Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network

Paceholder
Paceholder
Paceholder
Paceholder

Paceholder
Paceholder
Paceholder
Paceholder

Paceholder
Paceholder
Paceholder
Paceholder

ABSTRACT

In recent years, the increasing propagation of hate speech on social media and the urgent need for effective counter-measures have drawn significant investment from governments, companies, as well as empirical research. Despite a large number of emerging scientific studies to address the problem, existing methods are limited in several ways, such as the lack of comparative evaluations which makes it difficult to assess the contribution of individual works. This paper introduces a new method based on a deep neural network combining convolutional and long short term memory networks, and conducts an extensive evaluation of the method against several baselines and state of the art on the largest collection of publicly available datasets to date. We show that compared to previously reported results on these datasets, our proposed method outperforms on 6 out of 7 datasets by between 0.2 and 13.8 points in F1. We also carry out further analysis using automatic feature selection to understand the impact of the conventional manual feature engineering process that distinguishes most methods in this field. Our findings challenge the existing perception of the importance of manual feature engineering, as we show that: the automatic feature selection algorithm drastically reduces the original feature space by over 90% and selects predominantly generic features from datasets; nevertheless, machine learning algorithms perform better using automatically selected features than carefully crafted task-specific features.

CCS CONCEPTS

• **Computing methodologies → Information extraction**;

KEYWORDS

hate speech, CNN, LSTM, deep learning, SVM, feature selection, machine learning, classification, NLP

ACM Reference Format:
Paceholder, Paceholder, and Paceholder. 2018. Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network. In *Proceedings of ACM The Web conference (WWW'2018).* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The exponential growth of social media such as Twitter and community forums has revolutionised communication and content publishing, but is also increasingly exploited for the propagation of hate speech and the organisation of hate based activities [2, 3]. The anonymity and mobility afforded by such media has made the breeding and spread of hate speech - eventually leading to hate crime - effortless in a virtual landscape beyond the realms of traditional law enforcement.

The term 'hate speech' was formally defined as 'any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic' [23]. In the UK, there has been significant increase of hate speech towards the migrant and Muslim communities following recent events including leaving the EU, the murder of MP Jo Cox (in which case there was a surge of conversations chanting the murderer as 'hero' on Twitter [1]), the Manchester and the London attacks [13]. This correlates to record spikes of hate crimes [25, 34], and cases of threats to public safety due to its nature of inciting hate crimes, such as that followed the Finsbury van attack [21]. In the EU, surveys and reports focusing on young people in the EEA region show rising hate speech and related crime based on religious beliefs, ethnicity, sexual orientation or gender, as 80% of respondents have encountered hate speech online and 40% felt attacked or threatened [20]. Statistics also show that in the US, hate speech and crime is on the rise since the Trump election [24].

For years, social media companies such as Twitter, Facebook, and YouTube have been combating this issue [14, 18], and it has been estimated that hundreds of millions of euros are invested every year on counter-measures including manpower [10]. However, they are still being criticised for not doing enough. This is largely because such measures involve manually reviewing online contents to identify and delete offensive materials. The process is labour intensive, time consuming, and not sustainable or scalable in reality [4, 10, 34].

The pressing need for scalable, automated methods of hate speech detection has attracted increasing research from both the Natural Language Processing (NLP) and Machine Learning (ML) communities in the last few years. State of the art primarily casts the problem as a supervised document classification task [28]. These methods can be divided into two categories: one relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes, and Logistic Regression [3, 7, 9, 12, 16, 19, 32–36] (**classic methods**); the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multi-layers of abstract features from raw data [10, 22, 27, 31] (**deep learning methods**).

Existing works are still limited in a number of ways. First, despite the increasing amount of studies, there is still no comparative evaluation and it remains difficult to compare their performance on different tasks [28]. One primary reason for this is the lack of public datasets. Second, for deep learning methods, existing ones are largely based on convolutional neural networks (CNN) or long short term memory (LSTM), a type of recurrent neural networks. Intuitively, the first learns features similar to n-gram sequences while the second learns sequence orders that are both useful for classification tasks. Hence a natural question would be whether we can combine both types of networks in a single architecture to leverage their benefits. Third, for classic methods, while a plethora of task-specific features have been investigated in the literature [28], it is not clear what their individual contribution to the task really is, and the comparison against deep learning methods so far is rather inconclusive.

In this work, we introduce a CNN+LSTM neural network model extending that in [30] by adding dropout and pooling layers to regularise learning for better performance, and carry out extensive experiments to address the above mentioned limitations in the state of the art. We also create a public dataset of hate speech by collecting tweets on the subjects of religion and refugees, which extends currently available datasets [7, 33, 34] by both quantity and subject coverage. We test our model on all these public datasets, and set new benchmark as we show that compared to previously reported results, our model outperforms on 6 out of 7 datasets by as much as 13.8% in F1. Our experiments also contain a classic method (SVM) for comparison where our analysis shows that the benefits of the manual feature engineering practice significantly diminish with the use of automatic feature selection techniques, as the latter tends to select predominantly generic features, reducing the original feature space by more than 90% and often discarding certain types of features completely. Yet this still leads to better results on all datasets compared against cases when the full feature sets are used, and on some datasets even produces performance that is on par with our proposed CNN+LSTM model.

The remainder of this paper is structured as follows. Section 2 reviews related work on hate speech detection and other relevant fields; Section 3 introduces the CNN+LSTM model; Section 4 describes our data collection and annotation process; Section 5 discusses experiment results, including all comparative models and on all datasets; finally Section 6 concludes this paper.

2 RELATED WORK

2.1 Terminology

Recent years have seen an increasing number of research on hate speech detection as well as other related areas. As a result, the term 'hate speech' is often seen to co-exist or become mixed with other terms such as 'offensive', 'profane', and 'abusive languages', and 'cyber bullying'. To distinguish them, we identify that hate speech 1) targets individual or groups on the basis of their characteristics (targeting characteristics); 2) demonstrates a clear intention to incite harm, or to promote hatred; 3) may or may not use offensive or profane words. For example:

`'Assimilate? No they all need to go back to their own countries. #BanMuslims Sorry if someone disagrees too bad.'`

In contrast, `'All you perverts (other than me) who posted today, needs to leave the O Board. Dfasdfdasfadfs'` is an example of abusive language, which often bears the purpose of insulting individuals or groups, and can include hate speech, derogatory and offensive language [22]. `'i spend my money how i want bitch its my business'` is an example of offensive or profane language, which is typically characterised by the use of swearing or curse words. `'Our class prom night just got ruined because u showed up. Who invited u anyway?'` is an example of bullying, which has the purpose to harass, threaten or intimidate typically individuals rather than groups.

In the following, we cover state of the art in all these areas with a focus on hate speech. Our experiments will only involve hate speech, due to both dataset availability and the focus of this work.

2.2 State of the Art

As mentioned before, we divide state of the art into classic and deep learning based methods depending on whether there is an automated feature learning process.

Classic methods requires manually designing and encoding features of data instances into feature vectors, which are then directly used by classifiers.

[28] summarised several types of features used in the state of the art. *Simple surface features* such as bag of words, word and character n-grams have shown to be highly predictive in hate speech detection [3, 12, 16, 19, 31–34], as well as other related tasks [4, 22, 36, 37]. Recent research [19] has also shown character n-grams to be more effective than word n-grams, as they are more likely to capture the similarities of prevalent unusual spellings (e.g., `'kill yrslef'`). Other surface features can include URL mentions, hashtags, punctuations, word and document lengths, capitalisation, etc [4, 7, 22]. *Word generalisation* includes the use of word clusters [32], and techniques such as topic modelling [35, 37] and word embedding learning [9, 22, 31, 36] that learn low-dimensional, dense feature vectors for words from unlabelled corpora. Such word vectors are then used to construct feature vectors of messages. *Sentiment analysis* makes use of the degree of polarity expressed in a message [3, 7, 11, 31]. *Lexical resources* are often used to look up specific negative words (such as slurs, insults, etc.) in messages as the presence of such words can be predictive features [3, 11, 22, 35]. It is worth to note that early methods such as [29] are heavily based on lexical resources. However it has been shown that such features alone are not very effective [4]. *Linguistic features* utilise syntactic information such as Part of Speech (PoS) and certain dependency relations as features [3, 4, 7, 11, 37]. For example, [3] noted that 'othering phrases' denoting a 'we v.s. them' stance are common in hate speech, while [4] and [37] used dependency relations as features for detecting offensive language and cyber bullying. *Meta-information* refers to data about messages, such as gender identity of a user associated with a message [33, 34], or high frequency of profane words in a user's post history [6, 35]. In addition, *Knowledge-Based features* such as messages mapped to stereotypical concepts in a

knowledge base [8] and *multimodal information* such as image captions and pixel features [37] were used in cyber bully detection but only in very confined context [28].

In terms of classifiers, existing methods are predominantly supervised. Among these, Support Vector Machines (SVM) is the most popular algorithm [3, 4, 7, 12, 19, 32, 35, 36], while other algorithms such as Naive Bayes [4, 7, 16, 19, 36], Logistic Regression [7, 9, 19, 33, 34], and Random Forest [7, 35] are also used.

Despite the diverse types of features introduced, little is known about the contributions of different types of features in a single classifier. Most methods simply 'use them all' by concatenating all feature types into high-dimensional, sparse feature vectors that are prone to over-fitting, especially on short texts such as tweets [31]. Some [3, 7] applied an automated statistical feature selection process to reduce and optimise the feature space, while others [4, 11] did this manually. However, the impact of feature selection is unknown and therefore, whether the different types of features are contributing to the classification remains questionable.

Deep learning based methods employ Deep artificial Neural Networks (DNN) to learn abstract feature representations from input data through its multiple stacked layers for the classification of hate speech. The input can take various forms of feature encoding, including any of those used in the classic methods. However, the key difference is that in such a model the input features are not directly used for classification. Instead, the multi-layer structure learns new abstract feature representations that prove to be more effective for learning. For this reason, deep learning based methods typically shift its focus from manual feature engineering to the network topology, which is carefully designed to automatically extract useful features from a simple input feature representation. Note that this categorisation excludes those methods [9, 19, 36] that use DNN to learn word or text embeddings and subsequently apply another classifier (e.g., SVM, logistic regression) to use such embeddings as features for classification. Instead, we focus on DNN methods that perform the classification task itself.

To the best of our knowledge, methods belonging to this category in this domain and related areas include [2, 10, 27, 31], all of which use simple word and/or character based one-hot encoding as input features to their models, while [31] also use word polarity. The most popular network architectures are Convolutional Neural Network (CNN) and Long Short-Term Memory network (LSTM). In the literature, CNN is well known as an effective network to act as 'feature extractors', whereas LSTM is a type of powerful recurrent network for modelling orderly sequence learning problems [17, 26]. In the context of hate speech classification, intuitively, CNN extracts word or character combinations [2, 10, 27] (e.g., phrases, n-grams), LSTM learns long-range word or character dependencies in tweets [2, 31].

While each type of network has proved effective for hate speech classification, no work has explored combining both structures in a single network. In theory, CNN+LSTM networks are powerful structures to capture long-term dependencies between features extracted by CNNs. In practice, they are found to be more effective than structures solely based on CNNs or LSTMs in tasks such as gesture [30] and activity recognition [26] where the networks learn temporal evolutions of different regions between frames, and Named Entity Recognition [5] where the class of a word sequence can depend on the class of its preceding word sequence. We hypothesize that CNN+LSTM can be also effective for hate speech classification, as it may capture co-occurring word n-grams as useful patterns for classification, such as the pairs `(muslim refugees, deported)` and `(muslim refugees, not welcome)` in the sentence `'These muslim refugees are not welcome in my Country they should all be deported ...'`.

2.3 Datasets

It is widely recognised that a major limitation of state of the art is the lack of comparative evaluation on publicly available datasets [28]. The large majority of existing works were evaluated on privately collected datasets, often for different problems. [22] claimed to have created the largest datasets for abusive language by annotating comments posted on Yahoo! Finance and News. The datasets were later used by [19]. However, the datasets are not publicly available. Also, as we illustrated before, abusive language can be different from hate speech.

Currently, the only publicly available hate speech datasets include those reported in [7, 10, 27, 33, 34]. [34] annotated 16,914 tweets, including 3,383 as 'sexist', 1,972 as 'racist' and 11,559 as 'neither'. The corpus was collected by searching for tweets containing frequently occurring terms (based on some manual analysis) in tweets that contain hate speech and references to specific entities. It was then annotated by crowd-sourcing over 600 users. The dataset was later expanded in [33], where some 6,900 tweets were collected with about 4,000 new to their previous dataset. This dataset was then annotated by two groups of users to create two different versions: domain experts who are either feminist or anti-racism activist; and amateurs that are crowd-sourced. Experiments showed that amateur annotators are more likely than expert annotators to label tweets as hate speech. However, systems trained on expert annotations outperformed that trained on amateur annotations. Later in [10], the authors merged both expert and amateur annotations in this dataset by using majority vote, giving expert annotations double weight; and in [27], the dataset of [34] was merged with the expert annotations in [33] to create a single dataset. [7] annotated some 24,000 tweets for 'hate speech', 'offensive language but not hate', and 'neither'. They began with filtering tweets using a hate speech lexicon from Hatebase.org, and selected a random sample for annotation. It was found that distinguishing hate speech from non-hate offensive language is a challenging task, as hate speech does not always contain offensive words while offensive language does not always express hate.

3 METHODOLOGY

This section introduces our method for hate speech detection. We firstly describe a light pre-processing procedure (Section 3.1), then introduce our CNN+LSTM model (Section 3.2), followed by our choice of input features initialisation for the model (Section 3.3).

3.1 Pre-processing

Given a tweet, we apply light pre-processing to normalise its text as follows:

- remove the following characters: `| : , ; & ! ? \`

- normalise hashtags into words, so '`#refugeesnotwelcome`' becomes '`refugees not welcome`'. This is because such hashtags are often used to compose sentences. We use dictionary based look up to split such hashtags.
- lowercase and stemming, to reduce word inflections
- removing any tokens with a document frequency less than 5

The pre-process reduces vocabulary size in a dataset (particularly prominent in tweets due to the colloquial nature of the texts) and to some degree, addresses the sparsity in word-based feature representations. We also tested other alternatives such as keeping word inflections, using lemmatisation instead of stemming, and lower document frequencies. But empirically we found the above mentioned process to contribute to slightly better accuracy.

3.2 The CNN+LSTM architecture

Our CNN+LSTM network is illustrated in Figure 1. The first layer is a word embedding layer, which maps each text message (in generic terms, a 'sequence') into a a real vector domain. To do so, we map each word onto a 300 dimensional real valued vector, where each element is the weight for that dimension for that word. We also constrain each sequence to be 100 words, truncating long messages and pad the shorter messages with zero values. We experiment different ways of generating word vectors and describe these in Section 3.3.

The embedding layer passes an input feature space with a shape of 100×300 to a drop-out layer with a rate of 0.2, the purpose of which is to regularise learning to avoid overfitting. Intuitively, this can be thought of as randomly removing a word in sentences and forcing the classification not to rely on any individual words. The output feeds into a 1D convolutional layer with 100 filters with a window size of 4, padding the input such that the output has the same length as the original input. The rectified linear unit function is used for activation. This convolves the input feature space into a 100×100 representation, which is then further down-sampled by a 1D max pooling layer with a pool size of 4 along the word dimension, producing an output of shape 25×100. Each of the 25 dimension can be considered an 'extracted feature'. These then feed into the LSTM layer, which treats the extracted feature dimension as timesteps and outputs 100 hidden units per timestep. A global max pooling layer follows to 'flatten' the output space by taking the highest value in each timestep dimension, producing a 1×100 vector. Finally, a softmax layer takes this vector as input to predict probability distribution over all possible classes (n), which will depend on individual datasets. We use the categorical cross entropy loss function and the Adam optimiser [15] to train the model on a batch size of 100 and 10 epochs.

Our network architecture is similar to those in [5, 26, 30]. The key difference is the use of drop-out to regularise learning and global max pooling to extract features from the LSTM layer. Both designs are task-specific and found to be empirically helpful in our experiments. [5] also used bi-directional LSTM to capture context from both sides of a word in the Named Entity Classification task, while [26, 30] stacked multiple convolutional layers to extract hierarchical features in image processing. We do not use such complex models as we show that a simple CNN+LSTM is already performing

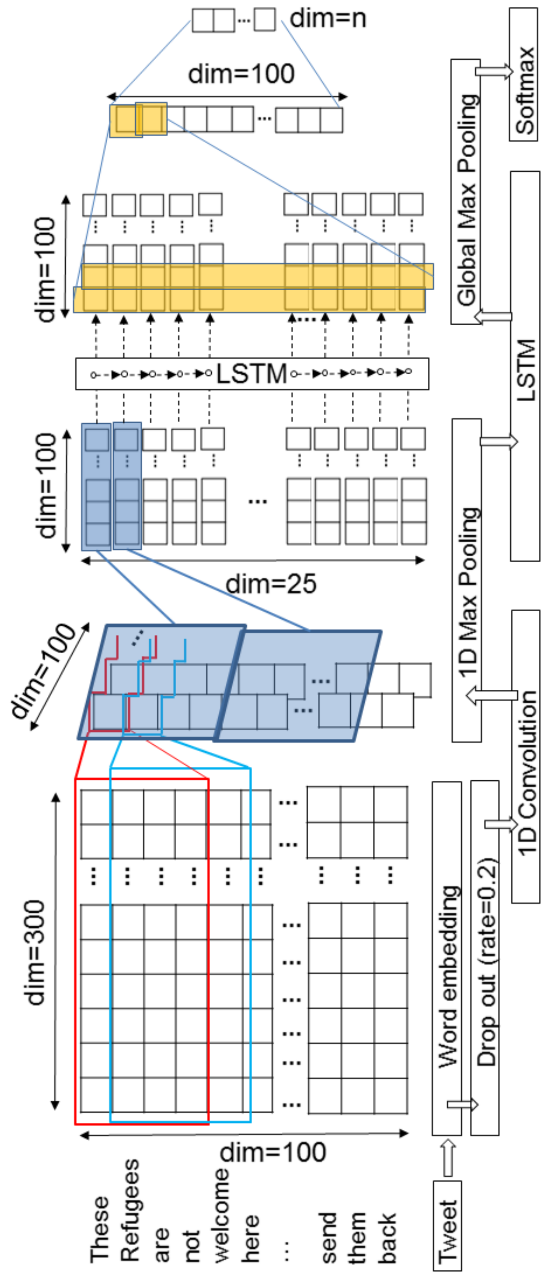

Figure 1: The CNN+LSTM network architecture

well in our task, which confirms our hypothesis. For the same reason, we build our model on only word embeddings although many suggest that character-level features can be more effective. We show later in experiments that such a structure is very powerful that it even outperforms DNN models based on character embeddings.

3.3 Word embedding learning

We test two types word embeddings. For the first, we initialise the weights in the embedding layer randomly and let our model learn the embeddings (i.e. weights) on the fly (**emb-learn**). For the second, we use pre-trained word embeddings to set the weights of our embedding layer. In this work, we use the one trained on the 3-billion-word Google News corpus with a skip-gram model[1].

A potential issue with using pre-trained word embeddings is out-of-vocabulary (OOV) words. We use two methods to address this: 1) randomly setting the OOV word vector following a continuous uniform distribution (**emb-ggl1**); 2) randomly selecting an in-vocabulary word to use its vector (**emb-ggl2**).

4 DATASET CREATION

As introduced before, the only publicly available hate speech datasets include that of [7] and [33, 34], which are later used to create variants used in [10, 27]. While [7] classify hate speech in general without identifying the targeting characteristics, [33, 34] collected data for sexism and racism. In this work, we create a different dataset by collecting tweets discussing refugees and Muslims, which were focus of discussion during the time of writing due to various recent incidents [1, 13, 21]. All tweets are annotated for two classes: hate and non-hate, firstly by a computational linguistic researcher and then cross-checked by a student researcher. Disputed annotations were discussed among them and corrected to ensure both agree with the correction. Annotators followed the general definition in [34] for annotation.

To collect the data, we follow the mainstream bootstrapping approach [34] that starts with an initial search for tweets containing common slurs and terms used pertaining to targeting characteristics, then manually identify frequently occurring terms in tweets that contain hate speech and references to specific entities (frequent keywords), then further filter the tweets with these frequent keywords.

Specifically, we started with using the Twitter Streaming API to collect tweets containing any of the following words for a period of 7 days: `muslim, islam, islamic, immigration, migrant, immigrant, refugee, asylum`. This created a corpus of over 300,000 tweets (duplicates and retweets removed), from which we randomly sampled 1,000 for annotation (batch 1). However, it was found that tweets annotated as hate speech were extremely rare (< 1%). Therefore, we manually inspected the annotations and further filtered the remaining tweets (disjoint with batch 1) by the following words found to be frequent for hate speech: `ban, kill, die, back, evil, hate, attack, terrorist, terrorism, threat, deport`. We then sampled another 1,000 tweets (batch 2) from this collection for annotation. However, the amount of true positives was still very low (1.1%).

Therefore we created another batch (batch 3) by using the Twitter Search API to retrieve another 1,500 tweets with the following hashtags considered to be strong indicators of hate speech: `#refugeesnotwelcome, #DeportallMuslims, #banislam, #banmuslims, #destroyislam, #norefugees, #nomuslims`. The dataset however, contains over 400 tweets after removing duplicates, and about 75% were annotated as hate speech. Finally we merge all three batches to create a single dataset, which we make public to encourage future comparative evaluation[2].

5 EXPERIMENT

5.1 Datasets

We use a total of 7 public datasets including ours for evaluation, as shown in Table 1. To our knowledge, this is by far the most comprehensive collection of hate speech datasets used in any studies.

Dataset	#Tweets	Classes (#tweets)	Targeting characteristics
WZ-L	16,093	racism (1,934) sexism (3,149) neither (11,010)	racism, sexism
WZ-S.amt	6,594	racism (123) sexism (1,073) both (15) neither (5,383)	racism, sexism
WZ-S.exp	6,594	racism (85) sexism (777) both (35) neither (5,697)	racism, sexism
WZ-S.gb	6,594	racism (90) sexism (911) both (27) neither (5,564)	racism, sexism
WZ-LS	18,625	racism (2,012) sexism (3,769) both (30) neither (12,810)	racism, sexism
DT	24,783	hate (1,430) non-hate (23,353)	general
RM	2,435	hate (414) non-hate (2,021)	refugee, Muslim

Table 1: Statistics of datasets used in the experiment

WZ-L is the larger dataset created in [34]. **WZ-S.amt** is the smaller dataset created in [33], annotated by amateurs; while **WZ-S.exp** is the same dataset annotated by experts. In [10], the authors took the WZ-S.amt and WZ-S.exp datasets to create a new version by taking the majority vote from both amateur and expert annotations where the expert was given double weights. We follow the same practice and in case of tie, we take the expert annotation. We refer to this dataset as **WZ-S.gb**. Further, [27] combined the WZ-L and the WZ-S.exp datasets into a single dataset and in case of duplicates, we take the annotation from WZ-L. We refer to this dataset as **WZ-LS**. All these datasets only contain the tweet IDs, some of which have been deleted or made private at the time of writing and therefore, the numbers in Table 1 may be slightly different from the original studies.

DT refers to the dataset created in [7]. It also contains tweets annotated as 'abusive (but non-hate)'. In this work, we set such

[1]https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[2]Find out at: https://github.com/ziqizhang/chase/tree/master/data

annotations to be 'non-hate' so the dataset contains only two classes. Finally, our dataset on refugees and Muslims is referred to as **RM**.

5.2 Baseline and comparative models

Baselines We create a number of baselines. **First**, we use a linear SVM model described in [7][3]. Each tweet is firstly pre-processed using the procedure described in Section 3.1. Next, following the original work, a number of different types of features are used as below. We refer to these as the **Basic** feature set:

- Surface features: word unigram, bigram and trigram each weighted by TF-IDF (**ngrams**); number of mentions (**#mentions**), and hashtags[4] (**#hashtags**); number of characters (**#char**), and words (**#word**);
- Linguistic features: Part-of-Speech (PoS) tag unigrams, bigrams, and trigrams (**pos-ngrams**), also weighted by their TF-IDF and removing any candidates with a document frequency lower than 5; number of syllables (**#syllb**); Flesch-Kincaid Grade Level (**FKGL**) and Flesch Reading Ease (**FRE**) scores that are functions calculating a numeric score based on the number words and syllables in a sentence, to measure the 'readability' of a document
- Sentiment features: sentiment polarity scores of the tweet (**sntm**).

Extending on this, we add additional surface based features as follows and refer to these as the **Enhanced** feature set:

- number of misspellings within a tweet (**#mspell**): we check the pre-processed tweet against both a UK and US English dictionaries, then return a score that is the ratio between the number of misspelled words and the number of all words in the tweet. The intuition is that, compared to normal tweets, hate speech may contain higher proportion of slangs and racial slurs, which may not be captured by dictionaries.
- number of emoji's (**#emoji**) uses regular expression to find tokens matching the format of an emoji from an original tweet and return a number.
- number of special punctuations (**#punc**) uses regular expression to count the numbers of question marks (?) and exclamation marks (!) as they can be used as an expletive.
- percentage of capitalisation (**CAP**) computes the percentage of characters that are capitalised within the tweet.
- hashtags (**hashtag**): the lowercase hashtags from tweets.

As discussed before, classic methods using diverse feature sets like these often end up with a high-dimensional, sparse feature space that can lead to problems such as overfitting. One approach to get around the problem is feature selection. While many techniques are available, we use a state of the art feature selection process based on Logistic Regression with L1-regularization as the estimator on the training data[5]. This calculates a 'feature importance' score for each feature, which is discarded if its score is below a threshold parameter[6]. To investigate the impact of feature selection, we create four baseline models: **SVM** using the linear SVM model on all Basic features identified above; **SVM$_{\mathbf{fs}}$** that applies the feature selection process to the Basic features; **SVM+** using the linear SVM model with the Enhance feature set; and **SVM$_{\mathbf{fs}}$+** that applies feature selection to the Enhanced features.

Second, we also create another baseline by modifying our CNN+LSTM network. To do so, we remove the drop-out between the embedding and convolutional layer, and the global max pooling layer between the LSTM and the softmax layers. We show that these two layers can contribute to improved accuracy in the task. This baseline model will be referred to as **CNN+LSTM$_{\mathbf{base}}$**.

We apply all three baselines on all seven datasets and compare the results against our model, **CNN+LSTM**.

State of the art In addition to the baselies, we also compare our results against those reported figures in [10, 27, 33, 34] on the corresponding datasets.

5.3 Implementation, parameter tuning, and evaluation metrics

We used the Python Keras[7] with TensorFlow backend [8] and the scikit-learn[9] library to implement all models[10]. For each dataset, we split it into 75:25 to use 75% for parameter tuning using 5-fold cross-validation experiments, and test the optimised model on the 25% held-out data. All the previously mentioned parameter settings are tuned in this way. We report our results using the standard Precision (P), Recall (R), and F1-measure below.

5.4 Results and discussion

To be comparable with results reported in the state of the art, we calculate micro-average P, R, F1 over all classes in each dataset. Tables 2 to 8 compare our model against the baselines as well as the reported figures by state of the art (on an 'as-is' basis where available, indicated by citations) on each of the seven datasets. The highest figures are highlighted in **bold**. Note that the SVM baseline represents the model in [7] and therefore, we do not repeat it in the tables.

Baseline SVMs First, comparing SVM+ against SVM, on all but one (DT) dataset, we notice improvement in all of Precision (0.8~2.8), Recall (0.2~2.9) and F1 (0.5~2.9), suggesting that in most cases, the enhanced features are indeed useful to the task. Second, comparing either of SVM or SVM+ against its feature selected version (SVM$_{fs}$, SVM+$_{fs}$), it is apparent that both can benefit significantly as we notice positive improvement in most datasets in Precision (1.8~9.3 for SVM$_{fs}$, 0.4~8.5 for SVM+$_{fs}$), Recall (0.9~9.4 for SVM$_{fs}$, 0.2~8.7 for SVM+$_{fs}$), and F1 (1.3~9.3 for SVM$_{fs}$, 0.5~8.7 for SVM+$_{fs}$), except only one occasion of a slight drop in Precision (SVM+$_{fs}$ vs SVM+ on WZ-S.exp). Notice also that with feature selection the SVM-based models achieved close-to-best performance on several datasets and even won on one (WZ-L). Third, comparing the two feature selected models SVM$_{fs}$ vs SVM+$_{fs}$, interestingly, the benefit of the enhanced

[3]Code: https://github.com/t-davidson/hate-speech-and-offensive-language
[4]These are extracted from the original tweet before pre-processing which splits hashtags.
[5]http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html
[6]For Logistic Regression with L1-regularization the default value of $1e - 5$ is used.

[7]https://keras.io/
[8]https://www.tensorflow.org/
[9]http://scikit-learn.org/
[10]Our code is shared here:https://github.com/ziqizhang/chase

Models	Precision	Recall	F1
SVM (also [7])	73.5	73.6	73.6
SVM_{fs}	**81.5**	**81.4**	**81.4**
SVM+	74.3	74.2	74.2
$SVM+_{fs}$	81.3	81.3	81.3
$CNN+LSTM_{base}$, emb-learn	78.8	78.8	78.8
$CNN+LSTM_{base}$, emb-ggl1	79.1	79.1	79.1
$CNN+LSTM_{base}$, emb-ggl2	80.2	80.2	80.2
Waseem et al. (best F1 in [34])	72.9	77.7	73.9
CNN+LSTM, emb-learn	79.0	78.7	78.8
CNN+LSTM, emb-ggl1	80.9	80.9	80.9
CNN+LSTM, emb-ggl2	80.3	80.2	80.2

Table 2: Results against baselines on the WZ-L dataset

Models	Precision	Recall	F1
SVM (also [7])	89.2	89.1	89.1
SVM_{fs}	91.0	90.0	90.4
SVM+	91.1	89.7	90.4
$SVM+_{fs}$	90.9	90.8	90.9
$CNN+LSTM_{base}$, emb-learn	90.3	90.0	90.2
$CNN+LSTM_{base}$, emb-ggl1	90.6	90.6	90.6
$CNN+LSTM_{base}$, emb-ggl2	91.0	90.9	91.0
Waseem et al. (The 'best' feature set in [33])	**92.5**	**92.5**	91.2
CNN+LSTM, emb-learn	89.9	89.8	89.8
CNN+LSTM, emb-ggl1	90.9	90.8	90.9
CNN+LSTM, emb-ggl2	91.4	91.3	**91.4**

Table 3: Results against baselines on the WZ-S.exp dataset

Models	Precision	Recall	F1
SVM (also [7])	86.1	86.3	86.2
SVM_{fs}	90.3	90.6	90.5
SVM+	86.9	86.5	86.7
$SVM+_{fs}$	90.2	89.8	90.0
$CNN+LSTM_{base}$, emb-learn	88.2	88.3	88.2
$CNN+LSTM_{base}$, emb-ggl1	**91.4**	91.4	**91.4**
$CNN+LSTM_{base}$, emb-ggl2	90.2	90.2	90.2
Waseem et al. (The 'best' feature set in [33])	85.5	86.7	83.9
CNN+LSTM, emb-learn	89.9	89.1	89.0
CNN+LSTM, emb-ggl1	91.2	**91.6**	**91.4**
CNN+LSTM, emb-ggl2	**91.4**	91.2	91.3

Table 4: Results against baselines on the WZ-S.amt dataset

Models	Precision	Recall	F1
SVM (also [7])	86.1	86.3	86.2
SVM_{fs}	90.3	90.6	90.5
SVM+	86.8	86.6	86.7
$SVM+_{fs}$	90.2	89.8	90.0
$CNN+LSTM_{base}$, emb-learn	90.7	90.4	90.5
$CNN+LSTM_{base}$, emb-ggl1	91.9	91.8	91.8
$CNN+LSTM_{base}$, emb-ggl2	91.4	91.3	91.3
Gamback et al. (Best F1 in [10])	85.7	72.1	78.3
CNN+LSTM, emb-learn	91.4	91.4	91.4
CNN+LSTM, emb-ggl1	**92.1**	**92.1**	**92.1**
CNN+LSTM, emb-ggl2	91.8	91.8	91.8

Table 5: Results against baselines on the WZ-S.gb dataset

Models	Precision	Recall	F1
SVM (also [7])	71.7	71.5	71.6
SVM_{fs}	81.0	80.9	80.9
SVM+	72.6	72.5	72.5
$SVM+_{fs}$	81.1	81.2	81.2
$CNN+LSTM_{base}$, emb-learn	79.8	79.2	79.5
$CNN+LSTM_{base}$, emb-ggl1	79.5	79.8	79.6
$CNN+LSTM_{base}$, emb-ggl2	79.5	79.2	79.4
Park et al. [27]			
- WordCNN	81.8	81.6	81.6
- CharacterCNN	80.1	81.1	81.1
- HybridCNN	**82.7**	**82.7**	**82.7**
CNN+LSTM, emb-learn	80.3	80.6	80.4
CNN+LSTM, emb-ggl1	81.9	81.8	81.8
CNN+LSTM, emb-ggl2	81.6	81.5	81.5

Table 6: Results against baselines on the WZ-LS dataset

Models	Precision	Recall	F1
SVM (also [7])	86.6	86.4	86.5
SVM_{fs}	89.5	89.4	89.4
SVM+	86.2	86.4	86.3
$SVM+_{fs}$	89.5	89.7	89.6
$CNN+LSTM_{base}$, emb-learn	93.3	93.3	93.3
$CNN+LSTM_{base}$, emb-ggl1	93.3	93.3	93.3
$CNN+LSTM_{base}$, emb-ggl2	92.7	92.4	92.6
CNN+LSTM, emb-learn	93.4	92.9	93.1
CNN+LSTM, emb-ggl1	**94.2**	93.9	**94.1**
CNN+LSTM, emb-ggl2	94.0	**94.1**	94.0

Table 7: Results against baselines on the DT dataset

features diminishes, as we notice only on 3 datasets (WZ-L, WZ-LS, DT) an improvement in F1 by $SVM+_{fs}$ over SVM_{fs}. The range of improvements also shrinks and on other datasets, dips into negative region (-0.3~0.1 for Precision, -0.8~0.8 in Recall, -0.5~0.5 in F1).

These observations suggest that, on the one hand, feature selection can be a very powerful technique to improve performance of classic methods while in the mean time, reducing the need for feature engineering as it appears that regardless of the initial input feature set used, SVM produced similar results when using feature selection. On the other hand, the Logistic Regression based feature selector may be discriminating features too aggressively, losing some useful features and resulting in decreased performance in $SVM+_{fs}$ against SVM_{fs} on some datasets. While investigating alternative feature selection algorithms is beyond the scope of this

Models	Precision	Recall	F1
SVM (also [7])	85.7	85.7	85.7
SVM_{fs}	88.8	88.7	88.7
SVM+	88.1	87.8	88.0
SVM+_{fs}	88.5	88.8	88.7
$\text{CNN+LSTM}_{\text{base}}$, emb-learn	87.4	87.4	87.4
$\text{CNN+LSTM}_{\text{base}}$, emb-ggl1	89.6	89.7	89.6
$\text{CNN+LSTM}_{\text{base}}$, emb-ggl2	90.5	90.5	90.5
CNN+LSTM, emb-learn	91.4	90.6	91.0
CNN+LSTM, emb-ggl1	90.9	90.9	90.9
CNN+LSTM, emb-ggl2	**91.9**	**91.9**	**91.9**

Table 8: Results against baselines on the RM dataset

work, in Section 5.5 we gauge into details of feature selection to understand the impact of different features.

Baseline neural network models Among the different types of word embeddings used, it is clear that those pre-trained on the Google News corpus (emb-ggl1, emb-ggl2) are more effective than that learned from the training data (emb-learn). This is likely because the size of the training data is comparatively very small. Compared against SVM based baselines, neural network based models seem to be slightly better as the best performing model outperforms the best performing SVM based model in F1 on all but one dataset (WS-L). This is likely because that the neural networks managed to capture abstract features that are useful for learning but cannot be modelled in SVM.

CNN+LSTM Results of our different CNN+LSTM models also show that the model learns better with the pre-trained word embeddings, where the best F1 is achieved with emb-ggl1 on 5 datasets and with emb-ggl2 on 2 datasets. Compared against baselines, our best performing model achieves the highest F1 on 6 datasets (an improvement in F1 between 0.2 and 4.5 on the 'best against best' basis) but WZ-L, on which it is close to the SVM_{fs} model. In particular, when compared against the baseline neural network models on the same word embedding choice, the proposed CNN+LSTM model has predominantly gained better F1, suggesting that the addition of the drop-out and the global max pooling layers is helpful for learning.

Compared against state of the art, it is interesting to note that the proposed baselines are very strong, particularly the feature selected SVM models and the baseline neural network models, which obtain higher than the previously reported figures on 5 datasets (except WZ-S.exp and WZ-LS). Our CNN+LSTM models extend the strength by winning on 6 datasets in F1 with an improvement of: 7 over [34] on WZ-L, 0.2 over [33] on WZ-S.exp, 7.5 over [33] on WZ-S.exp, 13.8 over [10] on WZ-S.gb, 7.6 over [7] on DT, and 6.2 over [7] on RM. On the WS-LS dataset, our best model outperforms [27] on both their word-only and character-only CNN models, losing 0.9 point to their hybrid model that combines both word and character features. Similarly, [10] also used character features in their CNN model. As discussed before, the literature generally acknowledges that character-based features are more effective than word-based. Hence taking into account the above results, we argue that the better results obtained by our CNN+LSTM models using only word-based features is due to the superiority in the network architecture. In other words, stacking CNN and LSTM together with drop-out and pooling has led to better feature extraction that are helpful for learning.

5.5 Impact of feature selection on SVM

Our results of SVM based models showed that feature selection can significantly enhance the model learning accuracy. Since this is the first work that conducts this comparison, we believe that it is beneficial to undertake further analysis to understand the impact of feature selection in classic methods.

To do so, we analyse: 1) the amount of original features retained by the feature selection algorithm; and 2) the distribution of different feature types in the selected features.

Table 9 shows the retained features after feature selection as percentage of the number of original features for the Basic and Enhanced feature sets. The figures indicate that the original features are drastically discarded after feature selection. And as shown before, on most occasions, this significantly improved learning accuracy.

Dataset	Retained features (Basic)	Retained features (Enhanced)
WZ-L	5.1%	5.1%
WZ-S.amt	3.4%	3.1%
WZ-S.exp	3.9%	3.9%
WZ-S.gb	3.4%	3.2%
WZ-LS	4.4%	4.0%
DT	4.4%	3.8%
RM	0.7%	0.6%

Table 9: Retained features as a percentage of the number of original features (each of the Basic and Enhanced feature sets) after applying feature selection to each dataset.

Tables 10 and 11 further explore this by analysing the distribution of selected features across different datasets. For discrete feature types such as word n-grams, we count the number of selected features of that type and calculate its percentage over the number of all selected features. For continuous feature types that have only one value per type such as the number of words (#words), we show the number of datasets on which it is selected (e.g., 2/7).

Feature type	Distribution across datasets
word n-gram	21~81% (avg. 51%) of all features, 7/7 datasets
pos n-gram	18~78% (avg. 49%) of all features, 7/7 datasets
#char	1/7 dataset
sntm	2/7
#syllb	1/7
#word	2/7

Table 10: Selected features from the Basic feature set and their distribution across datasets.

Feature type	Distribution across datasets
word n-gram	23~64% (avg. 47%) of all features, 7/7 datasets
pos n-gram	9~70% (avg. 40%) of all features, 7/7 datasets
hashtag	3~18% (avg. 11%) of all features, 7/7 datasets
CAP	1/7 dataset
#char	4/7
FRE	1/7
#hashtags	1/7
sntm	1/7
#syllb	1/7
#word	1/7

Table 11: Selected features from the Enhanced feature set and their distribution across datasets. Features that are additional to the Basic set are highlighted in bold.

Using Table 10 for example, only 6 out of the 10 feature types from the Basic feature set are used on at least one dataset. The completely discarded features include #mentions, #hashtags, FRE, and FKGL. Out of the 6 feature types, word and PoS n-grams are most indicative as they are selected on all datasets. Other feature types appear to be only useful on isolated cases (i.e., 1 or 2 datasets). Similar situation is found for the Enhanced feature set, with only 2 out of the 5 **added** feature types are selected on at least one dataset. The hashtag features are also used on all datasets, suggesting that they can be quite useful for the task. For both feature sets, word and PoS n-grams remain to be dominant.

Our results raise an arguably controversial question that is whether the practice of feature engineering found to be fundamental to classic methods is really worthwhile. As it appears that with basic features such as word and PoS n-grams combined with feature selection, the systems perform just as well or even better than sophisticated sets of features without feature selection, and on several occasions, achieving close-to-best F1 among all models. Notice for example, that on the WZ-L dataset where the SVM_{fs} model achieved the best results (Table 2), all feature types are discarded but word and PoS n-grams.

6 CONCLUSION AND FUTURE WORK

The propagation of hate speech on social media has been increasing significantly in recent years. Despite substantial effort from law enforcement departments, legislative bodies as well as millions of investment from social media companies, it is widely recognised that effective counter-measures rely on automated data mining techniques. This work makes several contributions to this problem. First, we introduced a method for automatically classifying hate speech using a deep neural network model combining CNN, LSTM with drop-out and pooling that are found to empirically improve classification accuracy. Second, we conducted comparative evaluation on the largest collection of public datasets and show that the proposed method outperformed baselines as well as state of the art in most cases. Our results make new reference for future comparative studies. Third, we created and published another hate speech dataset, complementing existing ones by focusing on religion (Muslim) and refugees. Finally, we showed that for classic methods that depend on pre-engineered features, there is often no need for sophisticated feature engineering activity; but using automatic feature selection techniques on generic features such as as n-grams can in fact, produce better results.

We will explore future work in the following directions: 1) to further fine tune our neural network to consider, e.g., stacking multiple convolutional layers which are good for extracting hierarchical features; 2) to integrate user-centric features, such as the frequency of a user detected for posting hate speech and the user's interaction with others; and 3) to study and quantify the difference between hate speech detection and other related tasks such as offensive language, and cyber bullying.

Acknowledgment Placeholder

REFERENCES

[1] I. Awan and I. Zempi. Last accessed: May 2017. Jo Cox 'deserved to die': Cyber Hate Speech Unleashed on Twitter. (Last accessed: May 2017). http://hopenothate.org.uk/cyber-hate/

[2] 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)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 759–760. https://doi.org/10.1145/3041021.3054223

[3] Pete Burnap and Matthew L. Williams. 2015. Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. *Policy and Internet* 7, 2 (2015), 223–242.

[4] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting Offensive Language in Social Media to Protect Adolescent Online Safety. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust (SOCIALCOM-PASSAT '12)*. IEEE Computer Society, Washington, DC, USA, 71–80. https://doi.org/10.1109/SocialCom-PASSAT.2012.55

[5] Jason P.C. Chiu and Eric Nichols. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4 (2016), 357–370.

[6] Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving Cyberbullying Detection with User Context. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR'13)*. Springer-Verlag, Berlin, Heidelberg, 693–696. https://doi.org/10.1007/978-3-642-36973-5_62

[7] Thoams Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the THE 11TH INTERNATIONAL AAAI CONFERENCE ON WEB AND SOCIAL MEDIA*. Association for the Advancement of Artificial Intelligence (AAAI).

[8] Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Common Sense Reasoning for Detection, Prevention, and Mitigation of Cyberbullying. *ACM Trans. Interact. Intell. Syst.* 2, 3, Article 18 (Sept. 2012), 30 pages. https://doi.org/10.1145/2362394.2362400

[9] 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)*. ACM, New York, NY, USA, 29–30. https://doi.org/10.1145/2740908.2742760

[10] Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-Speech. In *Proceedings of the First Workshop on Abusive Language Online*. Association for Computational Linguistics, Vancouver, Canada, 85–90.

[11] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering* 10, 10 (2015), 215–230.

[12] Edel Greevy and Alan F. Smeaton. 2004. Classifying Racist Texts Using a Support Vector Machine. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*. ACM, New York, NY, USA, 468–469. https://doi.org/10.1145/1008992.1009074

[13] The Guardian. Last accessed: July 2017. Anti-Muslim hate crime surges after Manchester and London Bridge attacks. (Last accessed: July 2017). https://www.theguardian.com/society/2017/jun/20/anti-muslim-hate-surges-after-manchester-and-london-bridge-attacks

[14] The Guardian. Last accessed: July 2017. Zuckerberg on refugee crisis: 'Hate speech has no place on Facebook'. (Last accessed: July 2017). https://www.theguardian.com/technology/2016/feb/26/mark-zuckerberg-hate-speech-germany-facebook-refugee-crisis

[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980* (2014).

[16] Irene Kwok and Yuzhou Wang. 2013. Locate the Hate: Detecting Tweets Against Blacks. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI'13)*. AAAI Press, 1621–1622. http://dl.acm.org/citation.cfm?id=2891460.2891697

[17] Qingshan Liu, Feng Zhou, Renlong Hang, and Xiaotong Yuan. 2017. Bidirectional-Convolutional LSTM Based Spectral-Spatial Feature Learning for Hyperspectral Image Classification. *CoRR* abs/1703.07910 (2017). http://arxiv.org/abs/1703.07910

[18] Natasha Lomas. Last accessed: July 2017. Facebook, google, twitter commit to hate speech action in Germany. (Last accessed: July 2017). http://techcrunch.com/2015/12/16/germany-fights-hate-speech-on-social-media/

[19] Yashar Mehdad and Joel Tetreault. 2016. Do Characters Abuse More Than Words?. In *Proceedings of the SIGDIAL 2016 Conference*. Association for Computational Linguistics, Los Angeles, USA, 299–303.

[20] BBC News. Last accessed: July 2017. Countering hate speech online. (Last accessed: July 2017). http://eeagrants.org/News/2012/Countering-hate-speech-online

[21] BBC News. Last accessed: May 2017. Finsbury Park attack: Son of hire boss held over Facebook post. (Last accessed: May 2017). http://www.bbc.co.uk/news/uk-wales-40347813

[22] 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)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 145–153. https://doi.org/10.1145/2872427.2883062

[23] John T. Nockleby. 2000. *Hate Speech*. Macmillan, New York, 1277–1279.

[24] A. Okeowo. Last accessed: July 2017. Hate on the Rise After Trump's Election. (Last accessed: July 2017). http://www.newyorker.com/news/news-desk/hate-on-the-rise-after-trumps-election

[25] A. Oksanen, J. Hawdon, E. Holkeri, M. Nasi, and P. Rasanen. 2014. *Exposure to online hate among young social media users*. Emerald, Bingley, UK, 253–273.

[26] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016).

[27] Jo Ho Park and Pascale Fung. 2017. One-step and Two-step Classification for Abusive Language Detection on Twitter. In *ALW1: 1st Workshop on Abusive Language Online*. Association for Computational Linguistics, Vancouver, Canada.

[28] Anna Schmidt and Michael Wiegand. 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*. Association for Computational Linguistics, Valencia, Spain, 1–10.

[29] Ellen Spertus. 1997. Smokey: Automatic Recognition of Hostile Messages. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence (AAAI'97/IAAI'97)*. AAAI Press, 1058–1065. http://dl.acm.org/citation.cfm?id=1867406.1867616

[30] Eleni Tsironi, Pablo Barros, Cornelius Weber, and Stefan Wermter. 2017. An Analysis of Convolutional Long Short-Term Memory Recurrent Neural Networks for Gesture Recognition. *Neurocomput.* 268, C (Dec. 2017), 76–86. https://doi.org/10.1016/j.neucom.2016.12.088

[31] 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*. 86–95.

[32] 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 (LSM '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 19–26. http://dl.acm.org/citation.cfm?id=2390374.2390377

[33] 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*. Association for Computational Linguistics, Austin, Texas, 138–142. http://aclweb.org/anthology/W16-5618

[34] 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*. Association for Computational Linguistics, San Diego, California, 88–93. http://www.aclweb.org/anthology/N16-2013

[35] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 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 (CIKM '12)*. ACM, New York, NY, USA, 1980–1984. https://doi.org/10.1145/2396761.2398556

[36] Shuhan Yuan, Xintao Wu, and Yang Xiang. 2016. A Two Phase Deep Learning Model for Identifying Discrimination from Tweets. In *Proceedings of 19th International Conference on Extending Database Technology*. 696–697.

[37] Haoti Zhong, Hao Li, Anna Squicciarini, Sarah Rajtmajer, Christopher Griffin, David Miller, and Cornelia Caragea. 2016. Content-driven Detection of Cyberbullying on the Instagram Social Network. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 3952–3958. http://dl.acm.org/citation.cfm?id=3061053.3061172