Neural Word Decomposition Models for Abusive Language Detection

Sravan Babu Bodapati   **Spandana Gella**
Kasturi Bhattacharjee   **Yaser Al-Onaizan**
Amazon AI, USA
{sravanb, sgella, kastb, onaizan}@amazon.com

Abstract

User generated text on social media often suffers from a lot of undesired characteristics including hatespeech, abusive language, insults etc. that are targeted to attack or abuse a specific group of people. Often such text is written differently compared to traditional text such as news involving either explicit mention of abusive words, obfuscated words and typological errors or implicit abuse i.e., indicating or targeting negative stereotypes. Thus, processing this text poses several robustness challenges when we apply natural language processing techniques developed for traditional text. For example, using word or token based models to process such text can treat two spelling variants of a word as two different words. Following recent work, we analyze how character, subword and byte pair encoding (BPE) models can be aid some of the challenges posed by user generated text. In our work, we analyze the effectiveness of each of the above techniques, compare and contrast various word decomposition techniques when used in combination with others. We experiment with finetuning large pretrained language models, and demonstrate their robustness to domain shift by studying Wikipedia attack, toxicity and Twitter hatespeech datasets.

1 Introduction

In recent years, with the growing popularity of social media applications, there has been an exponential increase in the amount of user-generated text including microblog posts, status updates and comments posted on the web. The power of communicating freely with large number of users has resulted in not only sharing news and exchanging content but has also led to a problem of large number of harmful, offensive and aggressive interactions online (Duggan, 2017). Previous work on identifying abusive language has tackled this problem by training computational methods that are capable of automatically recognizing offensive content for text on MySpace (Yin et al., 2009), Twitter (Waseem and Hovy, 2016; Davidson et al., 2017), Wikipedia comments (Wulczyn et al., 2017) and Facebook posts (Vigna et al., 2017; Kumar et al., 2018).

Most of these models are based on features extracted from words or word n-grams or the recurrent neural networks that operate on word embeddings (Pavlopoulos et al., 2017; Badjatiya et al., 2017) with few exceptions of models that utilize character n-grams that can model noisy text and out-of-vocabulary words (Waseem and Hovy, 2016; Nobata et al., 2016; Wulczyn et al., 2017). However, these models are not very effective at modeling obfuscated words such as *w0m3n, nlgg3r* which are prominent in user generated text that are intended at evading hate speech detection (Mishra et al., 2018). In this work, we aim to address this by investigating word, subword and character-based models for abusive language detection.

Recent advances in unsupervised pre-training of language models have led to strong improvements on various general natural language processing and understanding tasks such as question answering, sentiment and natural language inference (Peters et al., 2018; Devlin et al., 2018). However, it is unclear how such models trained on standard text would transfer information when fine-tuned on noisy user generated text. In additional to studying word, subword and character-based model performances on abusive language detection we also combine these with pre-trained embeddings and fine-tuning these pre-trained language models and understand their efficiency and robustness in identifying abusive text.

Specifically, in this work, we address the effectiveness of character-based models, subword or Byte Pair Encoding (BPE) based models and word

features based models along with pre-trained word embeddings and fine tuning pretrained language models for detecting abusive language in text. Precisely we make following contributions:

- We compare the effectiveness of end-to-end character based models, with word + character embedding models, byte pair encoding and subword models, to show which of the techniques perform better than pure word based models.

- We demonstrate how fine-tuning large pre-trained language models, the latest breakthrough in NLP, enhance state of the art on few of the abusive language datasets, and show that the domain shift isn't considerable when applied to abusive language datasets.

- We also examine how preprocessing documents with byte pair encoding model pretrained on a large corpus, boost the performance of several word embedding based models massively.

2 Related Work

Identifying abusive context on the web is one of the widely studied topics on social media text. This problem has been studied for Hate Speech detection (Kwok and Wang, 2013; Waseem and Hovy, 2016; Waseem, 2016; Ross et al., 2016; Saleem et al., 2017; Warner and Hirschberg, 2012), Harassment (Yin et al., 2009; Cheng et al., 2015), Cyberbullying (Willard, 2007; Tokunaga, 2010; Schrock and Boyd, 2011), Abusive language detection (Sahlgren et al., 2018; Nobata et al., 2016), aggression identification (Kumar et al., 2018; Aroyehun and Gelbukh, 2018; Modha et al., 2018), identifying toxic comments on forums (Wulczyn et al., 2017) and offensive language identification (Wiegand et al., 2018; Zampieri et al., 2019). While most of the work in identifying abusive on social media is predominantly studied for English social media posts some of the latest work include study on German (Wiegand et al., 2018), Italian (Bosco et al., 2018) and Mexican Spanish (Álvarez-Carmona et al., 2018).

Some of the early methods on identifying abusive text used word n-gram, part-of-speech (POS) tagging (syntactic features), manually created profanity lexicons or stereotypical words, TF-IDF features along with sentiment and contextual features

and trained supervised classifiers such as support vector machines. (Yin et al., 2009; Warner and Hirschberg, 2012). Waseem (2016) studied character n-grams, skipgrams, brown clusters and POS tag based features for identifying hatespeech. Waseem and Hovy (2016) studied usefulness of various socio-linguistic features such as gender, location, word-length distribution, Author Historical Salient Terms (AHST) features in identifying hatespeech.

Some of the recent work compared efficiency of both character n-gram based models as inputs to logistic regression and multi-layer perceptron models (Wulczyn et al., 2017). Nobata et al. (2016) showed that character n-grams features alone can perform well and can efficiently model noisy text. They also showed off-the-shelf word embeddings can be used to identify abusive text.

Pavlopoulos et al. (2017) used deep-learning based models specifically they employed RNN with a novel classification-specific attention mechanism and achieve state-of-the-art results on identifying attack and toxic content in Wikipedia comments. Badjatiya et al. (2017) investigated three different neural networks for hatespeech detection: (i) Convolutional neural network (inspired by CNN's for sentiment classification by Kim (2014)) (ii) Long short-term memory networks (LSTM) to capture long range dependencies and (iii) FastText classification model that represents document by averaging word vectors that can be fine-tuned for the hatespeech task.

While Badjatiya et al. (2017) analyzed various architectures to encode text for hatespeech detection, we are not aware of any work that studied various word decomposition models for identifying abusive language in text. Recent work on identifying offensive language in text include fine-tuning large pretrained language model BERT which use subword units to encode text (Zampieri et al., 2019; Zhu et al., 2019). For the SEMEVAL-2019 task of offensive language identification 7 out of top 10 submissions used BERT finet tuning. Zampieri et al. (2019) highlighted that 8% of 104 systems participated in the shared task used BERT based fine-tuning.

In this work, we analyze the effectiveness of different ways of learning representations with character-based models, subword or BPE based models and word features based models. We also combine these with well known pre-trained word embeddings and very large pretrained language

models for fine-tuning and detecting abusive language in text. In Section 3 we describe the datasets that we study in this work for hatespeech and abusive detection.

3 Datasets

We experiment with three datasets: Twitter dataset (Waseem and Hovy, 2016), Personal Attack and Toxicity datasets from Wikipedia Talk dataset (Wulczyn et al., 2017) that covers sexism/racism, personal attack and toxicity aspects of abuse in user generated text online.

3.1 Twitter Dataset

We use the hatespeech Twitter dataset (Hatespeech) provided by Waseem and Hovy (2016). This dataset was created from a corpus of 136k tweets collected from Twitter by searching for commonly used racist and sexist slurs on various ethnic, gender and religious minorities over a two-month period. The original data had 16,907 tweets corresponding to sexist, racist and neither labels (3378, 1970 and 11559 respectively). However, we could only retrieve 11170 of the tweets (2914: sexism, 17: racism and 8239: neither) with python's Tweepy library. Similar issue of missing tweets has been reported by Mishra et al. (2018). However, the percent of tweets we lost are much higher than theirs and most of the tweets lost are for the *racism* label. We have lost majority of the tweets corresponding to sexism label. Since we lost large chunk of tweets we conduct our experiments on cross validation of 5 splits and report scores on all of the 5 splits.

3.2 Wikipedia talk page

We use the personal attacks (W-ATT) and toxicity (W-TOX) datasets that were randomly sampled from 63 Million talk page comments from the public dump of English Wikipedia by Wulczyn et al. (2017). Each comment in both the datasets were annotated by at least 10 workers and we use the majority label as its gold label. Overall, we have 115.8k comments in W-ATT dataset (69.5k, 23.1k and 23.1k in train, dev and test splits respectively) and 159.6k comments in W-TOX dataset (95.6k, 32.1k and 31.8k in train, dev and test splits). Similar to hatespeech dataset both the W-ATT and W-TOX datasets also have skewed distribution of labels having 13.5% and 15.3% of them labeled as abusive.

4 Methods

In this section, we present various word decomposition methods and modeling architectures we analysed for studying Twitter and Wiki Talk page W-ATT and W-TOX comment datasets.

4.1 Word-based Model

As a baseline, we adpot the fastText (Grave et al., 2017) classification algorithm. The fastText algorithm performs mean pooling on top of the word embeddings w_i^{emb} to obtain a document representation. This document representation is passed through a Softmax layer to obtain classification scores. The embeddings can either be learned or can be initialized with pre-trained embeddings and fine-tuned during training. We run multiple variants of fastText in our experiments.

4.2 Subword-based Model

Subwords are formed by concatenating all the characters of a particular length within a word boundary. Addition of subwords gives the model ability to learn words which are misspelled such as *emnlp* and *emnnlp* are similar. A pure word based model would consider *emnlp* as out-of-vocabulary (OOV) word, if not seen in training set, but a subword model would decompose *emnlp* into "emn" and "nlp", and train subword embeddings w_{sub}^{emb} for each of these subwords. We take subword variant of fastText model to incorporate subword context into the model. The algorithm considers all subwords of varying lengths within the boundary of a word.

4.3 Joint Word and Character Embedding Model

Our joint word and character embedding based model is adapted from Kim (2014) and Peters et al. (2018). We refer to Kim (2014) as TextCNN going forward.

Let x_i be the input word and c_0^n be its character representation, where n is the number of characters in the word. We transform c_0^n representation by passing through a character embedding layer, which is a n-gram Character-CNN similar to (Peters et al., 2019). The output of the n-gram CharacterCNN is concatenated with the word's corresponding pretrained embedding to obtain w_{full}^{emb} as described in 1(a) Character-level features are concatenated with w_i^{emb}, the word embedding of word i, to form w_{full}^{emb}, the full set of word-level input
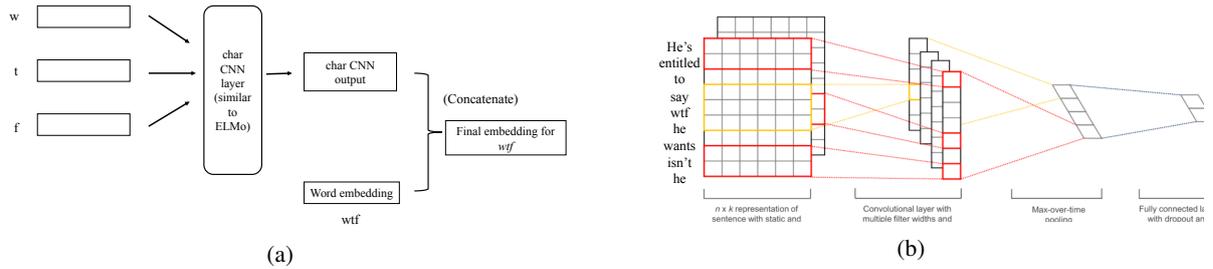
Figure 1: Architecture for model described in 4.3. In Figure 1(a), we present an example of for obtaining a word embedding by concatenating character embeddings with the embedding of the word itself. These final embeddings are then fed into the non-static variant of the Kim2014 (Kim, 2014) architecture (shown in Figure 1(b)). The layers of Kim2014 model alongwith the character CNN layer are updated during training.

features:

$$w_{full}^{emb} = (w_i^{char}; w_i^{word})$$

We randomly replace singleton words with special [UNK] (unknown) tokens for obtaining its w_i^{emb}, and also apply dropout (Srivastava et al., 2014) on w_{full}^{emb}. The input word embeddings w_{full}^{emb}, in a sentence with l tokens and convolutional window size h, $w_{i:i+h}^{emb}$ is transformed through a convolution filter w_c:

$$c_i = f(w_c.w_{i:i+h-1}^{emb} + b_c)$$

where b_c is a bias term and f is a non-linear function (ReLU). This produces a feature map c, on-top of which we apply a global max-over time pooling.

$$c' = max(\mathbf{c})$$

This process for one feature is repeated to obtain m filters with different window sizes h. The resulting filters are concatenated to form TextCNN document representation. The document representation is passed through Softmax layer to obtain classification predictions. We also experiment with original version of TextCNN, which is a pure word based model, without the character embedding variant.

4.4 End-to-end Character Embedding Model

To understand the potential of end to end character based models in dealing noisy text, we use Very Deep Convolutional Neural Network (VDCNN) architecture proposed by Conneau et al. (2017) that operates at character level by stacking multiple convolutional and pooling layers that sequentially extract a hierarchical representation of the text. This representation is fed into a fully connceted layer which is trained to maximize the classification accuracy on training data.

4.5 Byte Pair Encoding + Word + Char embedding models

We train a Byte Pair Encoding(BPE) based model introduced by Sennrich et al. (2016) on the given training corpus. We use this trained BPE model on training data to tokenize/encode our documents in training, validation and test data and use each BPE unit as a word to learn embeddings. We perform $30,000$ merge operations on each training dataset to learn subword or BPE units.

4.6 Pretrained Language Models

Recent liteature have shown that transferring knowledge from large pre-trained language models could benefit various NLP tasks either by adding a task specific architecture or by fine-tuning the language model for the end task (Peters et al., 2018; Devlin et al., 2018; Peters et al., 2019). In this work, we use $BERT$ model and we fine-tune the model for each of our train datasets.

5 Experiments

In this section, we present different variants of the models described in Section 4 presented in Table 1.

fastText: We use multiple variants of fastText model. Our fastText$_{ngrams=1}$ uses embeddings learned for each unigram. We treat this as our baseline model without any preprocessing of the text. Our fastText$_{ngrams=2}$ model also uses bigrams along with unigrams as independent tokens to learn embeddings. All pairs of bigrams are chosen wtihout ant frequency threshold. Our fastText$_{ngrams=2}$ + subword $(2-6)$ also uses all subwords within a word boundary within the range of $2-6$. All our models are trained with learning rate of 0.5 and for 5 epochs.
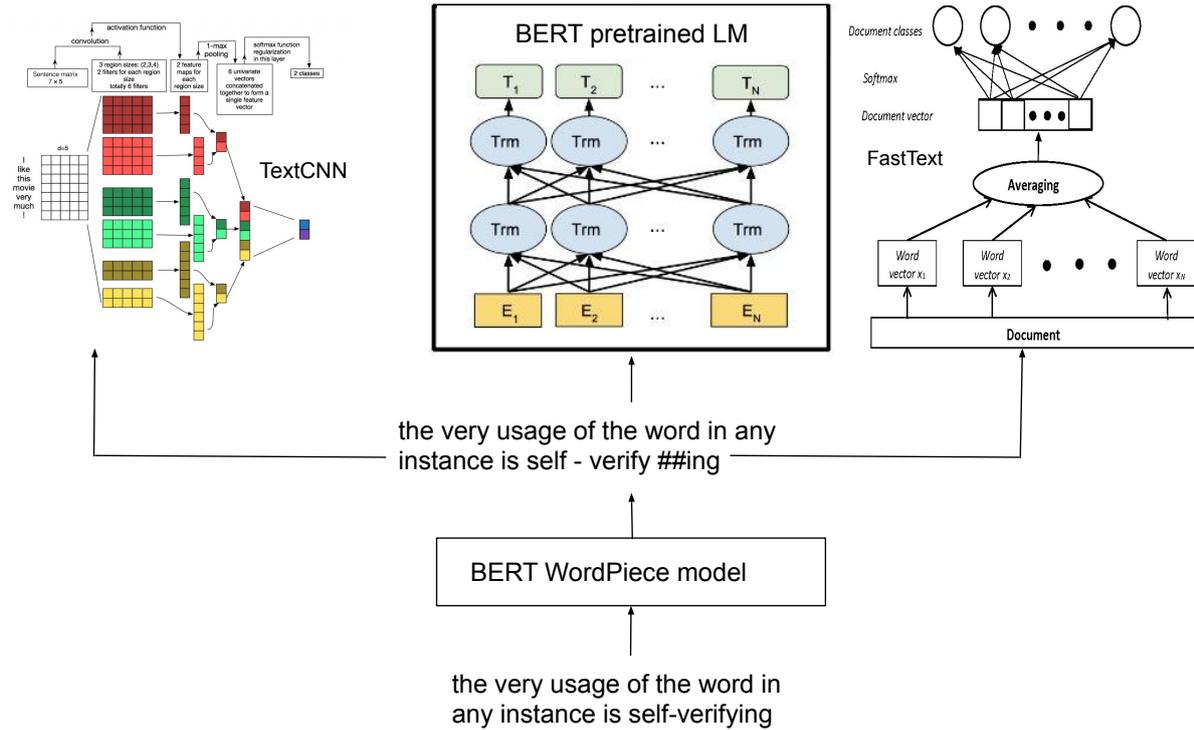
Figure 2: We present the approach discussed in 5.1. The Input text for a document is tokenized via the BERT Wordpiece tokenized model pretrained on GoogleNews and Wikipedia. This tokenized text is fed as input to the word based models which aids in forming representations from a more informative subword split as an independent unit.

TextCNN (Kim, 2014): We run the TextCNN for classification in non-static mode, with learning rate of 0.0001, dropout of 0.5 for 50 epochs. We have used default kernel window sizes $N_f = (3, 4, 5)$ with $m = 100$ filters. We initialize the embeddings layer with word2vec pretrained embeddings[1] publicly available from google. We used the non-static variant of TextCNN, with pretrained embedding initialization for word embedding layer.

TextCNN + char n-grams: The word embedding layer is constructed for this approach as mentioned in 1(b). The kernel window sizes h for character tokens are $N_f = (1, 2, 3, 4, 5, 6)$ with $m = (32, 32, 32, 64, 64, 64)$ filters respectively. Increasing the number of filters further to match those of parameters in Peters et al. (2018) for character tokens led to overfitting on our datasets, and hence we reduced the parameters. All the layers are allowed to be tuned while training. The character embeddings CNN layer is initialized randomly with Xavier initialization (Glorot and Bengio, 2010). We set the character embedding layer output to 300, upon concatenation the word embedding w_{full}^{emb}

---

length would be 600. This model is trained in exactly similar settings as the above mentioned word based TextCNN model.

Fully Character Embeddings Model: We run VDCNN (Conneau et al., 2017) with 9 convolution layers with learning rate of 0.0001 reducing the learning rate by hald every 15 intervals for 100 epochs. We use a batch size of 64 and use stochastic gradient descent (SGD) as optimizization function with 0.9 momemtum.

BERT: For our BERT experiments we use the $BERT_{base}$ (uncased) model. $BERT_{base}$ model consists of 12 Transformer layers with 12 self-attention heads with 768 hidden dimensions and consists of 110 M total parameters. This model is trained in BookCorpus and English Wikipedia corpus. We attach a linear layer on top of $BERT_{base}$ model and the [CLS] token representation is fine-tuned on the training set. We use a binary cross-entropy loss to fine-tune BERT for our datasets. The fine tuned model is evaluated on the test set. We experimented with dropout values set at $(0.1, 0.2)$ between the transformer encoder layers. We achieved best results at dropout of 0.2, which

Model	pre	sword	Tok	Hatespeech					W-ATT	W-TOX
				0	1	2	3	4		
fastText$_{ngrams=1}$	N	N	N	69.7	71.8	84.2	95.5	82.2	93.3	95.6
fastText$_{ngrams=1}$	Y	N	N	69.6	74.8	84.5	95.7	79.5	93.5	95.6
fastText$_{ngrams=1}$ + BERT tokentization	N	Y	Y	71.2	**83.0**	83.0	95.2	83.4	94.5	96.1
fastText$_{ngrams=1}$ + Custom BPE	N	Y	Y	66.3	72.0	74.8	73.2	72.4	81.5	84.6
fastText$_{ngrams=2}$ + subword $(2-6)$	N	Y	N	64.3	71.2	75.9	92.2	85.7*	93.1	95.8
fastText$_{ngrams=2}$ + subword $(2-6)$ + BERT tok	N	Y	Y	64.1	66.7	75.1	93.4	85.3	93.9*	95.7
fastText$_{ngrams=2}$ + subword $(2-6)$ + BERT tokentization + preE	Y	Y	N	**71.5**	76.9	**87.9**	93.2	75.7	93.4	95.8*
TextCNN (Kim, 2014)	N	N	N	69.8	76.9	85.3	95.7	**85.9**	92.8	95.6
TextCNN + Character n-grams	N	N	N	70.6	78.1	87.1	96.3	**85.9**	93.2	95.9
TextCNN + BERT tokenization	N	N	Y	**71.6**	76.8	84.2	96.6	85.2	94.1	96.2
VDCNN (9 layers)	N	N	N	65.3	71.6	80.7	89.3	85.9	91.6	93.9
BERT (dropout = 0.2)	N	N	N	72.2	80.1	85.2	**97.0**	78.2	**95.7**	**96.8**

Table 1: We report Weighted F1-scores for the different models on the Hatespeech, W-TOX and W-ATT datasets.

we report in our experiments.

5.1 BERT Wordpiece Tokenizer Model with Word models

We use Wordpiece (BPE) model of BERT (Devlin et al., 2018) pretrained on BooksCorpus and English Wikipedia, produced using 30000 merge operations. BERT uses this model as precursor before encoding the text through transformer. We try to examine the benefit of the wordpiece text encoding vs the benefit we obtain from fine-tuning the pretrained LM. We hypothesize that pretrained BPE model splits a word into most frequent subwords found in the wikipedia corpus, which can help in mining the informative subwords. The informative subwords might prove very beneficial in noisy settings where we observe missing spaces and typos. In order to achieve this, we use this pretrained BPE model for encoding the document text before inputing to our word based models, TextCNN and fastText word variant. This is demonstrated in Figure 2. We have tried following variants with BERT Wordpiece tokenization as preprocessing step.

BERT Tokenizer with fastText$_{ngrams=2}$ TextCNN Word model: We preprocess the given dataset text using pretrained BPE model, and run a fastText bigram classification model on the preprocessed output. We also evaluate the TextCNN word model with the preprocessed text as input.

BERT Tokenizer with fastText subword: The preprocessed dataset with BERT trained BPE for training fastText subword model as described in Section 5.

Custom BPE model on the dataset: We also tried to examine if we would get a similar performance boost we obtained from BERT Wordpiece model by encoding text via a custom wordpiece model trained on the text. This helps us differentiate if the gains are from training a wordpiece model on a large text or if the gains are from using subword splitting. We used 30,000 number of merge operations for the custom BPE model, which is the same as in BERT BPE to aim for a meaningful comparison. We have also tried other values of merge operations from the custom BPE model, but none have yielded substantially better performance.

6 Results and Analysis

Table 1 presents the Weighted F1 score based on the support of each of the classes in the test set for our classification task. For a classification problem with N samples in the test set and C classes, Weighted F1 score [2] is defined as

$$\frac{1}{N}\sum_{i=1}^{C} n_i * F_i \qquad (1)$$

_____

[2] we use sklearn library for computing macro and weighted f1 scores in the paper https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Technique	PROCESSED DOC
Original	a complaint about your disruptive behavior here
	: https : / / en . wikipedia . org / wiki / wikipedia : administrators
	% 27_noticeboard / incidents # disruptive_users_vandalizing_article_about_spiro_koleka
Custom BPE	complain about your disruptive behavior here
	: https : / / en . wikipedia . org / wi@@ ki / wikipedia : administrators
	% 27_noticeboard / incidents # disrup@@ ti@@ ve_@@ user@@ s_@@
	vandali@@ z@@ ing_@@ article_@@ about_@@ spi@@ ro@@ _@@ ko@@
	le@@ ka
BERT token-tization	complain about your disrupt @@ive behavior here
	: https : / / en . wikipedia . org / wiki / wikipedia : administrators
	% 27 _ notice @@board / incidents # disrupt @@ive _ users
	_ van @@dal @@izing @@_ article @@_ about @@_ sp @@iro @@_ ko @@le
	@@ka

Table 2: Sample document split created by BERT BPE tokenizer, Custom BPE tokenizer

Model	W-ATT	W-TOX
WS (Mishra et al., 2018)	84.4	85.4
CONTEXT HS + CNG (Mishra et al., 2018)	87.4	89.3
fastText(ngrams=2)	85.2	86.8
fastText(ngrams=2, BERT BPE)	85.9	88.6
fastText(ngrams=2, BERT BPE, PreE)	**86.8**	88.6
Kim2014	82.7	88.4
Kim2014 (BERT BPE)	83.4	**89.3**
BERT (dropout = 0.2)	**89.5**	**90.6**

Table 3: Macro F1 average on the W-TOX and W-ATT datasets.

where n_i denotes the number of samples in class i. We have reported weighted F1 as the twitter data we obtained had only 17 samples for racism, with stratified CV split having only 4 samples on average. As the results on this label could be very random and prone to lot of variance due to very little number of samples in the train and test set, we choose to use weighted $F1$ over macro $F1$. We also have observed very high variance among performance in different CV splits, hence report the numbers separately on each of them.

Table 1 also mentions if each of the experiment involves using word splitting via BPE, either by pre-trained BERT Wordpiece tokenization model, or by training a custom BPE model on our given dataset. We have also highlighted the individual best performance from a modeling architecture with a $*$.

Table 2 presents the Macro $F1$ score on W-ATT and W-TOX datasets. Macro $F1$ score is defined

as :

$$\frac{1}{C} \sum_{i=1}^{C} F_i \qquad (2)$$

We have picked the best performing models from 1 for macro $F1$ comparison. We have also compared to previous approaches that have achieved best performance on these datasets. Mishra et al. (2018) reported Macro F1 on both validation and test data together. From their work it is unclear if the model is tuned on validation, and same data was used along with test to report numbers. Hence, we only use their number as reference. The main conclusions of these experiments are fourfold:

1. Pretrained BPE models transfer well: Pre-training a Wordpiece model on a large general corpus like wikipedia, and using this for encoding input text by splitting words has shown significant improvements for all the word based models. The fastText word model with bigrams (row 3 in table 1) trained with BERT tokenization achieves the best performance on 1st split of the hatespeech data, and also shows improvement over the native fastText bigrams model on Wiki-ATT dataset. The same observation can be made with TextCNN word model with preprocessing by pretrained BERT Wordpiece tokenization model(row 11 in Table 1). However, we have either noticed a slight degradation or an insignificant improvement by applying BPE encoding with fastText subword based model. This is expected as breaking the informative subwords from BERT into much smaller units might result in lot of noisy updates.

Predicted Label	Technique	Text
not_attack *attack**	Original BPE	believe that he was the greatest mother-fucker in the world believe that he was the greatest mother## -## fuck## er in the world
not_attack *attack**	Original BPE	many thanks for your leaving all edits alone in future with such idiotic diatribes many thanks for your leaving all edit## s alone in future with such idiot## ic## dia## tri## bes

Table 4: Qualitative samples from original text, and BERT Wordpiece model text. Actual label is marked with an asterisk. We can observe that BERT BPE model can effectively mine informative subwords as observed in general domain wikipedia

2. Fine tuning pretrained language models: We observe that fine-tuning large pretrained language models achieve best performance on toxicity dataset. BERT with dropout=0.2 achieves the best performance on most of the datasets and splits. It achieves better or at par performance over any word based model. Only fastText subwords and textCNN/fastText word based model trained on BERT Wordpiece tokenization preprocessing achieve higher performance compared to BERT finetuning. The gains from BERT Wordpiece tokenization model encoding to fastText word model outperforms performance of BERT model itself. We leave it as future work to further investigate the contribution from BPE Wordpiece tokenization to other classification tasks.

3. End to End Char models arent as effective as subword or word + char models: Adding character based embedding to aid word embedding based models, and subword models enhance the performance over their pure word based modeling baselines. This proves the hyptohesis of modeling at subword level definitely is beneficial for detecting abusive language. Interestingly, end to end character models arent as effective, which demonstrates the basic fact knowledge of word leads to a powerful representation, and word boundary information is still informative in noisy settings.

4. State-of-the-art performance on W-TOX and W-ATT with BERT finetuning: Table 3 shows the results for Macro $F1$ score of our models in comparison to previous approaches that have achieved best performance on these datasets. Mishra et al. (2018) reported Macro F1 on both validation and test data together. From their work it is unclear if the model is tuned on validation, and

same data was used along with test to report numbers. Hence, we only use their number as reference. We have also observed better numbers with their approach. We have achieved state of the art macro $F1$ score on W-ATT and W-TOX datasets with BERT finetuning. We have also added performance of BERT Wordpiece tokenized text with word based models for comparison, with their numbers running really close to those of BERT.

5. Effect of custom BPE model trained on the dataset: We have noticed significant performance degaradation as reported in Table 1, by tokenizing the text with custom BPE model trained on the W-ATT and W-TOX corpus, in comparison to using the original text or the BERT BPE encoded text. It's interesting to notice the text tokenized by BERT yields very informative subwords, that can help the word based model in comparison to subwords yielded by custom BPE model, even though the vocabulary size of both the models is very similar. Table 4 presents a qualitative example on how the BERT BPE mines informative subwords compared to the custom BPE model. One can note that BERT BPE model clearly splits the text on underscores extracts stem of the word in few cases.

7 Qualitative Analysis

Table 4 represents couple of examples from W-ATT dataset, where the pure word based model has failed to detect abusive language, but the model trained and tested on BERT Wordpiece tokenized text is able to detect the *attack*. As we can see, Wordpiece model trained on large wikipedia text with 30k operations(BERT) doesnt merge or create relatively uncommon word like *idiotic* from *idiot* and *ic*. This helps the model to just learn about *idiot* clearly from training set, and later use this

for clear demarcation.

8 Conclusion and Future Work

Existing literature has shown the importance of using finer units such as character or subword units to learn better models and robust representations for identifying abusive language in social media. In this work, we explore various combinations of such word decomposition techniques and present experiments that bring new insights and/or confirm previous findings. Additionally, we study the effectiveness of large pretrained language models trained on standard text in understanding noisy user generated text. We further investigate the effectiveness of subword units ("wordpieces") learned for unsupervised language modeling can improve the performance of bag-of-words based text classification models such as fastText. We evaluate our models on Twitter hatespeeech, Wikipedia toxicity and attack datasets.

Our experiments demonstrate that encoding noisy text via BERT wordpiece tokenization model before passing it through word-based models (fastText and TextCNN) can boost the performance of word-based models and achieve state-of-the-art performance. Based on our experiments, we conclude that subword models perform competitively with character-based models and occasionally outperform them. We observe that adding character embeddings to TextCNN model can slightly boost the performance compared to word-CNN models.

Our experiments on fine-tuning BERT show improvements on both Wikipedia toxicity and attack datasets. We observe that BERT can effectively transfer pretrained information to classifying tweets and user comments despite the domain shift of pre-training on BookCorpus, Wikipedia Text . Future work in this direction could include pre-training BERT on huge collection of social media text, which might further enhance the performance of identifying abusive language on social media text. Recent work by Wiegand et al. (2019) highlights that most of the datasets that study abusive language are prone to data sampling bias and abusive language identification on realistic scenario is much harder with higher percentage of implicit content. A potential future direction would be to explore how pretrained models on generic text could incorporate or handle implicit abuse.

References

Miguel Á Álvarez-Carmona, Estefanıa Guzmán-Falcón, Manuel Montes-y Gómez, Hugo Jair Escalante, Luis Villasenor-Pineda, Verónica Reyes-Meza, and Antonio Rico-Sulayes. 2018. Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In *Notebook Papers of 3rd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL), Seville, Spain*, volume 6.

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97.

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*, pages 759–760. International World Wide Web Conferences Steering Committee.

Cristina Bosco, Dell'Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9. CEUR.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. In *Ninth International AAAI Conference on Web and Social Media*.

Alexis Conneau, Holger Schwenk, Yann LeCun, and Loïc Barrault. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1107–1116.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Maeve Duggan. 2017. Online harassment 2017.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Twenty-seventh AAAI conference on artificial intelligence*.

Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2018. Neural character-based composition models for abuse detection.

Sandip Modha, Prasenjit Majumder, and Thomas Mandl. 2018. Filtering aggression from multilingual social media feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC–1), Santa Fe, USA*.

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*, pages 145–153. International World Wide Web Conferences Steering Committee.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. *arXiv preprint arXiv:1705.09993*.

Matthew Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. In *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, volume 17 of *Bochumer Linguistische Arbeitsberichte*, pages 6–9.

Magnus Sahlgren, Tim Isbister, and Fredrik Olsson. 2018. Learning representations for detecting abusive language. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 115–123.

Haji Mohammad Saleem, Kelly P. Dillon, Susan Benesch, and Derek Ruths. 2017. A web of hate: Tackling hateful speech in online social spaces. *CoRR*, abs/1709.10159.

Andrew Schrock and Danah Boyd. 2011. Problematic youth interaction online: Solicitation, harassment, and cyberbullying. *Computer-mediated communication in personal relationships*, pages 368–398.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Robert S Tokunaga. 2010. Following you home from school: A critical review and synthesis of research on cyberbullying victimization. *Computers in human behavior*, 26(3):277–287.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017.*, pages 86–95.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

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.

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.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of Abusive Language: the Problem of Biased Datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 602–608.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.

Nancy E Willard. 2007. *Cyberbullying and cyberthreats: Responding to the challenge of online social aggression, threats, and distress*. Research press.

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*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

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

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Jian Zhu, Zuoyu Tian, and Sandra Kübler. 2019. Umiu@ ling at semeval-2019 task 6: Identifying offensive tweets using bert and svms. *arXiv preprint arXiv:1904.03450*.