# Neural Character-based Composition Models for Abuse Detection

**Pushkar Mishra**
Dept. of CS & Technology
University of Cambridge
United Kingdom
pm576@alumni.cam.ac.uk

**Helen Yannakoudakis**
The ALTA Institute
University of Cambridge
United Kingdom
hy260@cl.cam.ac.uk

**Ekaterina Shutova**
ILLC
University of Amsterdam
The Netherlands
e.shutova@uva.nl

## Abstract

The advent of social media in recent years has fed into some highly undesirable phenomena such as proliferation of offensive language, hate speech, sexist remarks, etc. on the Internet. In light of this, there have been several efforts to automate the detection and moderation of such abusive content. However, deliberate obfuscation of words by users to evade detection poses a serious challenge to the effectiveness of these efforts. The current state of the art approaches to abusive language detection, based on recurrent neural networks, do not explicitly address this problem and resort to a generic OOV (out of vocabulary) embedding for unseen words. However, in using a single embedding for all unseen words we lose the ability to distinguish between obfuscated and non-obfuscated or rare words. In this paper, we address this problem by designing a model that can compose embeddings for unseen words. We experimentally demonstrate that our approach significantly advances the current state of the art in abuse detection on datasets from two different domains, namely Twitter and Wikipedia talk page.

## 1 Introduction

*Pew Research Center* has recently uncovered several disturbing trends in communications on the Internet. As per their report (Duggan, 2014), 40% of adult Internet users have personally experienced harassment online, and 60% have witnessed the use of offensive names and expletives. Expectedly, the majority (66%) of those who have personally faced harassment have had their most recent incident occur on a social networking website or app. While most of these websites and apps provide ways of flagging offensive and hateful content, only 8.8% of the victims have actually considered using such provisions.

Two conclusions can be drawn from these statistics: (i) *abuse* (a term we use henceforth to collectively refer to toxic language, hate speech, etc.) is prevalent in social media, and (ii) passive and/or manual techniques for curbing its propagation (such as flagging) are neither effective nor easily scalable (Pavlopoulos et al., 2017). Consequently, the efforts to automate the detection and moderation of such content have been gaining popularity (Waseem and Hovy, 2016; Wulczyn et al., 2017).

In their work, Nobata et al. (2016) describe the task of achieving effective automation as an inherently difficult one due to several ingrained complexities; a prominent one they highlight is the deliberate structural obfuscation of words (for example, *fcukk*, *w0m3n*, *banislam*, etc.) by users to evade detection. Simple spelling correction techniques and edit-distance procedures fail to provide information about such obfuscations because: (i) words may be excessively fudged (e.g., *a55h0le*, *n1gg3r*) or concatenated (e.g., *stupidbitch*, *feminismishate*), and (ii) they fail to take into account the fact that some character sequences like *musl* and *wom* are more frequent and more indicative of abuse than others (Waseem and Hovy, 2016).

Nobata et al. (2016) go on to show that simple character n-gram features prove to be highly promising for supervised classification approaches to abuse detection due to their robustness to spelling variations; however, they do not address obfuscations explicitly. Waseem and Hovy (2016) and Wulczyn et al. (2017) also use character n-grams to attain impressive results on their respective datasets. That said, the current state of the art methods do not exploit character-level information, but instead utilize recurrent neural network (RNN) models operating on word embeddings alone (Pavlopoulos et al., 2017; Badjatiya et al., 2017). Since the problem of deliberately noisy input is not explicitly accounted for, these approaches resort to the use of a generic OOV (out of vocabulary) embedding for words not seen in the training phase. However, in using a single embedding for all unseen words, such approaches lose the ability to distinguish obfuscated words from non-obfuscated or rare ones. Recently, Mishra et al. (2018) and Qian et al. (2018), working with the same Twitter dataset as we do, reported that many of the misclassifications by their RNN-based methods happen due to intentional misspellings and/or rare words.

Our contributions are two-fold: first, we experimentally demonstrate that character n-gram features are complementary to the current state of the art RNN approaches to abusive language detection and can strengthen their performance. We then explicitly address the problem of deliberately noisy input by constructing a model that operates at the character level and learns to predict embeddings for unseen words. We show that the integration of this model with the character-enhanced RNN methods further advances the state of the art in abuse detection on three datasets from two different domains, namely Twitter and Wikipedia talk page. To the best of our knowledge, this is the first work to use character-based word composition models for abuse detection.

## 2 Related Work

Yin et al. (2009) were among the first ones to apply supervised learning to the task of abuse detection. They worked with a linear support vector machine trained on local (e.g., n-grams), contextual (e.g., similarity of a post to its neighboring posts), and sentiment-based (e.g., presence of expletives) features to recognize posts involving harassment.

Djuric et al. (2015) worked with comments taken from the Yahoo Finance portal and demonstrated that distributional representations of comments learned using the *paragraph2vec* framework (Le and Mikolov, 2014) can outperform simpler bag-of-words BOW features under supervised classification settings for hate speech detection. Nobata et al. (2016) improved upon the results of Djuric et al. by training their classifier on an amalgamation of features derived from four different categories: linguistic (e.g., count of insult words), syntactic (e.g. part-of-speech POS tags), distributional semantic (e.g., word and comment embeddings) and n-gram based (e.g., word bi-grams). They noted that while the best results were obtained with all features combined, character n-grams had the highest impact on performance.

Waseem and Hovy (2016) utilized a logistic regression (LR) classifier to distinguish amongst racist, sexist, and clean tweets in a dataset of approximately $16k$ of them. They found that character n-grams coupled with gender information of users formed the optimal feature set for the task. On the other hand, geographic and word-length distribution features provided little to no improvement. Experimenting with the same dataset, Badjatiya et al. (2017) improved on their results by training a gradient-boosted decision tree (GBDT) classifier on averaged word embeddings learnt using a long short-term memory (LSTM) models initialized with random embeddings. Mishra et al. (2018) went on to incorporate community-based profiling features of users in their classification methods, which led to the state of the art performance on this dataset.

Waseem (2016) studied the influence of annotators' knowledge on the task of hate speech detection. For this, they sampled $7k$ tweets from the same corpus as Waseem and Hovy (2016) and recruited expert and amateur annotators to annotate the tweets as *racism*, *sexism*, *both* or *neither*. Combining this dataset with that of Waseem and Hovy (2016), Park et al. (2017) evaluated the efficacy of a 2-step classification process: they first used an LR classifier to separate abusive and non-abusive tweets, and then used another LR classifier to distinguish between the racist and sexist ones. They showed that this setup had comparable performance to a 1-step classification approach based on convolutional neural networks (CNNs) operating on word and character embeddings.

Wulczyn et al. (2017) created three different datasets of comments collected from the English Wikipedia Talk page: one was annotated for personal attacks, another for toxicity, and the third for aggression. They achieved their best results with a multi-layered perceptron classifier trained on character n-gram features. Working with the personal attack and toxicity datasets, Pavlopoulos et al. (2017) outperformed the methods of Wulczyn et al. by having a gated recurrent unit (GRU) to model the comments as dense low-dimensional representations, followed by an LR layer to classify the comments based on those representations.

Davidson et al. (2017) produced a dataset of about $25k$ *racist*, *offensive* or *clean* tweets. They evaluated several multi-class classifiers with the aim of discerning clean tweets from racist and offensive tweets, while simultaneously being able to distinguish between the racist and offensive ones. Their best model was an LR classifier trained using TF–IDF and POS n-gram features coupled with features like count of hash tags and number of words.

## 3 Datasets

Following the proceedings of the *1st Workshop on Abusive Language Online* (Waseem et al., 2017), we use three datasets from two different domains.

### 3.1 Twitter

Waseem and Hovy (2016) prepared a dataset of $16,914$ tweets from a corpus of approximately $136k$ tweets retrieved over a period of two months. They bootstrapped their collection process with a search for commonly used slurs and expletives related to religious, sexual, gender and ethnic minorities. After having manually annotated $16,914$ of the tweets as *racism*, *sexism* or *neither*, they asked an expert to review their annotations in order to mitigate against any biases. The inter-annotator agreement was reported at $\kappa = 0.84$, with further insight that $85\%$ of all the disagreements occurred in the *sexism* class alone.

The authors released the dataset as a list of $16,907$ tweet IDs and their corresponding annotations. We could only retrieve $16,202$ of the tweets with python's *Tweepy* library since some of them have been deleted or their visibility has been limited. Of the ones retrieved, 1,939 (12%) are *racism*, 3,148 (19.4%) are *sexism*, and the remaining 11,115 (68.6%) are *neither*; the original dataset has a similar distribution, i.e., 11.7% *racism*, 20.0% *sexism*, and 68.3% *neither*.

### 3.2 Wikipedia talk page

Wulczyn et al. (2017) extracted approximately $63M$ talk page comments from a public dump of the full history of English Wikipedia released in January 2016. From this corpus, they randomly sampled comments to form three datasets on personal attack, toxicity and aggression, and engaged workers from *CrowdFlower* to annotate them. Noting that the datasets were highly skewed towards the non-abusive classes, the authors oversampled comments from banned users to attain a more uniform distribution.

In this work, we utilize the toxicity and personal attack datasets, henceforth referred to as W-TOX and W-ATT respectively. Each comment in both of these datasets was annotated by at least 10 workers. We use the majority annotation of each comment to resolve its gold label: if a comment is deemed toxic (alternatively, attacking) by more than half of the annotators, we label it as *abusive*; otherwise, as *non-abusive*. 13,590 (11.7%) of the 115,864 comments in W-ATT and 15,362 (9.6%) of the 159,686 comments in W-TOX are abusive. Wikipedia comments, with an average length of 25 tokens, are considerably longer than the tweets which have an average length of 8.

## 4 Methods

We experiment with ten different methods, eight of which have an RNN operating on word embeddings. Six of these eight also include character n-gram features, and four further integrate our word composition model. The remaining two comprise an RNN that works directly on character inputs.

**Hidden-state (HS).** As our first baseline, we adopt the "RNN" method of Pavlopoulos et al. (2017) since it produces state of the art results on the Wikipedia datasets. Given a text formed of a sequence $w_1, \ldots, w_n$ of words (represented by $d$-dimensional word embeddings), the method utilizes a 1-layer GRU to encode the words into hidden states $h_1, \ldots, h_n$. This is followed by an LR layer that classifies the text based on the last hidden state $h_n$. We modify the authors' original architecture in two minor ways: we extend the 1-layer GRU to a 2-layer GRU and use softmax as the activation in the LR layer instead of sigmoid.[1]

Following Pavlopoulos et al., we initialize the word embeddings to GLoVe vectors (Pennington et al., 2014). In all our methods, words not present in the GLoVe set are randomly initialized in the range $\pm 0.05$, indicating the lack of semantic information. By not mapping these words to a single random embedding, we mitigate against the errors that may arise due to their conflation (Madhyastha et al., 2015). A special OOV (out of vocabulary) token is also initialized in the same range. All the embeddings are updated during training, allowing for some of the randomly-initialized ones to get

[1]We also experimented with 1-layer GRU/LSTM and 1/2-layer bi-directional GRUs/LSTMs but the performance only worsened or showed no gains; using sigmoid instead of softmax did not have any noteworthy effects on the results either.

task-tuned (Kim, 2014); the ones that do not get tuned lie closely clustered around the OOV token to which unseen words in the test set are mapped.

**Word-sum (WS).** The "LSTM+GLoVe+GBDT" method of Badjatiya et al. (2017) constitutes our second baseline. The authors first employ an LSTM to task-tune GLoVe-initialized word embeddings by propagating error back from an LR layer. They then train a gradient-boosted decision tree (GBDT) classifier to classify texts based on the average of the constituent word embeddings.[2] We make two minor modifications to the original method: we utilize a 2-layer GRU[3] instead of the LSTM to tune the embeddings, and we train the GBDT classifier on the $L_2$-normalized sum of the embeddings instead of their average.[4]

**Hidden-state + char n-grams (HS + CNG).** Here we extend the *hidden-state* baseline: we train the 2-layer GRU architecture as before, but now concatenate its last hidden state $h_n$ with $L_2$-normalized character n-gram counts to train a GBDT classifier.

**Augmented hidden-state + char n-grams (AUGMENTED HS + CNG).** In the above methods, unseen words in the test set are simply mapped to the OOV token since we do not have a way of obtaining any semantic information about them. However, this is undesirable since racial slurs and expletives are often deliberately fudged by users to prevent detection. In using a single embedding for all unseen words, we lose the ability to distinguish such obfuscations from other non-obfuscated or rare words. Taking inspiration from the effectiveness of character-level features in abuse detection, we address this issue by having a character-based word composition model that can compose embeddings for unseen words in the test set (Pinter et al., 2017). We then augment the *hidden-state + char n-grams* method with it.

[2]In their work, the authors report that initializing embeddings randomly rather than with GLoVe yields state of the art performance on the Twitter dataset that we are using. However, we found the opposite when performing 10-fold stratified cross-validation (CV). A possible explanation of this lies in the authors' decision to not use stratification, which for such a highly imbalanced dataset can lead to unexpected outcomes (Forman and Scholz, 2010). Furthermore, the authors train their LSTM on the entire dataset including the test part without any early stopping criterion; this facilitates over-fitting of the randomly-initialized embeddings.

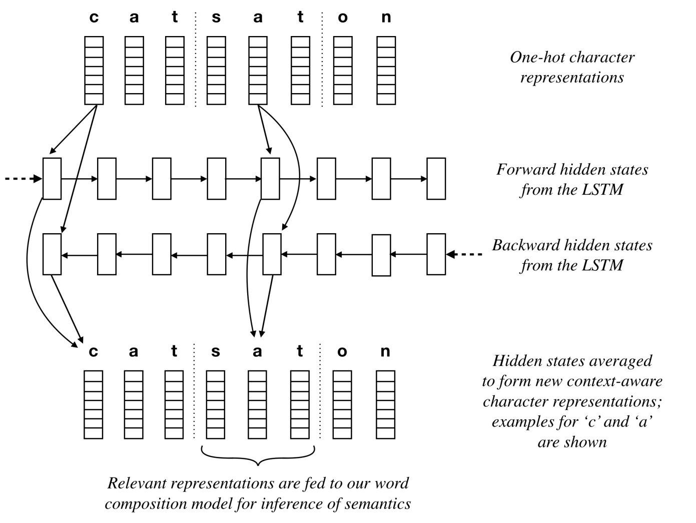
[3]The deeper 2-layer GRU slightly improves performance.

[4]$L_2$-normalized sum ensures uniformity of range across the feature set in all our methods; GBDT, being a tree based model, is not affected by the choice of monotonic function.

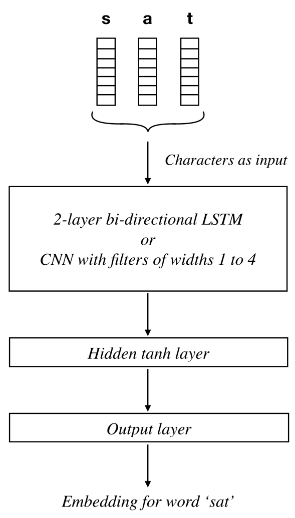
Specifically, our model (Figure 1b) comprises a 2-layer bi-directional LSTM, followed by a hidden layer with *tanh* non-linearity and an output layer at the end. The model takes as input a sequence $c_1, \ldots, c_k$ of characters, represented as one-hot vectors, from a fixed vocabulary (i.e., lowercase English alphabet and digits) and outputs a $d$-dimensional embedding for the word '$c_1 \ldots c_k$'. Bi-directionality of the LSTM allows for the semantics of both the prefix and the suffix (last hidden forward and backward state) of the input word to be captured, which are then combined to form the hidden state for the input word. The model is trained by minimizing the mean squared error (MSE) between the embeddings that it produces and the task-tuned embeddings of words in the training set. This ensures that newly composed embeddings are endowed with characteristics from both the GLoVe space as well as the task-tuning process. While approaches like that of Bojanowski et al. (2017) can also compose embeddings for unseen words, they cannot endow the newly composed embeddings with characteristics from the task-tuning process; this may constitute a significant drawback (Kim, 2014).

During the training of our character-based word composition model, to emphasize frequent words, we feed a word as many times as it appears in the training corpus. We note that a 1-layer CNN with global max-pooling in place of the 2-layer LSTM provides comparable performance while requiring significantly less time to train. This is expected since words are not very long sequences, and the filters of the CNN are able to capture the different character n-grams within them.

**Context hidden-state + char n-grams (CONTEXT HS + CNG).** In the *augmented hidden-state + char n-grams* method, the word composition model infers semantics of unseen words solely on the basis of the characters in them. However, for many words, semantic inference and sense disambiguation require context, i.e., knowledge of character sequences in the vicinity. An example is the word *cnt* that has different meanings in the sentences "*I cnt undrstand this!*" and "*You feminist cnt!*", i.e., *cannot* in the former and the sexist slur *cunt* in the latter. Yet another example is an obfuscation like ''*You mot otherf uc ker!* where the expletive *motherfucker* cannot be properly inferred from any fragment without the knowledge of surrounding character sequences.


c a t s a t o n
One-hot character representations
Forward hidden states from the LSTM
Backward hidden states from the LSTM
c a t s a t o n
Hidden states averaged to form new context-aware character representations; examples for 'c' and 'a' are shown
Relevant representations are fed to our word composition model for inference of semantics

(a)



(b)

Figure 1: Context-aware approach to word composition. The figure on the left shows how the encoder extracts context-aware representations of characters in the phrase "*cat sat on*" from their one-hot representations. The dotted lines denote the space character ␣ which demarcates word boundaries. Semantics of an unseen word, e.g., *sat*, can then be inferred by our word composition model shown on the right.

To address this, we develop *context-aware representations* for characters as inputs to our character-based word composition model instead of one-hot representations.[5] We introduce an encoder architecture to produce the context-aware representations. Specifically, given a text formed of a sequence $w_1, \ldots, w_n$ of words, the encoder takes as input one-hot representations of the characters $c_1, \ldots, c_k$ within the concatenated sequence '$w_1 ␣ \ldots ␣ w_n$', where ␣ denotes the space character. This input is passed through a bi-directional LSTM that produces hidden states $h_1, \ldots, h_k$, one for every character. Each hidden state, referred to as context-aware character representation, is the average of its designated forward and backward states; hence, it captures both the preceding as well as the following contexts of the character it corresponds to. Figure 1 illustrates how the context-aware representations are extracted and used for inference by our character-based word composition model. The model is trained in the same manner as done in the *augmented hidden-state + char n-grams* method, i.e., by minimizing the MSE between the embeddings that it produces and the task-tuned embeddings of words in the training set (initialized with GLoVe). However, the inputs now are context-aware representations of characters instead of one-hot representations.

**Word-sum + char n-grams (WS + CNG), Augmented word-sum + char n-grams (AUGMENTED WS + CNG),** and **Context word-sum + char n-grams (CONTEXT WS + CNG).** These methods are identical to the *(context/augmented) hidden-state + char n-grams* methods except that here we include the character n-grams and our character-based word composition model on top of the *word-sum* baseline.

**Char hidden-state (CHAR HS)** and **Char word-sum (CHAR WS).** In all the methods described up till now, the input to the core RNN is word embeddings. To gauge whether character-level inputs are themselves sufficient or not, we construct two methods based on the *character to word* (C2W) approach of Ling et al. (2015). For the *char hidden-state* method, the input is one-hot representations of characters from a fixed vocabulary. These representations are encoded into a sequence $w_1, \ldots, w_n$ of intermediate word embeddings by a 2-layer bi-directional LSTM. The word embeddings are then fed into a 2-layer GRU that transforms them into hidden states $h_1, \ldots, h_n$. Finally, as in the *hidden-state* baseline, an LR layer with softmax activation uses the last hidden state $h_n$ to perform classification while propagating error backwards to train the network. The *char word-sum* method

[5]We also experimented with word-level context but did not get any significant improvements. We believe this is due to higher variance at word level than at the character level.

is similar except that once the network has been trained, we use the intermediate word embeddings produced by it to train a GBDT classifier in the same manner as done in the *word-sum* baseline.

## 5 Experiments and Results

### 5.1 Experimental setup

We normalize the input by lowercasing all words and removing stop words. For the GRU architecture, we use exactly the same hyper-parameters as Pavlopoulos et al. (2017),[6] i.e., 128 hidden units, Glorot initialization, cross-entropy loss, and Adam optimizer (Kingma and Ba, 2015). Badjatiya et al. (2017) also use the same settings except they have fewer hidden units. The LSTM in our character-based word composition model has 256 hidden units while that in our encoder has 64; the CNN has filters of widths varying from 1 to 4. The results we report are with an LSTM-based word composition model. In all the models, besides dropout regularization (Srivastava et al., 2014), we hold out a small part of the training set as validation data to prevent over-fitting. We use 300d embeddings and 1 to 5 character n-grams for Wikipedia and 200d embeddings and 1 to 4 character n-grams for Twitter. We implement the models in *Keras* (Chollet et al., 2015) with *Theano* back-end. We employ *Lightgbm* (Ke et al., 2017) as our GDBT classifier and tune its hyper-parameters using 5-fold grid search.

### 5.2 Twitter results

For the Twitter dataset, unlike previous research (Badjatiya et al., 2017; Park and Fung, 2017), we report the macro precision, recall, and $F_1$ averaged over 10 folds of stratified CV (Table 1). For a classification problem with $N$ classes, macro precision (similarly, macro recall and macro $F_1$) is given by:

$$Macro\ P = \frac{1}{N}\sum_{i=1}^{N} P_i$$

where $P_i$ denotes precision on class $i$. Macro metrics provide a better sense of effectiveness on the minority classes (Van Asch, 2013).

We observe that character n-grams (CNG) consistently enhance performance, while our augmented approach (AUGMENTED) further improves upon the results obtained with character n-grams. All the improvements are statistically significant with $p < 0.05$ under 10-fold CV paired t-test.

As Ling et al. (2015) noted in their POS tagging experiments, we observe that the CHAR HS and CHAR WS methods perform worse than their counterparts that use pre-trained word embeddings, i.e., the HS and WS baselines respectively.

To further analyze the performance of our best methods (CONTEXT/AUGMENTED WS/HS + CNG), we also examine the results on the racism and sexism classes individually (Table 2). As before, we see that our approach consistently improves over the baselines, and the improvements are statistically significant under paired t-tests.

| Method | P | R | $F_1$ |
|---|---|---|---|
| HS | 79.14 | 77.06 | 78.01 |
| CHAR HS | 79.48 | 69.00 | 72.36 |
| HS + CNG$^\dagger$ | 80.36 | **78.20** | 79.19 |
| AUGMENTED HS + CNG$^\dagger$ | 81.28 | 77.84 | **79.37** |
| CONTEXT HS + CNG$^\dagger$ | **81.39** | 77.47 | 79.21 |
| WS | 80.78 | 72.83 | 75.93 |
| CHAR WS | 80.04 | 68.17 | 71.94 |
| WS + CNG$^\dagger$ | 83.16 | 76.60 | 79.31 |
| AUGMENTED WS + CNG$^\dagger$ | **83.50** | **77.20** | **79.80** |
| CONTEXT WS + CNG$^\dagger$ | 83.44 | 77.06 | 79.67 |

Table 1: Results on the Twitter dataset. The methods we propose are denoted by $^\dagger$. Our best method (AUGMENTED WS + CNG) significantly outperforms all other methods.

| Method | P | R | $F_1$ |
|---|---|---|---|
| HS | 74.15 | 72.46 | 73.24 |
| AUGMENTED HS + CNG$^\dagger$ | 76.28 | **72.72** | **74.40** |
| CONTEXT HS + CNG$^\dagger$ | **76.61** | 72.15 | 74.26 |
| WS | 76.43 | 67.77 | 71.78 |
| AUGMENTED WS + CNG$^\dagger$ | **78.17** | 72.20 | **75.01** |
| CONTEXT WS + CNG$^\dagger$ | 77.90 | **72.26** | 74.91 |

(a) Racism

| Method | P | R | $F_1$ |
|---|---|---|---|
| HS | 76.04 | 68.84 | 72.24 |
| AUGMENTED HS + CNG$^\dagger$ | 80.07 | **69.28** | **74.26** |
| CONTEXT HS + CNG$^\dagger$ | **80.29** | 68.52 | 73.92 |
| WS | 81.75 | 57.37 | 67.38 |
| AUGMENTED WS + CNG$^\dagger$ | 85.61 | **65.91** | **74.44** |
| CONTEXT WS + CNG$^\dagger$ | **85.80** | 65.41 | 74.18 |

(b) Sexism

Table 2: The baselines (WS, HS) vs. our best approaches ($^\dagger$) on the racism and sexism classes.

Additionally, we note that the AUGMENTED WS + CNG method improves the $F_1$ score of the WS

[6]The authors have not released their models; we replicate their method based on the details in their paper.

+ CNG method from 74.12 to 75.01 for the racism class, and from 74.03 to 74.44 for the sexism class. The AUGMENTED HS + CNG method similarly improves the $F_1$ score of the HS + CNG method from 74.00 to 74.40 on the racism class while making no notable difference on the sexism class.

We see that the CONTEXT HS/WS + CNG methods do not perform as well as the AUGMENTED HS/WS + CNG methods. One reason for this is that the Twitter dataset is not able to expose the methods to enough contexts due to its small size. Moreover, because the collection of this dataset was bootstrapped with a search for certain commonly-used abusive words, many such words are shared across multiple tweets belonging to different classes. Given the above, context-aware character representations perhaps do not provide substantial distinctive information.

## 5.3 Wikipedia results

Following previous work (Pavlopoulos et al., 2017; Wulczyn et al., 2017), we conduct a standard 60:40 train–test split experiment on the two Wikipedia datasets. Specifically, from W-TOX, $95,692$ comments ($10.0\%$ abusive) are used for training and $63,994$ ($9.1\%$ abusive) for testing; from W-ATT, $70,000$ ($11.8\%$ abusive) are used for training and $45,854$ ($11.7\%$ abusive) for testing. Table 3 reports the macro $F_1$ scores. We do not report scores from the CHAR HS and CHAR WS methods since they showed poor preliminary results compared to the HS and WS baselines.

| Method | W-TOX | W-ATT |
|---|---|---|
| HS | 88.65 | 86.28 |
| HS + CNG$^\dagger$ | 89.29 | 87.32 |
| AUGMENTED HS + CNG$^\dagger$ | 89.31 | 87.33 |
| CONTEXT HS + CNG$^\dagger$ | **89.35** | **87.44** |
| WS | 85.49 | 84.35 |
| WS + CNG$^\dagger$ | 87.12 | 85.80 |
| AUGMENTED WS + CNG$^\dagger$ | 87.02 | 85.75 |
| CONTEXT WS + CNG$^\dagger$ | **87.16** | **85.81** |

Table 3: Macro $F_1$ scores on the two Wikipedia datasets. The current state of the art method for these datasets is HS. $^\dagger$ denotes the methods we propose. Our best method (CONTEXT HS + CNG) outperforms all the other methods.

Mirroring the analysis carried out for the Twitter dataset, Table 4 further compares the performance of our best methods for Wikipedia (CONTEXT/AUGMENTED HS + CNG) with that of the state of the art baseline (HS) on specifically the abusive classes of W-TOX and W-ATT.

| Method | P | R | $F_1$ |
|---|---|---|---|
| HS | 84.48 | 74.60 | 79.24 |
| AUGMENTED HS + CNG$^\dagger$ | **85.43** | 76.02 | 80.45 |
| CONTEXT HS + CNG$^\dagger$ | 85.42 | **76.17** | **80.53** |

(a) W-TOX

| Method | P | R | $F_1$ |
|---|---|---|---|
| HS | 78.61 | 72.88 | 75.64 |
| AUGMENTED HS + CNG$^\dagger$ | 81.23 | 74.06 | 77.48 |
| CONTEXT HS + CNG$^\dagger$ | **81.39** | **74.28** | **77.67** |

(b) W-ATT

Table 4: The current state of the art baseline (HS) vs. our best methods ($^\dagger$) on the abusive classes of W-TOX and W-ATT.

We observe that the augmented approach substantially improves over the state of the art baseline. Unlike in the case of Twitter, our context-aware setup for word composition is now able to further enhance performance courtesy of the larger size of the datasets which increases the availability of contexts. All improvements are significant ($p < 0.05$) under paired t-tests. We note, however, that the gains we get here with the word composition model are relatively small compared to those we get for Twitter. This difference can be explained by the fact that: (i) Wikipedia comments are less noisy than the tweets and contain fewer obfuscations, and (ii) the Wikipedia datasets, being much larger, expose the methods to more words during training, hence reducing the likelihood of unseen words being important to the semantics of the comments they belong to (Kim et al., 2016).

Like Pavlopoulos et al. (2017), we see that the methods that involve summation of word embeddings (WS) perform significantly worse on the Wikipedia datasets compared to those that use hidden state (HS); however, their performance is comparable or even superior on the Twitter dataset. This contrast is best explained by the observation of Nobata et al. (2016) that taking average or sum of word embeddings compromises contextual and word order information. While this is beneficial in the case of tweets which are short and loosely-structured, it leads to poor performance of the WS and WS + CNG methods on the Wikipedia datasets, with the addition of the word composition model (CONTEXT/AUGMENTED WS + CNG) providing little to no improvements.

| Abusive sample | Predicted class | | |
|---|---|---|---|
| | WS | WS + CNG | AUGMENTED WS + CNG |
| *@mention I love how the Islamofascists recruit 14 and 15 year old jihadis and then talk about minors in reference to 17 year olds.* | *neither* | *racism* | *racism* |
| *@mention @mention @mention As a certified inmate of the Islamasylum, you don't have the ability to judge.* | *neither* | *racism* | *racism* |
| *@mention "I'll be ready in 5 minutes" from a girl usually means "I'll be ready in 20+ minutes." #notsexist #knownfrom-experience* | *neither* | *sexism* | *sexism* |
| *RT @mention: #isis #muslim #Islamophobia? I think the word you're searching for is #Islamorealism http://t.co/NyihT8Bqyu http://t.c...* | *neither* | *neither* | *racism* |
| *@mention @mention And looking at your page, I can see that you are in the business of photoshopping images, Islamist cock-sucker.* | *neither* | *neither* | *racism* |
| *I think Kat is on the wrong show. #mkr is for people who can cook. #stupidbitch #hopeyouareeliminated* | *neither* | *neither* | *sexism* |
| *@mention because w0m3n are a5sh0les #feminismishate* | *neither* | *neither* | *sexism* |

Table 5: Improved classification upon the addition of character n-grams (CNG) and our word composition model (AUGMENTED). Names of users have been replaced with *mention* for anonymity.

## 6 Discussion

To investigate the extent to which obfuscated words can be a problem, we extract a number of statistics. Specifically, we notice that out of the approximately $16k$ unique tokens present in the Twitter dataset, there are about $5.2k$ tokens that we cannot find in the English dictionary.[7] Around 600 of these $5.2k$ tokens are present in the racist tweets, $1.6k$ in the sexist tweets, and the rest in tweets that are neither. Examples from the racist tweets include *fuckbag*, *ezidiz*, *islamofascists*, *islamistheproblem*, *islamasylum* and *isisaremuslims*, while those from the sexist tweets include *c*nt*, *bbbbitch*, *feminismisawful*, and *stupidbitch*. Given that the racist and sexist tweets come from a small number of unique users, 5 and 527 respectively, we believe that the presence of obfuscated words would be even more pronounced if tweets were procured from more unique users.

In the case of the Wikipedia datasets, around $15k$ unique tokens in the abusive comments of both W-TOX and W-ATT are not attested in the English dictionary. Examples of such tokens from W-TOX include *fuggin*, *n*gga*, *fuycker*, and *1d10t*; and from W-ATT include *f**king*, *beeeitch*, *musulmans*, and *motherfucken*. In comparison to the tweets, the Wikipedia comments use more "standard" language. This is validated by the fact that only 14% of the tokens present in W-TOX and W-ATT are absent from the English dictionary as opposed to 32% of the tokens in the Twitter dataset even though the Wikipedia ones are almost ten times larger.

Across the three datasets, we note that the addition of character n-gram features enhances the performance of RNN-based methods, corroborating the previous findings that they capture complementary structural and lexical information of words. The inclusion of our character-based word composition model yields state of the art results on all the datasets, demonstrating the benefits of inferring the semantics of unseen words. Table 5 shows some abusive samples from Twitter that are misclassified by the WS baseline method but are correctly classified upon the addition of character n-grams (WS + CNG) and the further addition of our character-based word composition model (AUGMENTED WS + CNG).

Many of the abusive tweets that remain misclassified by the AUGMENTED WS + CNG method are those that are part of some abusive discourse (e.g., *@Mich_McConnell Just "her body" right?*) or contain URLs to abusive content (e.g., *@salmonfarmer1: Logic in the world of Islam http://t.co/6nALv2HPc3*).

In the case of the Wikipedia datasets, there are abusive examples like *smyou have a message re your last change, go fuckyourself!!!* and *F-uc-k you, a-ss-hole Motherf–ucker!* that are misclassified by the state of the art HS baseline and the HS + CNG method but correctly classified by our best method for the datasets, i.e., CONTEXT HS + CNG.

[7] We use the US English spell-checking utility provided by the *PyEnchant* library of python.

| Word | Similar words in training set |
|---|---|
| *women* | *girls*, *woman*, *females*, *chicks*, *ladies* |
| *w0m3n*[†] | *woman*, *women*, *girls*, *ladies*, *chicks* |
| *cunt* | *twat*, *prick*, *faggot*, *slut*, *asshole* |
| *a5sh0les*[†] | *assholes*, *stupid*, *cunts*, *twats*, *faggots* |
| *stupidbitch*[†] | *idiotic*, *stupid*, *dumb*, *ugly*, *women* |
| *jihad* | *islam*, *muslims*, *sharia*, *terrorist*, *jihadi* |
| *jihaadi*[†] | *terrorists*, *islamist*, *jihadists*, *muslims* |
| *terroristislam*[†] | *terrorists*, *muslims*, *attacks*, *extremists* |
| *fuckyouass*[†] | *fuck*, *shit*, *fucking*, *damn*, *hell* |

Table 6: Words in the training set that exhibit high cosine similarity to the given word. The ones marked with † are not seen during training; embeddings for them are composed using our word composition model.

To ascertain the effectiveness of our task-tuning process for embeddings, we conducted a qualitative analysis, validating that semantically similar words cluster together in the embedding space. Analogously, we assessed the merits of our word composition model by verifying the neighbors of embeddings formed by it for obfuscated words not seen during training. Table 6 provides some examples. We see that our model correctly infers the semantics of obfuscated words, even in cases where obfuscation is by concatenation of words.

## 7 Conclusions

In this paper, we considered the problem of obfuscated words in the field of automated abuse detection. Working with three datasets from two different domains, namely Twitter and Wikipedia talk page, we first comprehensively replicated the previous state of the art RNN methods for the datasets. We then showed that character n-grams capture complementary information, and hence, are able to enhance the performance of the RNNs. Finally, we constructed a character-based word composition model in order to infer semantics for unseen words and further extended it with context-aware character representations. The integration of our composition model with the enhanced RNN methods yielded the best results on all three datasets. We have experimentally demonstrated that our approach to modeling obfuscated words significantly advances the state of the art in abuse detection. In the future, we wish to explore its efficacy in tasks such as grammatical error detection and correction. We will make our models and logs of experiments publicly available at `https://github.com/pushkarmishra/AbuseDetection`.

## Acknowledgements

Special thanks to the anonymous reviewers for their valuable comments and suggestions.

## References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

François Chollet et al. 2015. Keras.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 29–30, New York, NY, USA. ACM.

Maeve Duggan. 2014. Online harassment.

George Forman and Martin Scholz. 2010. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.*, 12(1):49–57.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3149–3157. Curran Associates, Inc.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '15.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, ICML '14.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530. Association for Computational Linguistics.

Pranava Swaroop Madhyastha, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Mapping unseen words to task-trained embedding spaces. *CoRR*, abs/1510.02387.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2018. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098. Association for Computational Linguistics.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45. Association for Computational Linguistics.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112. Association for Computational Linguistics.

J. Qian, M. ElSherief, E. Belding, and W. Wang. 2018. Leveraging intra-user and inter-user representation learning for automated hate speech detection. *NAACL HLT, New Orleans, LA, June 2018.*, page *to appear*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Vincent Van Asch. 2013. Macro-and micro-averaged evaluation measures [[basic draft]]. *Computational Linguistics & Psycholinguistics, University of Antwerp, Belgium.*

Zeerak Waseem. 2016. Are you a racist or am I seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142. Association for Computational Linguistics.

Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault. 2017. Proceedings of the first workshop on abusive language online. In *Proceedings of the First Workshop on Abusive Language Online*. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1391–1399, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Dawei Yin, Brian D. Davison, Zhenzhen Xue, Liangjie Hong, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on web 2.0. In *Processings of the Content Analysis in the WEB 2.0*, 2:1-7.