook," 2017.

[45] P. Burnap and M. L. Williams, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy & Internet*, vol. 7, no. 2, pp. 223–242, 2015.

[46] R. Nishi, T. Takaguchi, K. Oka, T. Maehara, M. Toyoda, K.-i. Kawarabayashi, and N. Masuda, "Reply trees in twitter: data analysis and branching process models," *Social Network Analysis and Mining*, vol. 6, no. 1, p. 26, 2016.

[47] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[48] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, "Learning from bullying traces in social media," in *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*. Association for Computational Linguistics, 2012, pp. 656–666.

[49] G. Wiedemann, E. Ruppert, R. Jindal, and C. Biemann, "Transfer learning from lda to bilstm-cnn for offensive language detection in twitter," *arXiv preprint arXiv:1811.02906*, 2018.

[50] B. Suh, L. Hong, P. Pirolli, and E. H. Chi, "Want to be retweeted? large scale analytics on factors impacting retweet in twitter network," in *2010 IEEE Second International Conference on Social Computing*. IEEE, 2010, pp. 177–184.

[51] F. Godin, B. Vandersmissen, W. De Neve, and R. Van de Walle, "Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations," in *Proceedings of the workshop on noisy user-generated text*, 2015, pp. 146–153.

[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[53] A. Karpathy, "Cs231n convolutional neural networks for visual recognition." [Online]. Available: http://cs231n.github.io/neural-networks-3/

[54] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[55] Comet.ml, "Comet.ml - supercharging machine learning." [Online]. Available: https://www.comet.ml/

Appendix A

Experiments

A.1 Code

All code can be found at `https://github.com/ashemag/hate_speech_detection`

A.2 Machine Configuration

We use the Google Cloud Platform and create an instances with:

- 32 vCPUs, 208 GB memory
- 2 NVIDIA Tesla P4 GPUs
- 100 GB disk of the Deep-Learning-Pytorch Image

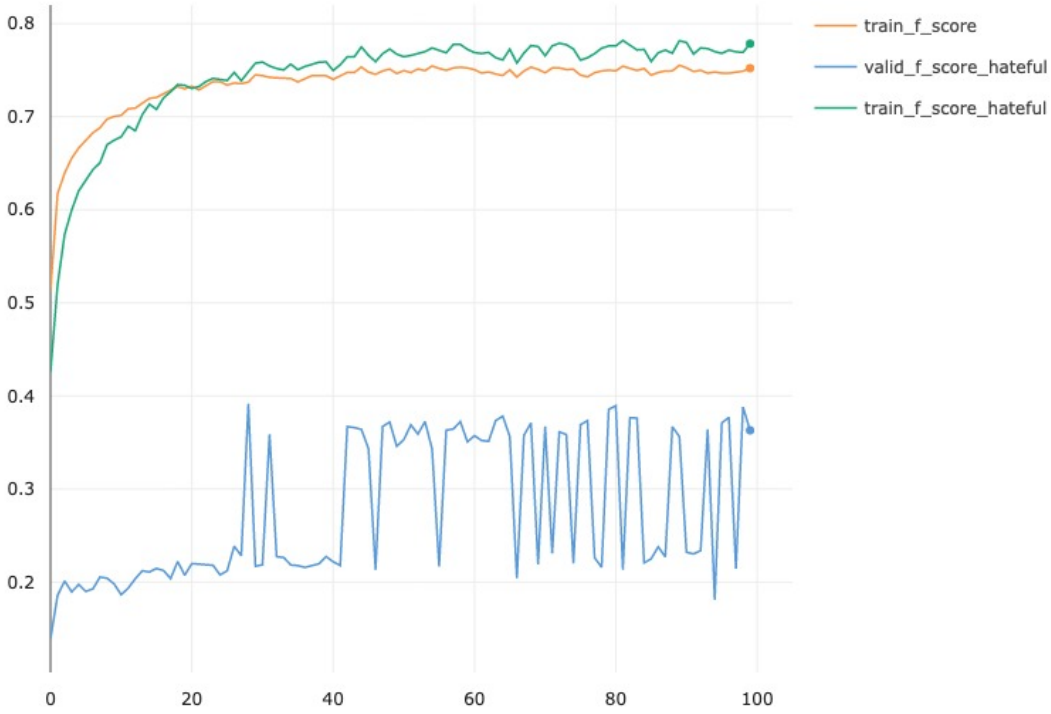
A.3 Additional Results

Table A.1: Tuned Experiment Results

MODEL	F_H	P_H	R_H	F_A	P_A	R_A	F_N	P_N	R_N
PHASE-1	0.32 ± 0.02	0.26 ± 0.02	0.42 ± 0.05	0.77 ± 0.01	0.75 ± 0.04	0.8 ± 0.03	0.76 ± 0.02	0.88 ± 0.01	0.67 ± 0.03
PHASE-2-1	0.33 ± 0.01	0.27 ± 0.02	0.42 ± 0.01	0.78 ± 0.01	0.76 ± 0.02	0.8 ± 0.01	0.77 ± 0.01	0.88 ± 0.01	0.68 ± 0.02
PHASE-2-2	0.32 ± 0.0	0.27 ± 0.02	0.41 ± 0.03	0.77 ± 0.01	0.73 ± 0.02	0.81 ± 0.0	0.75 ± 0.01	0.88 ± 0.0	0.66 ± 0.02
PHASE-3	0.32 ± 0.01	0.27 ± 0.03	0.44 ± 0.08	0.78 ± 0.01	0.77 ± 0.05	0.79 ± 0.04	0.75 ± 0.03	0.88 ± 0.01	0.65 ± 0.04

Table A.2: Test set results for classes hateful, abusive, and normal.

Figure A.1: Training vs. Validation Performances



Training f-score overall on and the hateful class is more consistent than on the validation set. This may be because our training set consists of balanced batches; even though our model is learning from the training set its unable to consistently classify the validation. This may be due to poor annotation quality.

Appendix B

Glossary

API: application program interface; allows applications to communicate with one another.

Convolutional layer: a mathematical operation that takes two inputs, such as an image and a filter, and learns features by using smaller parts of the input.

Cross validation: technique that reserves a sample of a dataset that the model is not trained on. Later, this sample is used for testing.

Crowdsourcing: the practice of obtaining information for a task by enlisting the services of a large number of people, typically via the Internet.

Feature embedding: the input fed into a machine learning model. It can consist of embeddings of different text features, like a tweet or topic. Embeddings are mappings of data into numerical vectors.

Fully connected layer: layer that connects every input neuron (i.e. operation) to outputs. We use this layer to map our learned feature representations to our final classes (i.e. hateful, abusive, normal, spam).

General Data Protection Regulation: a regulation in EU law (put into practice in 2018) on data protection and privacy for all individual citizens of the European Union (EU) and the European Economic Area (EEA).

Ground truth labels: the reality, or labels, we want our model to predict.

Hyperparameter: a parameter whose value is set before the learning process begins.

In-degree: in graph theory, this is the number of edges going into a node.

Logistic regression: a classification algorithm used to assign observations to classes. In our case, we use multiclass logistic regression.

Natural language processing (NLP): the process of a computer extracting information from natural language input and producing natural language output.

N-gram: a contiguous sequence of n items from a given sample of text or speech.

NumPy: package for scientific computing with Python.

Out-degree: in graph theory, this is the number of edges going out of a node.

Overfitting: model behavior where training corresponds too closely to data and thus may fail to fit additional data.

Personally identifiable information: data that could potentially identify an individual.

Pooling layer: combines several values into a single one. Helps the layer generalize because it effectively decreases the chance of overfitting.

PyTorch: a machine learning library based on the Torch library, for Python, used for applications such as deep learning and natural language processing.

Regularization: a way of penalizing model complexity to prevent overfitting.

Softmax layer: used to map the output of a network to a probability distribution over predicted classes. From this, we select label samples with the class with the high-

est probability.

Appendix C

Workshop Paper

The Benefits of BERT: A Survey of Deep Learning Approaches to Online Hate Speech Detection

Ashe Magalhaes
University of Edinburgh
Edinburgh, Scotland
a.m.magalhaes@ed.ac.uk

Walid Magdy
University of Edinburgh
Edinburgh, Scotland
wmagdy@inf.ed.ac.uk

Abstract

The problem of automating the detection of online hate speech is made more critical by the moral and legal consequences of its propagation in a social media network. This research surveys the current landscape of online hate speech detection, deep learning model architectures, embedding choices, and the inclusion of user behavior metrics, in order to improve performance on detecting the hateful class in a Twitter dataset. We find that Google's pretrained BERT embeddings outperform the inclusion of user behavior metrics across many model choices and that a single-layer multiple input Convolutional Neural Network with many filters performs best. The contribution of this paper is a range of experiments that inform machine learning researchers on promising future paths and policymakers on the capabilities and limitations of automated systems on the task of detecting hate speech.

Introduction

In recent years social media companies like Facebook, Twitter, and Alphabet Inc.'s YouTube have been widely criticized for failing to detect and remove hate speech from their platforms (Cave 2019). In March 2019, a white supremacist posted racist manifestos–which spread through Twitter–and live-streamed the shooting of over 51 people on Facebook and YouTube. While both companies leverage machine learning algorithms to automatically detect and remove such content, they failed to take down the content quickly enough. Facebook's head of public policy defended the platform's slow response time:

> "The video was a new type that our machine learning system hadn't seen before. It was a first person shooter with a GoPro on his head...This is unfortunately an adversarial space. Those sharing the video were deliberately splicing and cutting it and using filters to subvert automation. There is still progress to be made with machine learning (Wakefield 2019)."

Progress needs to be made sooner rather than later. Hateful content on social media contributes to real-world violence (Müller and Schwarz 2018), recruitment to and propaganda for terrorist individuals/groups (Awan 2017), makes other users feel less safe and secure on social platforms (ElSherief et al. 2018), and triggers increased levels of toxicity in the network (Cheng et al. 2017).

Governments are enforcing the need to find reasonable solutions, fast. The Sri Lanken government temporarily banned social media networks three separate times in the wake of the Easter 2019 suicide bombings that killed hundreds of people, in order to prevent "social unrest via hate messages" (Marcin 2019). French leader Emmanuel Macron has made fighting online hate speech a priority, taking meetings with Mark Zuckerberg and other high-level Facebook representatives; as a result Facebook has agreed to hand over data on French users suspected of spewing hate speech online. This is an international first, and according to a counsel at law firm Linklaters, "a strong signal in terms of regulation" (Rosemain 2019). EU countries have threatened to fine social networks up to 50 million euro per year if they continue to fail to act faster (Thomasson 2017).

Social media has contributed to a more open and connected world (Rainie, Anderson, and Albright 2017). It has promoted Western liberal values through its effects on protest mobilization (Breuer, Landman, and Farquhar 2015), community building (Jackson, Bailey, and Foucault Welles 2018), and accountability in governments and institutions (Enikolopov, Petrova, and Sonin 2018). It is critical that we do not lose the benefits of these platforms as they operate under increased regulation and scrutiny. So, while automated systems can and should enforce uniform policies for user behavior that fosters welcoming online environments, this should not come at the cost of dissenting and fringe views.

This paper aims to navigate the challenge of creating systems that distinguish hateful content from abusive, normal, and spam content. We echo the concerns of Facebook's head of public policy by asking:

How can we improve the performance of automated systems on identifying hate speech when they must learn from few hateful samples?

This research surveys the capabilities and limitations of state-of-the-art (SOTA) deep learning model architectures with an emphasis on comparing the inclusion of user behavior features to using advanced pretrained word embeddings. We begin by describing related research and the model architectures which have shaped the current landscape of automated hate speech detection. We explain how this research

informs our methodology and go on to describe our data, how it is collected and processed, the details of our model architectures, and metrics for evaluation.

We find that Google's pretrained BERT embeddings add enough semantic information to our model that the inclusion of user behavior metrics does not offer further improve performance.

Related Work

Using Deep Learning Approaches

In spite of the growing phenomenon of hate speech, prior to 2016, NLP research on hate speech was limited (ElSherief et al. 2018) (Waseem and Hovy 2016). Since then, statistical machine learning techniques like logistic regression and naive Bayes classifiers have been applied to this area with success; logistic regression in particular is widely considered a baseline model for experiments today (Waseem and Hovy 2016) (Badjatiya et al. 2017) (Schmidt and Wiegand 2017). Recently, the aforementioned pressures, coupled with concurrent advances in NLP and deep learning, has sparked a resurgence in interest in the field. The success of deep neural networks on NLP tasks like sentiment analysis (Tang et al. 2014), named entity recognition (Habibi et al. 2017), and part-of-speech tagging (Santos and Zadrozny 2014) have made it a natural choice to apply to the task of classifying tweets or posts as hateful.

The baseline machine learning model choice, which tends to perform well, is logistic regression (Davidson et al. 2017) (Xiang et al. 2012). Another baseline model, support vector machines (SVMs)–with linear kernels–perform similarly to logistic regression in practice (Del Vignal et al. 2017). Linear models tend not to capture the sparse features in document classification tasks, undermining the detection of relations between words or symbols in sentences. CNNs and RNNs (particularly LSTMs) have achieved SOTA performance due to their ability to capture long-term dependencies in a sentence, tweet, or post through their ability to maintain information in memory for a period of time (Badjatiya et al. 2017) (Pitsilis, Ramampiaro, and Langseth 2018).

The Difficulty with Hate Speech Datasets

A connected area of research is the task of creating robust hate speech datasets. The common practice is to deploy a collection of tweets or posts to a crowdsourcing site, where a number of human annotators identify an appropriate classification (i.e. hateful, normal, abusive, spam). This is problematic for a few reasons and highlights the ways in which human-annotated data reflects social biases. First, the subjective nature of identifying hate speech makes it difficult to extract ground truth labels for tweets or posts (Founta et al. 2018). There are many definitions of hate speech, which vary across different languages, cultures, and governments. Many of the definitions involve the hateful targeting of members of a group, like this one:

> "Language used to express hatred towards a targeted individual or group, or is intended to be derogatory, to humiliate, or to insult the members of the group, on the basis of attributes such as race, religion, ethnic origin, sexual orientation, disability, or gender." (Davidson et al. 2017)

Second, because there is no universally agreed upon definition of hate speech [18], it is often difficult for human annotators to separate hate speech from offensive language. Given the legal consequences of hate speech, it is important that annotators and machine learning classifiers distinguish between hateful language and offensive language (Founta et al. 2018) (Davidson et al. 2017). It has been found that showing users a definition of hate speech does not improve annotation reliability, hinting that the presence of hate speech is not a binary decision and annotators likely need more context (Schmidt and Wiegand 2017), such as sample post-label pairings.

Third, it is likely that annotators skim through tweets or posts too quickly and fail to pick up on the context of the language. Past research demonstrates that annotators will incorrectly label normal text as hate speech if it contains hate or curse words (Davidson et al. 2017). Annotators are likely to miss out on hate speech without a slur as well as interpret racist and homophobic slurs as hateful but sexist slurs as 'just offensive' (Waseem and Hovy 2016) (Davidson et al. 2017).

The Inclusion Of User Behavior Metrics

Hateful users tend to make themselves central to the Twitter social network: they target more popular and popular users and they have more statuses and followees per day, although they have younger accounts (ElSherief et al. 2018) (Ribeiro et al. 2017). While this may increase the user's visibility, the inclusion of hateful or antagonistic content in a tweet reduces the rate of retweet by a factor of 45 (Burnap and Williams 2015). Despite this, hateful users are central to Twitter reply networks as determined by features such as betweenness and eigenvector centrality (Ribeiro et al. 2017). Furthermore, the modeling of reply trees–rooted at a tweet and joined by replies–reveals that the in-degree of the tweet, or the number of retweets, plays an important role in the overall shape of the reply tree and a component of the Twitter reply network as a whole (Nishi et al. 2016).

Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) is a method of topic modeling that extracts the main topics in text. It has been used to improve the performance of linear machine learning classifiers on the task of identifying abusive social media content (Xu et al. 2012). While LDA-derived topics have been used as annotations (Wiedemann et al. 2018), to the best of our knowledge it has not been included in feature embeddings in order to improve the performance of deep learning classifiers on the task.

Methodology

The overall goal of this research is to conduct a survey of deep learning model architectures, embedding choices, and feature inputs in order to automate the identification of hateful tweets from a dataset with few hateful samples. Unlike previous work, we experiment with user behavior metrics in multiple input model architectures to provide our deep

Figure 1: Word clouds of tweets from the hateful, abusive, normal, and spam classes.

learning classifiers with more context to identify hate speech with.

Dataset

The dataset for this research comes from Founta et al.'s work that describes the process of large scale crowdsourcing for annotations of hateful, normal, abusive, and spam tweets (Founta et al. 2018). Founta et al. provide status ids for others to obtain tweets through the Twitter API. After learning that a number of tweets were no longer available on Twitter, they provided 100k tweets' status ids and their associated majority labels. Upon using the Twitter API, we find that only 64% of those tweets are still available online; we assume the remainder have been taken down by Twitter or the users. The class distribution is as follows: **4% hateful, 20% abusive, 62% normal, and 14% spam**; there are **64,149 tweets total**. Half the tweets achieved annotation majority with an agreement of more than 3 out of 5 votes, which should give us some confidence in the annotations. Figure 1 illustrates a word cloud of each class.

The average retweet count of hateful content is 15% higher than the average retweet count of normal content; the average favorite count of hateful content is 76% higher than the average favorite count of normal content. Finally, on average hateful content is a reply to existing tweets 31% more often than normal content is.

For each tweet, if it is in reply to a tweet we use Twitter's API to collect the context tweet. 19% of the total tweets are a reply. Of those, 14% are no longer available on Twitter. For each tweet, we crawl the user timeline of the author and collect their 200 most recent tweets, giving us around 12.8 million additional tweets for LDA topic modeling.

Data Processing

We clean the tweets by tokenizing, lowercasing, and removing punctuation. We keep URLs because they provide important context; tweets with URLs are more likely to get retweeted (Suh et al. 2010). We experiment with three types of text embeddings:

1. **Term Frequency-Inverse Document Frequency (TF-IDF) embeddings.** TF-IDF fits the training set into a weighted vector by normalized frequency of the 10,000 most common words in our vocabulary. Our validation and test set are transformed using the learned weights.
2. **Pretrained Twitter embeddings.** These embeddings are from a Word2Vec model trained on 400 million raw English tweets, with an embedding dimension of 400 (Godin et al. 2015).
3. **Pretrained BERT embeddings.** Google's BERT, or Bidirectional Encoder Representations from Transforms, is a novel method of pre-training language representations which obtains SOTA results on a range of NLP tasks devlin2018BERT. Specifically, we use the BERT-12-768-12 model with an embedding dimension of 768.

For our word embeddings, we trim or pad each tweet to 17 words, the average size of a tweet in our dataset. When padding, we add randomly generated embeddings so that each tweet embedding is a n-gram, where n=17. To account for the imbalance of classes in our dataset, for our training experiments we draw samples through a multinomial distribution that accounts for each class weight. Our balanced batches allow the classifier to learn class representations more consistently each epoch. We do not do this for validation and testing because a) we run the risk of tweet samples that are seen multiple times or never in our evaluation and b) it is not a realistic representation of tweets in the wild.

Model Architectures

Logistic Regression Our baseline model is multi-class logistic regression which uses the L2 norm penalty as a form of regularization.

Multilayer Perceptron. Known as a "vanilla neural network", this model consists of fully connected layers with the non-linear activation function leaky ReLu applied. We use Leaky ReLus here and in other models because it allows a small, positive gradient when the unit is not active and as a result, tends to perform better than nonleaky ReLu in practice.

Convolutional Neural Network (CNN). While this model is traditionally used for image inputs, it has performed well on NLP tasks and has had success when applied to hate speech detection (Gambäck and Sikdar 2017). The convolutional layer computes the output of neurons (i.e. mathematical operations) that are connected to local regions in the input. It consists of a set of learnable filters that produce an activation map used to ultimately compute the class scores. Our CNN consists of a series of convolutional layers, batch normalization, leaky ReLu, and dropout. The output is processed through a max-pooling layer and then a fully connected layer which computes class probabilities. Our CNN contains a context list that gives every layer access to all the previous layers, allowing for more connectivity and greater chance of the network learning from long-term dependencies in the tweet.

DenseNet. CNNs are more accurate and efficient if they contain shorter connections between layers close to the input and output (Huang et al. 2017). DenseNet leverages this

observation by connecting every layer to every other layer: for each layer, the feature maps of all preceding layers are used as inputs into all subsequent layers. Our DenseNet implementation has achieved SOTA results for image datasets; we experiment with its application to NLP.

LSTM. LSTMs (and RNNs more broadly) improve on traditional deep neural networks by propagating information in both directions. This loop in the architecture acts as a memory state by which the network makes adjustments in the information flow, allowing the model to selectively remember or forget. This model has performed well on NLP tasks in practice because of its ability to remember dependencies and between sentences; it is applied to the task of hate speech detection by (Badjatiya et al. 2017) (Del Vignal et al. 2017) (Gambäck and Sikdar 2017). Our implementation passes our input through a series of LSTM layers to a final fully connected layer.

Experiment

Phase 1: Tweet Embedding

We compare our deep learning models to a baseline logistic regression model to begin with an idea around how much of an improvement they can offer. Here the feature embedding to our deep learning models are simply the tweet embeddings. Table 1 enumerates regularization choices, number of layers, and model-specific hyperparameter values. Regularization reduces the number of parameters and amount of computation in the network, which helps to control overfitting.

Phase 2: Tweet + Reply Embeddings

The goal of this round of experiments is to apply some type of context to our embeddings. First, because of the statistics collected around the behavior of hateful users and retweeting, we define pairs of tweets: the original tweet and a context tweet. The context tweet is for the case that the tweet was a response to something else. If the tweet is not a response to something else, we add null embeddings to indicate that there is no context for the tweet. Although only 19% of our tweets in the dataset are replies, the extra information around context (when it is there and not there) may help our network learn in some way.

Next, because tweet in-degree has been shown to be significant ((Nishi et al. 2016), see Background) we focus on in-degree in terms of number of times someone has retweeted a given tweet and number of times someone has favorited a given tweet. Our maximum **retweet value is 154,565** and our **maximum favorite value is 27,741**. Because these numbers are so large, we log each (and use 0 when the count is 0), so as to not skew the learned weights. We concatenate the logged retweet and favorite counts to our tweet and reply embeddings. We are interested to see if the network can better learn from the retweet and favorite numbers as a global measure of context.

Here and for the remaining experiments, we shift to building a neural network with multiple inputs in order to have the network learn from the annotated tweet in addition to other types of embeddings. We have the annotated tweet embedding and the context tweet embedding as separate inputs; they are processed by different parts of the model architecture. The learned features for each are concatenated and fed into a final fully connected layer. See Figure 2 for a visualization of a multiple input network architecture.

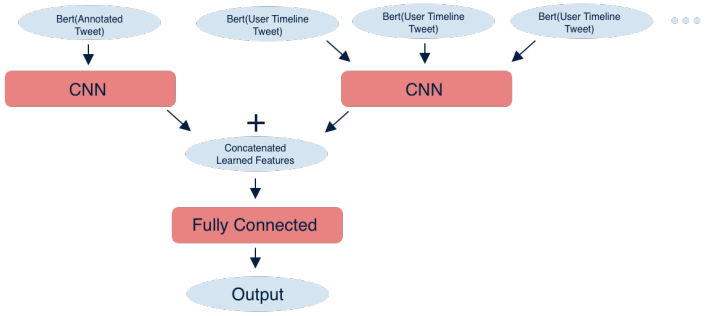


Figure 2: Multiple Input CNN Architecture

Phase 3: Tweet + Dominant Topic Embeddings

This phase aims to add context in a more sophisticated way. For each tweet in our dataset, we crawl the author's user timeline and collect 200 tweets. We then conduct topic modeling through the LDA approach (discussed in Background). For each of our users, we find the dominant topic in their timeline tweets and create embeddings of a sequence of the top 10 words related to the topic. We choose the same embedding that is used to embed our tweets for a given experiment. We also concatenate the coherence and perplexity scores of the user's timeline to each embedded topic word in order to add a global measure of topic modeling. Perplexity is a measure of how well a probability model can predict a topic (lower is better) and coherence captures how well topics can be defined (higher is better).

Our multiple input model architecture processes the topic embeddings and tweet embeddings. Again, we concatenate the learned features and use this as input to the final fully connected layer.

Hyperparameter Tuning & Evaluation

We tune our best performing models–by model type and embedding type–by experimenting with learning rate, regularization, number of layers, and the model specific parameter. We define best performing model as the model with the greatest validation f-score on the hateful classes. We evaluate our results by overall metrics of f-score, precision, and recall as well as f-score of the hateful and abusive classes. This is because we are concerned with our classifier's ability to correctly identify hateful and abusive samples. To achieve stability in the results, we evaluate the mean and standard deviation of 3 experiments each.

MODEL	REGULARIZATION	LAYERS
LR	L2-NORM PENALTY	-
MLP	MP, λ=1E-4	3 FC-LERELU-FC
CNN	MP, BN, δ=0.5, λ=1E-4	3 CONV-BN-LERELU-FC
DENSENET	AP, BBN, δ=0.5, λ=1E-4	4 DENSE-BBN-RELU-CONV-FC
LSTM	δ=0.5, λ=1E-4	3 LSTM-FC

Table 1: Baseline model architectures. Terms include Weight decay (λ), Max Pooling (MP), Average Pooling (AP), Dropout (δ), Batch Normalization (BN), and Bottleneck Layer, which includes BN (BBN). Model-specific hyperparameters include CNN filters=8, DENSENET growth-rate=12, LSTM hidden-layers=5.

Limitations

We acknowledge that there are alternative experiment design choices within our methodology that are worth exploring in future work. First, we choose to trim or pad tweets instead of processing tweets in batch sizes by tweet length or by applying a pooling layer. Second, we do not experiment with character embeddings. Third, we are limited by the data returned by Twitter's publicly available API. The reply tree study conducted by (Nishi et al. 2016) in 2016 is no longer easily replicable, as Twitter only provides a subset of data. For example, when trying to collect replies from a tweet with 78 replies, only 15 were available through the API. According to Twitter: *Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface.*

Results

Phase 1 Results

Table 2 illustrates Phase 1 results across our evaluation metrics. Randomly classifying a sample as hate speech with the distribution found in our dataset results in **4%** classification accuracy. Relative to this, logistic regression is a strong baseline with **0.20** f-score on the hateful class. This is in line with the results from (Gambäck and Sikdar 2017). The BERT embeddings perform the best across all models, with CNN-BERT being the best performing combination for both f-score on the hateful class (**0.29**) and f-score overall (**0.71**). The MLP does not perform well; we hypothesize that this is because of the scarcity of hateful tweets in addition to the fact that our feature embeddings only consist of tweet embedddings. As a result, the MLP and LSTM models do not have enough signals or cues to interpret the input. In contrast, we hypothesize that the CNN and DenseNet models perform well because, similar to how they handle images, they are interpreting patterns from an otherwise noisy input.

Phase 2 Results

Table 3 illustrates Phase 2 results across our evaluation metrics. While different model-embedding combinations see different performance boosts/losses from this round of experiments, we achieve our highest score with the CNN-BERT with network metrics as embedding coefficients, but only by a slight margin. We see the most benefit to TF-IDF embeddings for CNN and DenseNet, where the overall f-score is improved with both types of reply metrics. For pretrained Twitter and BERT embeddings, using reply metrics as embedding coefficients by passing the context tweets through a tweet-level version of our model did not help the final model in learning. MLP and LSTM were likely not affected much because similar to our hypothesis from phase 1, the sparse amount of hateful samples coupled with the lack of cues or signal around the content of our tweets and reply tweets inhibits the model from learning. This tells us that it is not enough to feed our models more context: we need to find a better way to define that context within our feature embeddings.

Phase 3 Results

Figure 3 illustrates the topic keywords from our LDA model of user timeline tweets from a user that authored a sample hateful tweet. We see that our largest topic, with **16.1% of tweet tokens**, includes terms like 'potus', 'russian', 'nytime', and 'vote', implying that this user frequently tweets about US political issues. Another tweet annotated as hateful listed the dominant topic's top words as 'film', 'good', 'award', 'location', 'series', 'international'; upon further inspection of the tweet and the account, we found that the tweet was quoting a film and was mislabelled as hateful. In this case, topic modeling highlighted the difference between hateful users and hateful tweets by providing much needed context.

Table 4 illustrates phase 3 results across our evaluation metrics. We see that we achieve our best performing model on the hateful class, CNN-BERT, by using topic modeling features, but with only a slight improvement from our previous experiment phases. Figure 2 illustrates this multiple input model architecture. Notably, adding the user behavior metrics to the TF-IDF embedded models improves performance for the CNN and DenseNet classifiers, more than in phase 2 of our experiments. This tell us that the pretrained Twitter and BERT embeddings add a great deal of semantic value and cannot be much improved upon by user behavior metrics, such as topics from the user timeline. In contrast, TF-IDF embeddings do not offer this, and dominant topic words compensate for that.

Hyperparameter Tuning

We take our best performing model and embedding combination, CNN-BERT with user behavior metrics, and tune it by experimenting with dropout, number of layers, number of filters, and learning rate. Because the CNN-BERT

MODEL	F_H	F_A	F	P	R
LR-TFIDF	0.13 ± 0.05	0.28 ± 0.12	0.52 ± 0.05	0.53 ± 0.07	0.57 ± 0.06
LR-TWIT	0.15 ± 0.01	0.63 ± 0.0	0.65 ± 0.01	0.65 ± 0.01	0.64 ± 0.01
LR-BERT	0.2 ± 0.0	0.74 ± 0.0	0.71 ± 0.0	0.71 ± 0.0	0.72 ± 0.0
MLP-TFIDF	0.0 ± 0.0	0.22 ± 0.16	0.05 ± 0.02	0.03 ± 0.01	0.18 ± 0.03
MLP-TWIT	0.08 ± 0.04	0.42 ± 0.01	0.32 ± 0.13	0.57 ± 0.03	0.32 ± 0.07
MLP-BERT	0.05 ± 0.04	0.24 ± 0.16	0.19 ± 0.18	0.33 ± 0.22	0.23 ± 0.16
CNN-TFIDF	0.1 ± 0.01	0.22 ± 0.09	0.38 ± 0.06	0.49 ± 0.06	0.33 ± 0.06
CNN-TWIT	0.27 ± 0.04	0.78 ± 0.02	0.68 ± 0.01	0.75 ± 0.01	0.65 ± 0.01
CNN-BERT	0.29 ± 0.03	0.75 ± 0.02	0.71 ± 0.03	0.76 ± 0.02	0.69 ± 0.03
LSTM-TFIDF	0.18 ± 0.01	0.62 ± 0.06	0.52 ± 0.03	0.67 ± 0.02	0.49 ± 0.03
LSTM-TWIT	0.22 ± 0.01	0.67 ± 0.02	0.62 ± 0.02	0.72 ± 0.01	0.58 ± 0.02
LSTM-BERT	0.25 ± 0.02	0.74 ± 0.01	0.67 ± 0.01	0.76 ± 0.0	0.64 ± 0.01
DENSE-TFIDF	0.08 ± 0.0	0.14 ± 0.1	0.27 ± 0.19	0.29 ± 0.21	0.26 ± 0.15
DENSE-TWIT	0.22 ± 0.01	0.74 ± 0.01	0.67 ± 0.01	0.7 ± 0.0	0.66 ± 0.01
DENSE-BERT	0.26 ± 0.01	0.73 ± 0.0	0.68 ± 0.01	0.72 ± 0.0	0.67 ± 0.01

Table 2: **Phase 1 Experiment Results, Tweet Embeddings.** F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall. The red highlight indicates best performing model and the blue highlight indicates the TF-IDF embeddings that see the most improvement with the inclusion of user behavior metrics.

MODEL	F_H	F_A	F	P	R
MLP-TFIDF	0.06 ± 0.04	0.11 ± 0.16	0.03 ± 0.03	0.02 ± 0.02	0.1 ± 0.07
MLP-TWIT	0.12 ± 0.01	0.36 ± 0.06	0.26 ± 0.14	0.44 ± 0.2	0.28 ± 0.09
MLP-BERT	0.07 ± 0.02	0.38 ± 0.01	0.4 ± 0.0	0.52 ± 0.02	0.38 ± 0.02
CNN-TFIDF	0.06 ± 0.04	0.23 ± 0.08	0.26 ± 0.14	0.38 ± 0.1	0.28 ± 0.05
CNN-TWIT	0.25 ± 0.01	0.77 ± 0.03	0.66 ± 0.03	0.75 ± 0.01	0.64 ± 0.03
CNN-BERT	0.29 ± 0.01	0.76 ± 0.02	0.7 ± 0.02	0.75 ± 0.01	0.68 ± 0.01
LSTM-TFIDF	0.19 ± 0.0	0.69 ± 0.0	0.56 ± 0.0	0.7 ± 0.0	0.53 ± 0.0
LSTM-TWIT	0.13 ± 0.0	0.41 ± 0.0	0.5 ± 0.0	0.61 ± 0.0	0.46 ± 0.0
LSTM-BERT	0.22 ± 0.0	0.7 ± 0.0	0.65 ± 0.0	0.74 ± 0.0	0.61 ± 0.0
DENSE-TFIDF	0.11 ± 0.01	0.3 ± 0.16	0.48 ± 0.07	0.5 ± 0.06	0.48 ± 0.07
DENSE-TWIT	0.09 ± 0.01	0.31 ± 0.07	0.46 ± 0.03	0.55 ± 0.02	0.43 ± 0.04
DENSE-BERT	0.19 ± 0.01	0.69 ± 0.03	0.64 ± 0.02	0.7 ± 0.0	0.62 ± 0.02
MLP-TFIDF	0.0 ± 0.0	0.22 ± 0.16	0.2 ± 0.19	0.15 ± 0.16	0.34 ± 0.19
MLP-TWIT	0.06 ± 0.02	0.37 ± 0.02	0.14 ± 0.02	0.38 ± 0.21	0.23 ± 0.01
MLP-BERT	0.05 ± 0.0	0.35 ± 0.01	0.13 ± 0.03	0.29 ± 0.23	0.21 ± 0.02
CNN-TFIDF	0.1 ± 0.01	0.25 ± 0.07	0.41 ± 0.08	0.5 ± 0.05	0.39 ± 0.12
CNN-TWIT	0.26 ± 0.03	0.77 ± 0.01	0.66 ± 0.01	0.75 ± 0.01	0.63 ± 0.01
CNN-BERT	0.3 ± 0.01	0.76 ± 0.0	0.69 ± 0.01	0.76 ± 0.0	0.67 ± 0.01
LSTM-TFIDF	0.17 ± 0.01	0.64 ± 0.04	0.52 ± 0.03	0.67 ± 0.02	0.49 ± 0.03
LSTM-TWIT	0.21 ± 0.01	0.65 ± 0.04	0.6 ± 0.03	0.71 ± 0.02	0.56 ± 0.02
LSTM-BERT	0.25 ± 0.0	0.73 ± 0.01	0.66 ± 0.01	0.75 ± 0.01	0.63 ± 0.01
DENSE-TFIDF	0.07 ± 0.02	0.29 ± 0.05	0.49 ± 0.06	0.53 ± 0.07	0.5 ± 0.1
DENSE-TWIT	0.22 ± 0.02	0.75 ± 0.01	0.67 ± 0.0	0.71 ± 0.02	0.65 ± 0.0
DENSE-BERT	0.25 ± 0.0	0.73 ± 0.01	0.68 ± 0.0	0.71 ± 0.0	0.67 ± 0.01

Table 3: **Phase 2 Experiment Results, Tweet + Reply (Top) & Network Metrics (Bottom) Embeddings.** F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall. The red highlight indicates best performing model and the blue highlight indicates the TF-IDF embeddings that see the most improvement with the inclusion of user behavior metrics.

MODEL	F_H	F_A	F	P	R
MLP-TFIDF	0.14 ± 0.02	0.47 ± 0.02	0.39 ± 0.04	0.58 ± 0.01	0.39 ± 0.03
MLP-TWIT	0.08 ± 0.04	0.36 ± 0.1	0.31 ± 0.16	0.41 ± 0.24	0.32 ± 0.11
MLP-BERT	0.06 ± 0.01	0.39 ± 0.02	0.35 ± 0.09	0.53 ± 0.03	0.34 ± 0.07
CNN-TFIDF	0.24 ± 0.01	0.82 ± 0.01	0.64 ± 0.02	0.77 ± 0.01	0.61 ± 0.02
CNN-TWIT	0.19 ± 0.07	0.71 ± 0.08	0.62 ± 0.06	0.71 ± 0.04	0.61 ± 0.05
CNN-BERT	0.3 ± 0.02	0.76 ± 0.01	0.68 ± 0.03	0.76 ± 0.01	0.65 ± 0.03
LSTM-TFIDF	0.2 ± 0.0	0.72 ± 0.01	0.59 ± 0.01	0.73 ± 0.0	0.55 ± 0.01
LSTM-TWIT	0.21 ± 0.01	0.66 ± 0.02	0.61 ± 0.02	0.72 ± 0.01	0.57 ± 0.02
LSTM-BERT	0.25 ± 0.01	0.73 ± 0.01	0.65 ± 0.01	0.75 ± 0.01	0.62 ± 0.01
DENSE-TFIDF	0.21 ± 0.0	0.76 ± 0.01	0.66 ± 0.01	0.72 ± 0.02	0.64 ± 0.01
DENSE-TWIT	0.19 ± 0.01	0.74 ± 0.02	0.69 ± 0.01	0.71 ± 0.0	0.67 ± 0.01
DENSE-BERT	0.24 ± 0.0	0.73 ± 0.0	0.68 ± 0.02	0.72 ± 0.0	0.66 ± 0.02

Table 4: **Phase 3 Experiment Results, Tweet + User Topic Embeddings**. F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall. The red highlight indicates best performing model and the blue highlight indicates the TF-IDF embeddings that see the most improvement with the inclusion of user behavior metrics.

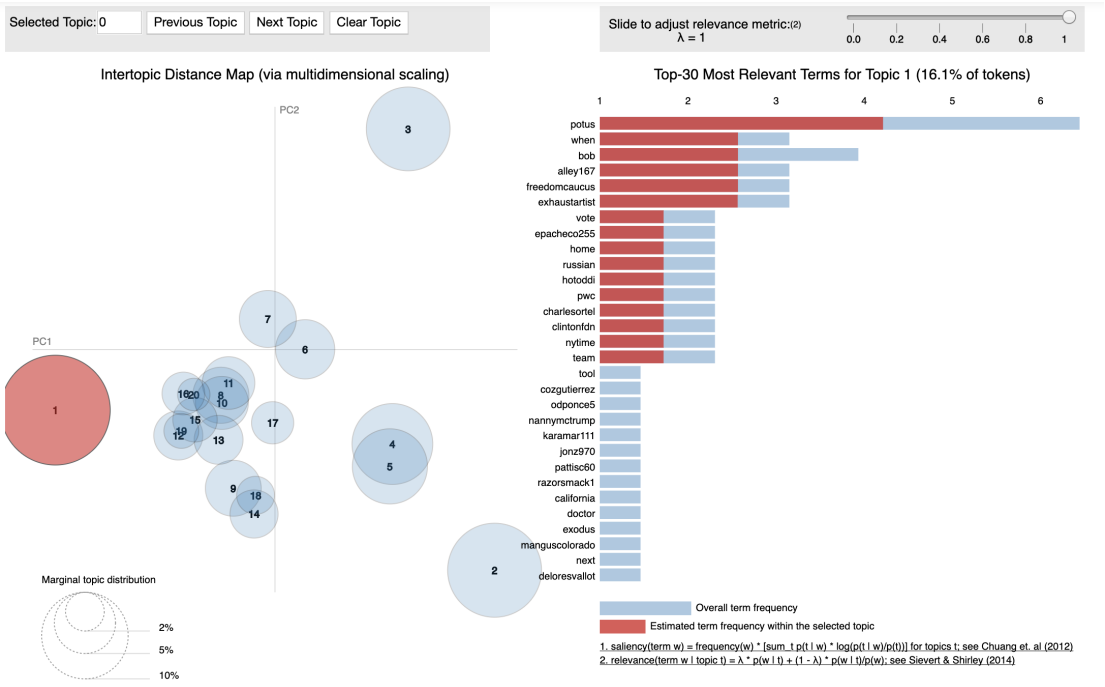


Figure 3: **Topic LDA Modeling of User's Timeline from Sample Hateful Tweet.** Sample hateful tweet is: "Muslims immigrants are living like they're used to where their from, squalor, filth and violence are all they know." This illustrates output from the Gensim library for LDA Topic modeling.

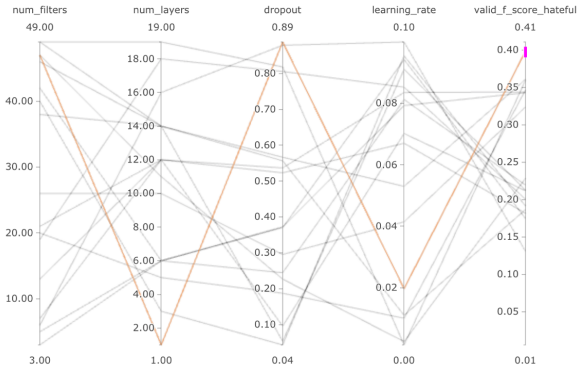


Figure 4: **Hyperparameter Tuning Parallel Coorodinates Chart.** This chart illustrates the parameter choices and validation f-score on the hateful class across 17 experiments.

model converged on average at **epoch 44**, we do not experiment with number of epochs. Figure 4 illustrates the 17 rounds of tuning experiments, with runtimes ranging from 5 hours and 20 minutes to 4 minutes (see Appendix X for machine configurations). We use the Bayes algorithm with the adaptive Parzen-Rosenblatt estimator (facilitated by the Comet.ML library's optimization framework (Comet.ml)), which balances exploring unknown space with exploiting the known hyperparameter values that yield the best results. The following values perform best of the values explored: **num-filters=47, num-layers=1, dropout=0.8877, learning-rate=0.0196**. This leaves us with a single layer CNN with many filters and with a large regularization value for dropout.

Tuned Model Results Across Experiments

Table and Figure 5 illustrate the tuned model performances. All the tuned models demonstrate higher f-score on the minority class, with the tuned phase 2 reply-pairings model performing best at **0.33** $\pm$ **0.01**. This model also performs best on all classes, with an overall f-score of **0.72** $\pm$ **0.01**. It achieves the highest recall but not the highest precision of the experiments conducted so far. We hypothesize that the tuned phase 2 reply-pairings model performs better than the other models because the single layer CNN with many filters is overfitting with the tweet and topic embeddings. With only the tweet and reply-pairing BERT embeddings, the network seems to have a more generalizeable notion of context.

In order to better understand our precision and recall scores, we analyze a confusion matrix in table 6. We see that the hateful class was correctly classified **40%** of the time and was most commonly confused with the abusive class. The abusive class was correctly classified **82%** of the time, which is higher than the model's ability to detect normal tweets or spam tweets. In the below samples, the model predicts a tweet is hateful when it is labeled as hateful, abusive, and normal, respectively. Personally identifiable information is removed.

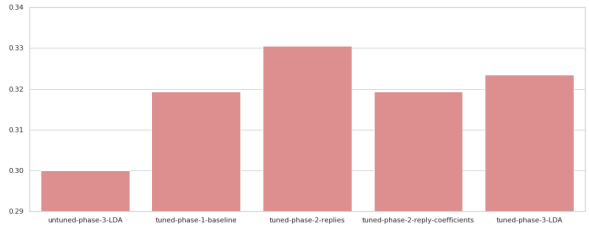


Figure 5: **Tuned CNN-BERT Experiment Results.** All the tuned models demonstrate higher f-score on the minority class, with the tuned phase 2 model performing best at 0.33.

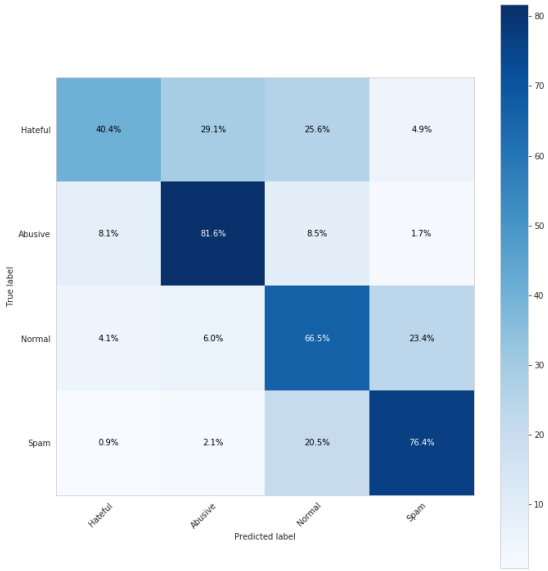


Figure 6: **Confusion Matrix of Final Model**. The model can detect the abusive class most accurately.

*RHate when folk ask me questions that they already know the answers to . B*TCH what you fishing for ?* (Hateful Label)

That was the most patronising thing I've ever heard about young people and propaganda. What about old people and the mail?! (Abusive Label)

TEACHERS too! Another "trusted" member of society, doing the nasty with teen. (Normal Label)

There are a few problems with this. First, the tweet that is labelled as hateful does not appear to be hate speech from the definitions explored in our background. Second, the tweet that is labelled as normal could be considered abusive towards teachers. All of these tweets have negative sentiment, so it does appear our model has picked up on that. This deeper dive into the actual content that is misclassified hints at a problem with the annotation quality of our dataset.

MODEL	F_H	F_A	F	P	R
PHASE-1	0.32 ± 0.02	0.77 ± 0.01	0.71 ± 0.01	0.76 ± 0.01	0.69 ± 0.01
PHASE-2-1	0.33 ± 0.01	0.78 ± 0.01	0.72 ± 0.01	0.76 ± 0.0	0.7 ± 0.01
PHASE-2-2	0.32 ± 0.0	0.77 ± 0.01	0.7 ± 0.01	0.76 ± 0.0	0.69 ± 0.01
PHASE-3	0.32 ± 0.01	0.78 ± 0.01	0.7 ± 0.02	0.77 ± 0.01	0.68 ± 0.02

Table 5: **Tuned CNN-BERT Model Performance.** F is f1 score, F_H indicates hateful class, and F_A indicates abusive class. P is precision and R is recall.

Conclusion

Our final model successfully picked up on negative tweet sentiment and identified the abusive class at the highest rate, of **82%** accuracy and **0.78** f-score. The models that used Google's pretrained BERT embeddings performed better than TF-IDF and Twitter pretrained embeddings across most models in our three phases of experiments. CNN-BERT outperformed the logistic regression, MLP, LSTM, and DenseNet models for all three phases of experiments. Before tuning, our best performing model is the CNN multiple input model architecture with tweet and user topic BERT embeddings. After tuning, our best performing model is the CNN multiple input model architecture with tweet and reply BERT embeddings. We hypothesize that this is because the parameter choices of a single layer, 47 filter CNN with high dropout will overfit with BERT and a large measure of user context. The pretrained BERT embeddings add enough semantic information to give us our most competitive models and adding additional metrics of context through aspects of the social network actually hurts performance.

The task of automating the detection of hate speech on social media platforms remains a challenge, in part due to the difficulty in obtaining high-quality, large-scale annotated datasets and the scarce hateful samples available for machine learning models to learn from. Our experiments reflect this and suggest that improving the quality and consistency of annotations in our dataset is likely to result in more accurate automated systems.

Even so, our final model offers a significant improvement on detecting hate speech, as we are able to improve on our logistic regression baseline performance on the hateful class by **.13** f-score on a dataset with scarce hateful labels. If we were to randomly annotate a tweet as hateful with 4% probability, we'd achieve around 4% accuracy on the hateful class. Thus, we interpret the final f-score on the hateful class of **.33** as relatively high.

The benefits of using BERT in our experiments suggest that further work on pretrained embeddings, from larger or more targeted datasets, holds more promise than adding information specific to the social media platform. However, LDA topic modeling is a useful tool for gauging the quality of the dataset annotations, as it allowed us to quickly see a hateful tweet was mislabeled by clarifying that the user is a film reviewer. Policy and decision-makers may benefit from using topic modeling and other forms of user behavior metrics in a contagion tool, i.e. a tool that monitors trends and patterns of hate speech and how they might spread through a network or change in response to triggering events.

This research hopes to inform interdisciplinary coalitions on the current landscape of hate speech detection. While our model's were able to classify abusive text with a high accuracy, they struggled with the task of distinguishing hate speech from normal and abusive content. A "good" automated hate speech detector may mean detecting detecting 30% of hate speech and passing merely abusive content to human moderators. The technical limitations described indicate that policy-makers must cooperate with social media companies in order to guide the handling of classified hateful and abusive content. Ultimately, social media has contributed to a more open and connected world. It is critical that we mitigate the negative consequences of hate speech while preserving the benefits of online discussion.

References

[Awan 2017] Awan, I. 2017. Cyber-extremism: Isis and the power of social media. *Society* 54(2):138–149.

[Badjatiya et al. 2017] Badjatiya, P.; Gupta, S.; Gupta, M.; and Varma, V. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 759–760. International World Wide Web Conferences Steering Committee.

[Blei, Ng, and Jordan 2003] Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

[Breuer, Landman, and Farquhar 2015] Breuer, A.; Landman, T.; and Farquhar, D. 2015. Social media and protest mobilization: Evidence from the tunisian revolution. *Democratization* 22(4):764–792.

[Burnap and Williams 2015] Burnap, P., and Williams, M. L. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet* 7(2):223–242.

[Cave 2019] Cave, D. 2019. Countries want to ban 'weaponized' social media. what would that look like?

[Cheng et al. 2017] Cheng, J.; Bernstein, M.; Danescu-Niculescu-Mizil, C.; and Leskovec, J. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*, 1217–1230. ACM.

[Comet.ml] Comet.ml. Comet.ml - supercharging machine learning.

[Davidson et al. 2017] Davidson, T.; Warmsley, D.; Macy, M.; and Weber, I. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*.

[Del Vignal et al. 2017] Del Vignal, F.; Cimino, A.; DellâĂŹOrletta, F.; Petrocchi, M.; and Tesconi, M. 2017. Hate me, hate me not: Hate speech detection on facebook.

[ElSherief et al. 2018] ElSherief, M.; Nilizadeh, S.; Nguyen, D.; Vigna, G.; and Belding, E. 2018. Peer to peer hate: Hate speech instigators and their targets. In *Twelfth International AAAI Conference on Web and Social Media*.

[Enikolopov, Petrova, and Sonin 2018] Enikolopov, R.; Petrova, M.; and Sonin, K. 2018. Social media and corruption. *American Economic Journal: Applied Economics* 10(1):150–74.

[Founta et al. 2018] Founta, A. M.; Djouvas, C.; Chatzakou, D.; Leontiadis, I.; Blackburn, J.; Stringhini, G.; Vakali, A.; Sirivianos, M.; and Kourtellis, N. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*.

[Gambäck and Sikdar 2017] Gambäck, B., and Sikdar, U. K. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online*, 85–90.

[Godin et al. 2015] Godin, F.; Vandersmissen, B.; De Neve, W.; and Van de Walle, R. 2015. Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the workshop on noisy user-generated text*, 146–153.

[Habibi et al. 2017] Habibi, M.; Weber, L.; Neves, M.; Wiegandt, D. L.; and Leser, U. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics* 33(14):i37–i48.

[Huang et al. 2017] Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

[Jackson, Bailey, and Foucault Welles 2018] Jackson, S. J.; Bailey, M.; and Foucault Welles, B. 2018. # girlslikeus: Trans advocacy and community building online. *New Media & Society* 20(5):1868–1888.

[Marcin 2019] Marcin, T. 2019. Facebook, youtube, whatsapp banned again in sri lanka after violence against muslims.

[Müller and Schwarz 2018] Müller, K., and Schwarz, C. 2018. Fanning the flames of hate: Social media and hate crime. *Available at SSRN 3082972*.

[Nishi et al. 2016] Nishi, R.; Takaguchi, T.; Oka, K.; Maehara, T.; Toyoda, M.; Kawarabayashi, K.-i.; and Masuda, N. 2016. Reply trees in twitter: data analysis and branching process models. *Social Network Analysis and Mining* 6(1):26.

[Pitsilis, Ramampiaro, and Langseth 2018] Pitsilis, G. K.; Ramampiaro, H.; and Langseth, H. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

[Rainie, Anderson, and Albright 2017] Rainie, H.; Anderson, J. Q.; and Albright, J. 2017. *The future of free speech, trolls, anonymity and fake news online*. Pew Research Center Washington, DC.

[Ribeiro et al. 2017] Ribeiro, M. H.; Calais, P. H.; Santos, Y. A.; Almeida, V. A.; and Meira Jr, W. 2017. " like sheep among wolves": Characterizing hateful users on twitter. *arXiv preprint arXiv:1801.00317*.

[Rosemain 2019] Rosemain, M. 2019. Exclusive: In a world first, facebook to give data on hate speech...

[Santos and Zadrozny 2014] Santos, C. D., and Zadrozny, B. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 1818–1826.

[Schmidt and Wiegand 2017] Schmidt, A., and Wiegand, M. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, 1–10.

[Suh et al. 2010] Suh, B.; Hong, L.; Pirolli, P.; and Chi, E. H. 2010. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *2010 IEEE Second International Conference on Social Computing*, 177–184. IEEE.

[Tang et al. 2014] Tang, D.; Wei, F.; Qin, B.; Liu, T.; and Zhou, M. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 208–212.

[Thomasson 2017] Thomasson, E. 2017. German cabinet agrees to fine social media over hate speech.

[Wakefield 2019] Wakefield, J. 2019. Hate speech: Facebook, twitter and youtube told off by mps.

[Waseem and Hovy 2016] Waseem, Z., and Hovy, D. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, 88–93.

[Wiedemann et al. 2018] Wiedemann, G.; Ruppert, E.; Jindal, R.; and Biemann, C. 2018. Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. *arXiv preprint arXiv:1811.02906*.

[Xiang et al. 2012] Xiang, G.; Fan, B.; Wang, L.; Hong, J.; and Rose, C. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 1980–1984. ACM.

[Xu et al. 2012] Xu, J.-M.; Jun, K.-S.; Zhu, X.; and Bellmore, A. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, 656–666. Association for Computational Linguistics.