Semantic Web 1 (0) 1–5
IOS Press

Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter

Ziqi Zhang *
Information School, University of Sheffield, Regent Court, 211 Portobello, Sheffield, S1 4DP, UK
E-mail: ziqi.zhang@sheffield.ac.uk
Lei Luo
College of Pharmaceutical Science, Southwest University, Chongqing, 400716, China
E-mail: drluolei@swu.edu.cn

Editors: First Editor, University or Company name, Country; Second Editor, University or Company name, Country
Solicited reviews: First Solicited Reviewer, University or Company name, Country; Second Solicited Reviewer, University or Company name, Country
Open reviews: First Open Reviewer, University or Company name, Country; Second Open Reviewer, University or Company name, Country

Abstract. In recent years, the increasing propagation of hate speech on social media and the urgent need for effective countermeasures have drawn significant investment from governments, companies, and empirical research. Despite a large number of emerging, scientific studies to address the problem, the performance of existing automated methods at identifying specific types of hate speech - as opposed to identifying non-hate - is still very unsatisfactory, and the reasons behind are poorly understood. This work undertakes the first in-depth analysis towards this problem and shows that, the very challenging nature of identifying hate speech on the social media is largely due to the extremely unbalanced presence of real hateful content in the typical datasets, and the lack of unique, discriminative features in such content, both causing them to reside in the 'long tail' of a dataset that is difficult to discover. To address this issue, we propose novel Deep Neural Network structures serving as effective feature extractors, and explore the usage of background information in the form of different word embeddings pre-trained from unlabelled corpora. We empirically evaluate our methods on the largest collection of hate speech datasets based on Twitter, and show that our methods can significantly outperform state of the art, as they are able to obtain a maximum improvement of between 4 and 16 percentage points (macro-average F1) depending on datasets.

Keywords: hate speech, classification, neural network, CNN, GRU, skipped CNN, deep learning, natural language processing

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, 5]. 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 characteristics such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics' [32]. 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 [21]. This cor-

*Correspondence author

1570-0844/0-1900/$35.00 © 0 – IOS Press and the authors. All rights reserved

relates to record spikes of hate crimes [34, 44], and cases of threats to public safety due to its nature of inciting hate crimes, such as that followed the Finsbury van attack [4]. In the EU, surveys and reports focusing on young people in the EEA region show rising hate speech and related crimes based on religious beliefs, ethnicity, sexual orientation or gender, as 80% of respondents have encountered hate speech online and 40% felt attacked or threatened [3]. Statistics also show that in the US, hate speech and crime is on the rise since the Trump election [33]. The urgency of this matter has been increasingly recognised, as a range of international initiatives have been launched towards the qualification of the problems and the development of counter-measures. These include, e.g., the UNESCO's research on countering online hate speech [15], and the Media Against Hate campaign[1] led by the European Federation of Journalists and a coalition of civil society organisations to join efforts on countering hate speech and discrimination in the media.

Building effective counter measures for online hate speech requires identifying and tracking hate speech online, which is the first step towards the quantification and qualification of the problem. For years, social media companies such as Twitter, Facebook, and YouTube have been investing hundreds of millions of euros every year on this task [16, 22, 26], but are still being criticised for not doing enough. This is largely because such efforts are primarily based on manual content review to identify and delete offensive materials. The process is labour intensive, time consuming, and not sustainable or scalable in reality [7, 16, 44].

This work studies automatic, scalable methods of hate speech detection in the social media domain using Twitter as a data source. It employs semantic content analysis techniques borrowed from the Natural Language Processing (NLP) and Machine Learning (ML) research, both of which are core pillars of the Semantic Web research. Despite a plethora of studies conducted for hate speech detection from the social media, results still show that the task - particularly detecting specific types of hate, e.g., religion - remains very challenging [6].

This work explores **two research questions** concerning this problem: 1) what makes the detection of hate speech difficult, and 2) what are the methods that help address it. We undertake research which makes three major contributions to the literature. **First**, we make the first in-depth analysis to qualify the characteristics of such content on the social media and we show that hate speech exhibits a 'long tail' pattern compared to non-hate content due to their lack of unique, discriminative features, and this makes them very challenging to identify. **Second**, using a Deep Neural Network (DNN) based classification model as a solution, we propose and experiment with three methods to tackle the challenge. These include: improving the network architecture by adding layers serving as 1) skip-gram like and 2) orderly feature extractors, to extract features that empirically prove to be very effective at identifying hate speech in the long tail. Both contribute to novel DNN architectures for the task of hate speech detection; and 3) exploring the effect of using background information from large unlabelled corpora in the form of different pre-trained word embeddings on the task. **Finally**, evaluated on the largest collection of Twitter datasets, we show that our proposed DNN based methods can outperform state of the art methods on all datasets by up to 13 percentage points in macro-average F1. More importantly, on the more challenging task of identifying specific types of hate Tweets (i.e., excluding non-hate), the best of our methods can obtain an improvement of $\geqslant 4$ points on all datasets, and $\geqslant 10$ points (with a maximum attainable of 16 points) on four datasets over state of the art. Our thorough evaluation on all currently available public Twitter datasets sets new benchmark for future research in this area. And our findings encourage future work to take a renewed perspective, i.e., to consider the challenging case of long tail.

The remainder of this paper is structured as follows. Section 2 reviews related work on hate speech detection and other relevant fields; Section 3 describes our data analysis to understand the challenges of hate speech detection on Twitter; Section 4 introduces a set of methods we use to tackle the task; Section 5 presents experiments and results; and finally Section 6 concludes this work and discusses future work.

2. Related Work

2.1. Terminology and Scope

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 of-

[1] http://europeanjournalists.org/mediaagainsthate/, last accessed: 12 Feb 2018

ten seen to co-exist or become mixed with other terms such as 'offensive', 'profane', and 'abusive languages', and 'cyberbullying'. To distinguish them, we identify that hate speech 1) targets individual or groups on the basis of their characteristics (to be referred to as **types of hate** or **hate classes**); 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 [31]. `'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[2]. Our methods and experiments will only address hate speech, due to both dataset availability and the goal of this work. In addition, the methods we have proposed in this work relate to many general problems in the areas of NLP and ML, such as the design of network structures in DNN, and the use of word embeddings in NLP. We will briefly compare against state of the art in these areas when each of our methods are introduced in Section 4.

2.2. Methods of Hate Speech Detection and Related Problems

Existing methods primarily cast the problem as a supervised document classification task [38]. These 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 [5, 12, 14, 20, 24, 28, 42–46] (**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 [16, 31, 36, 41] (**deep learning methods**).

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

Schmidt et al. [38] 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 [5, 6, 12, 20, 24, 41–44], as well as other related tasks such as the detection of offensive and abusive content [7, 28, 31], discrimination in Tweets [46], and cyberbullying on Instagram [47]. Recent research [28] 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., `'ki11 yrslef'`). Other surface features can include URL mentions, hashtags, punctuations, word and document lengths, capitalisation, etc [7, 12, 31]. *Word generalisation* includes examples where word clusters were used for detecting hate speech in Yahoo! group comments [42], Xiang et al. [45] and Zhong et al. [47] who used topic modelling in offensive Tweets and cyberbullying detection respectively, and word embedding learning [14, 31, 41, 46] that learns 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 [5, 12, 18, 41]. *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 [5, 18, 31, 45]. It is worth to note that early methods such as Spertus et al. [39] are heavily based on lexical resources. However it has been shown that such features alone are not very effective [7]. *Linguistic features* utilise syntactic information such as Part of Speech (PoS) and certain dependency relations as features [5, 7, 12, 18, 47]. For example, Burnap et al. [5] noted that 'othering phrases' denoting a 'we v.s. them' stance are common in hate speech, while Chen et al. [7] and Zhong et al. [47] used dependency relations as features for detecting offensive language and cyberbullying. *Meta-information* refers to data about messages, such as gender identity of a user associated with a message [43, 44], or high frequency of profane words in a user's post history [11, 45]. In addition, *Knowledge-Based features* such as messages mapped to stereotypical concepts in a knowledge base [13] and *multimodal*

[2]We will indicate explicitly where works address a related problem rather than hate speech.

information such as image captions and pixel features [47] were used in cyberbullying detection but only in very confined context [38].

In terms of classifiers, existing methods are predominantly supervised. Among these, Support Vector Machines (SVM) is the most popular algorithm [5, 7, 12, 20, 28, 42, 45, 46], while other algorithms such as Naive Bayes [7, 12, 24, 28, 46], Logistic Regression [12, 14, 28, 43, 44], and Random Forest [12, 45] are also used.

Deep learning based methods employ deep artificial neural networks 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 multilayer 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 structure, which is carefully designed to automatically extract useful features from a simple input feature representation. Note that this categorisation excludes those methods [14, 28, 46] that used 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 belong to this category include [2, 16, 36, 41], all of which used simple word and/or character based one-hot encoding as input features to their models, while Vigna et al. [41] also used word polarity. The most popular network architectures are Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), typically Long Short-Term Memory network (LSTM). In the literature, CNN is well known as an effective network to act as 'feature extractors', whereas RNN is good for modelling orderly sequence learning problems [35]. In the context of hate speech classification, intuitively, CNN extracts word or character combinations [2, 16, 36] (e.g., phrases, n-grams), RNN learns word or character dependencies (orderly information) in Tweets [2, 41]. Compared against classic methods, recent research has obtained better results with DNN based approaches [2, 16, 36].

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 [38]. The large majority of existing works were evaluated on privately collected datasets, often for different problems. Nobata et al. [31] 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 Mehdad et al [28]. 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 [12, 16, 36, 43, 44]. Waseem et al. [44] annotated about 17,000 Tweets, split into three classes based on the type of hate including 'sexist', 'racist' and 'non-hate'. 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 Waseem et al. [43], where about 7,000 Tweets were collected with roughly 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 were either feminist or anti-racism activist; and amateurs that were crowdsourced. Experiments showed that amateur annotators were 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 Gamback et al. [16], the authors merged both expert and amateur annotations in this dataset by using majority vote, giving expert annotations double weight; and in Park et al. [36], the dataset by [44] was merged with the expert annotations in [43] to create a single dataset. Davidson et al. [12] 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 was a challenging task, as hate speech does not always contain offensive words while offensive language does not always express hate.

2.4. Performance of Hate Speech Detection Methods

Evaluating the performance of hate speech (and also other related content) detection typically adopts

the classic Precision, Recall and F1 metrics. Precision measures the percentage of true positives among the set of hate speech messages identified by a system; Recall measures the percentage of true positives among the set of real hate speech messages we expect the system to capture (also called '**ground truth**' or '**gold standard**'), and F1 calculates the harmonic mean of the two. The three metrics are usually applied to each class in a dataset, and often an aggregated figure is computed either using **micro-average** or **macro-average**. The first sums up the individual true positives, false positives, and false negatives identified by a system for different classes to calculate overall Precision, Recall and F1 scores. The second take the average of the Precision, Recall and F1 on different classes.

Existing studies on hate speech detection have primarily reported their results using micro-average Precision, Recall and F1 [2, 16, 36, 43, 44]. The problem with this is that in an unbalanced dataset where instances of one class (to be called the 'dominant class') significantly out-number others (to be called 'minority classes'), micro-averaging can mask the poor performance on minority classes. As we will show in Section 3, hate speech detection is a typical task dealing with extremely unbalanced datasets, where real hate messages only account for a very small percentage of the entire dataset, while the large majority are non-hate but exhibits similar lexical-syntactic patterns to hate messages. From a limited number of studies that report performance on a per-class basis [6, 12, 36], it is apparent that our ability to automatically detect real hate content and classify their types is still very unsatisfactory. For example, Burnap et al. [6] reported a F1 of 98% on classifying non-hate Tweets, but only between 18 and 49% on classifying hate Tweets of sexual orientation. Our experiments later in Section 5 also confirmed this.

Considering the goal of hate speech detection and its implications on developing effective counter-measures, it is reasonable to argue that identifying hate content and the types of hate is far more important than identifying non-hate. Unfortunately, state of the art results show that this still remains a major, unsolved problem.

2.5. Remark

Recognising the challenges of hate speech detection on the social media, this work makes a first effort to quantify and qualify the characteristics of the typical data involved in such a task. We show that the difficulty of identifying hate-content from non-hate is due to the lack of unique, discriminative features in individual instances, which as a result, appear in the long tail of the entire dataset. To address this very challenging task, we explore different directions of research and contribute several methods including developing DNN architectures that are able to extract richer features from the dataset, and employing background information from different, large unlabelled corpora in the form of feature vectors pre-trained on such data.

3. Dataset Analysis - the Case of Long Tail

In this section, we describe the typical datasets used in the studies of hate speech detection on Twitter, and analyse their characteristics to understand the challenges of the task.

3.1. Public Twitter Datasets

As introduced before, the only publicly available hate speech datasets include that of Davidson et al. [12] and Waseem et al. [43, 44], which were later used to create other variants [16, 36]. Davidson et al. [12] classified hate speech in general without identifying the types of hate. The dataset also contains Tweets annotated as 'abusive (but non-hate)'. In this work, we set such annotations to be 'non-hate' so the dataset contains only two classes. We refer to this dataset as **DT**. The dataset created in Waseem et al. [44] has three classes: 'sexism', 'racism', and 'non-hate', and will be referred to as **WZ**. The smaller dataset [44] created later by the same authors added a fourth class 'both', to include Tweets that are both sexism and racism. And as described before in Section 2, it was annotated by two groups of annotators. We will use **WZ-S.amt** to denote the dataset annotated by amateurs, and **WZ-S.exp** to denote the dataset annotated by experts. The authors showed that the two sets of annotations were different as the supervised classifiers obtained different results on them. In Gamback et al. [16], 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, Park et al. [36] combined the WZ and the WZ-S.exp datasets into a single dataset and in case of duplicates, we take the annotation from WZ. We refer to this dataset as **WZ.pj**. 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.

In this work, we also 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, 4, 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 [44] for annotation.

To collect the data, we follow the mainstream bootstrapping approach [44] that starts with an initial search for Tweets containing common slurs and terms used pertaining to different types of hate, 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[3]. We will refer to this dataset as **RM**.

The above process creates a total of 7 publicly available Twitter datasets for hate speech, the statistics of which are as shown in Table 1. To our knowledge, this is by far the most comprehensive collection of Twitter hate speech datasets used in any studies.

Dataset	**#Tweets**	**Classes** (%)
WZ	16,093	racism (12%) sexism (19.6%) neither (68.4%)
WZ-S.amt	6,594	racism (1.8%) sexism (16.3%) both (0.2%) neither (81.6%)
WZ-S.exp	6,594	racism (1.3%) sexism (11.8%) both (0.5%) neither (86.4%)
WZ-S.gb	6,594	racism (1.4%) sexism (13.8%) both (0.4%) neither (84.4%)
WZ.pj	18,625	racism (10.8%) sexism (20.2%) both (0.2%) neither (68.8%)
DT	24,783	hate (5.8%) non-hate (94.2%)
RM	2,435	hate (17%) non-hate (83%)

Table 1
Statistics of datasets used in the experiment

3.2. Dataset Analysis

As shown in Table 1, all datasets are significantly biased towards non-hate, as hate Tweets account between only 5.8% (DT) and 31.6% (WZ). When we inspect specific types of hate, some can be even more scarce, such as 'racism' and the extreme case of 'both' (between only 15 and 35 instances) in the three WZ-S.x datasets. The implication is that compared to non-hate, the training data for hate Tweets are very scarce.

Adding to the problem is the lack of unique, discriminative features in hate Tweets. To quantify this, we measure the '**level of uniqueness**' of the features for 1) each class in a dataset in general, and 2) each instance of each class in a dataset. However, as introduced before, real features used for hate speech detection can cover a wide range of different types which are infeasible to measure extensively. Here we propose

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

to analyse the word stems[4] as a proxy to the problem. Since most types of features are derived from words, we argue that this is a reasonable approximation.

For 1), Let $words(c_i)$ denote the set of different word stems (to be simply referred to as 'words' in the following) of all Tweets belong to the class c_i in a dataset, we define **Unique words of Class (UoC)**:

$$UoC(c_i) = words(c_i) - \bigcup_{c_j, i \neq j} words(c_j) \quad (1)$$

as the set of different words unique to that class (i.e., they are found *only* in that class and not in other classes), then we calculate for each dataset:

$$U2C(c_i) = \frac{|UoC(c_i)|}{|words(c_i)|} \quad (2)$$

, where **U2C** stands for **Unique words to Class** ratio that divides the number of different class-unique words by the number of all different words in that class (i.e., the words can be present in other classes at the same time). Intuitively, if a class has a lot of words that are unique to itself, it may have many class-unique features that make it easier to classify. We show the statistics for each class from each dataset in Table 2.

	Racism	Sexism	Both	Non-hate	Hate
WZ-S.amt	0.15	0.29	0.10	0.65	-
WZ-S.exp	0.21	0.26	0.16	0.72	-
WZ-S.gb	0.20	0.28	0.14	0.70	-
WZ.pj	0.15	0.27	0.10	0.55	-
WZ	0.16	0.27	-	0.55	-
DT	-	-	-	0.82	0.15
RM	-	-	-	0.78	0.27

Table 2
Analysis of Uwords-2-Class (U2C) ratios.

For 2), let t_m denote a Tweet in a dataset, $l(t_m) = c_i$ returns the class label of t_m, and $words(t_m)$ returns the set of different words from t_m, then for each dataset, we measure for each Tweet:

$$TU2C(t_m) = \frac{|words(t_m) \cap UoC(l(t_m))|}{|words(t_m)|} \quad (3)$$

[4]Tweets are pre-processed then stems are extracted. See Section 4.1 for details.

, where **TU2C** stands for **Unique words to Class in Tweet** ratio that measures the fraction (i.e., within [0, 1.0]) of different class-unique words of a Tweet subject to the class of this Tweet. Intuitively, the TU2C score can be considered as a measure of 'uniqueness' of the features found in a Tweet. A high value indicates that the Tweet can potentially contain more features that are unique to its class, and as a result, we can expect the Tweet to be relatively easy to classify. On the other hand, a low value indicates that many features of this Tweet are potentially non-discriminative as they may also be found across multiple classes, and therefore, we can expect the Tweet to be relatively difficult to classify. In Figure 1 we plot for each dataset, the distribution of Tweets across all classes in that dataset within different sub-ranges of the TU2C scores.

Findings. Table 2 shows that compared to any types of 'hate' Tweets, 'non-hate' has a much higher percentage of words (roughly between 3 and 6 times more) that are unique to that class and therefore, we can expect it to have much more class-unique features. Figure 1 reveals at instance level that the majority of hate Tweets potentially lack discriminative features and as a result, they 'sit in the long tail' of the dataset as ranked by the feature uniqueness of Tweets. For example, on the WZ-S.amt dataset, Table 2 shows that the non-hate Tweets contain four, two and six times more class-unique words than the racism, sexism and 'both' Tweets respectively. Inspecting individual Tweets of these classes in Figure 1, between 40% and 50% of instances of the three 'hate' classes of Tweets are found to have TU2C=0, i.e., they do not contain any class-unique words at all; and the fractions of Tweets with a TU2C$\geqslant$0.4 are only 3%, 2% and 0 for racism, sexism and 'both' classes. In contrast, a significantly larger fraction of non-hate Tweets contain more class-unique words (e.g., 17% of instances with TU2C$\geqslant$0.4). These statistics quantify the potential scarcity of class-unique features of hate Tweets in a typical hate speech detection task, from both a class and instance level. We believe this is the main reason behind the much poorer performance of state of the art on detecting hate than non-hate.

4. Methodology

In this section, we describe three types of methods explored in this work to tackle hate speech detection in the long tailed Twitter datasets. We start with briefly

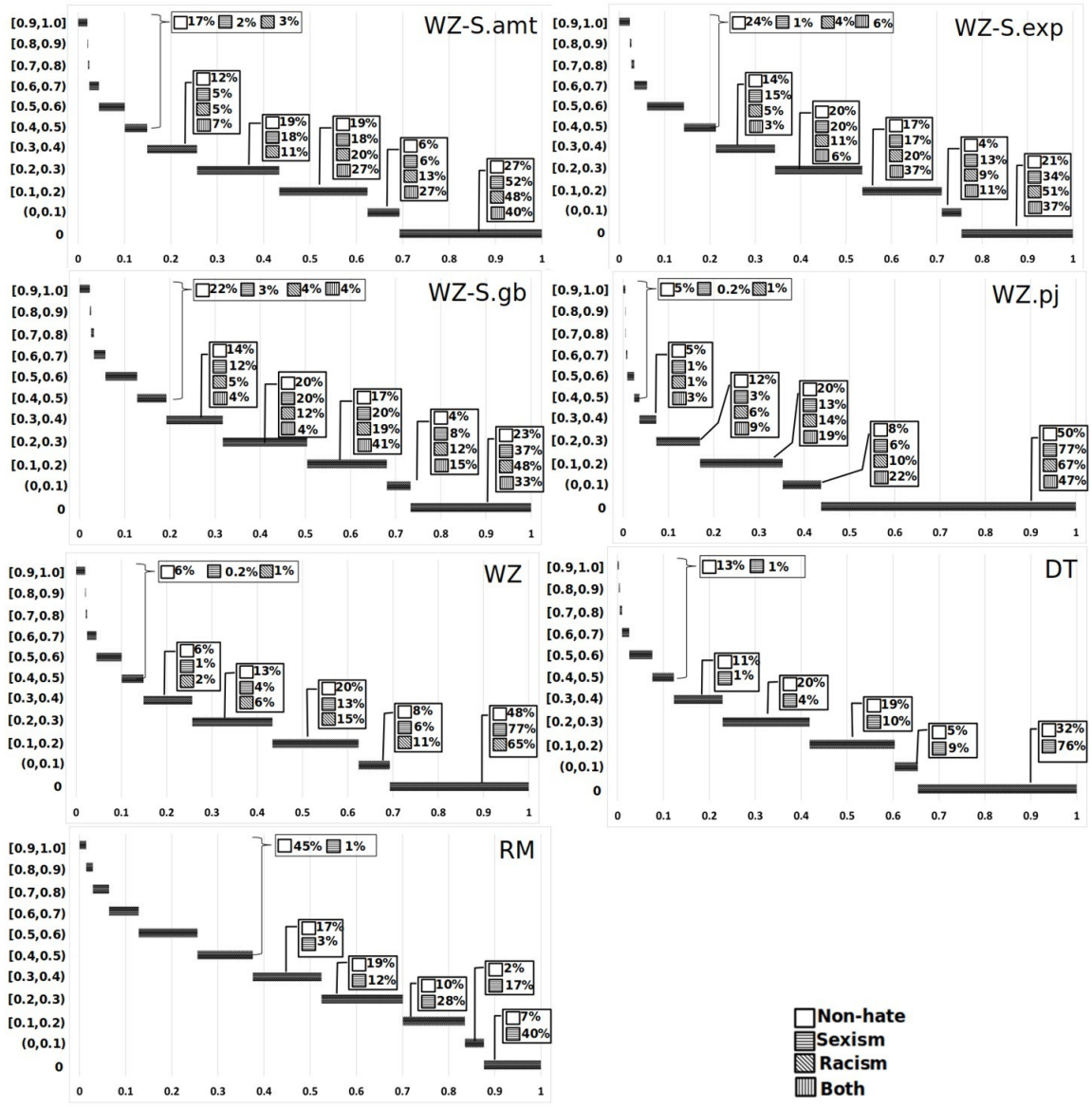

Fig. 1. Distribution of Tweets across all classes in each dataset within different sub-ranges of the TU2C scores. The *x*-axis shows the incremental percentage of a dataset; the *y*-axis shows the sub-ranges of TU2C scores. The call-out boxes show for each class, the fraction of its instances fall under that sub-range (in case a class is not present it has a fraction of 0%). As an example, on the WZ-S.amt dataset, within the $[0.4, 1.0]$ range (as enclosed by the big curly bracket), 17% of non-hate, 2% of sexism, 3% of racism and 0% of 'both' Tweets have a TU2C score within that range.

explaining the data cleaning pre-process (Section 4.1, then discuss in details our proposed DNN structures for better feature extraction (Section 4.2), followed by the study of using different word embeddings learned from large unlabelled datasets (Section 4.3).

4.1. Pre-processing

Given a tweet, we adopt the following light pre-processing procedure to normalise its content.

- remove the following characters: `| : , ; & ! ? \`
- lowercase and stemming, to reduce word inflections
- 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.

We do not use existing tools dedicated for Tweet cleaning such as the GATE framework[5]. While we expect these tools may further improve the data quality and hence contribute to improved classification performance, this is beyond the scope of this work.

4.2. The DNN Structures

As discussed before, the DNN models pre-dominantly used in the hate speech detection literature are based on CNNs and RNNs. We propose two different structures below, both extend a base model (**Base CNN**) that concatenates multiple convolutional layers each using a different window size to extract different features from data. The first adds CNNs using the so-called 'gapped window', which can be considered as extractors of skip-gram like features (**CNN+sCNN**, where 'sCNN' stands for **skipped CNN**). The second adds a Gated Recurrent Units (GRU) layer, a specific type of RNN, to the base model to extract orderly features (**CNN+GRU**).

4.2.1. The Base CNN Model

The Base CNN model is illustrated in Figure 2. The first layer is a word embedding layer, which maps each text message (in generic terms, a 'sequence') into a real vector domain (word embeddings). To do so, we map each word onto a fixed dimensional real valued vector, where each element is the weight for that dimension for that word. The word embeddings can be obtained by learning from unlabelled corpus, or as part of the training process on the annotated hate speech data. We discuss the different options of word embeddings in Section 4.3.

The embedding layer passes an input feature space with a shape of 100×300 to three 1D convolutional layers, each uses 100 filters but different window sizes of 2, 3, and 4 respectively. Intuitively, each CNN layer can be considered as extractors of bi-gram, tri-gram and quad-gram features. The rectified linear unit function is used for activation in these CNNs. The output of these CNNs are concatenated to a single layer, which is then further down-sampled by a 1D max pooling layer with a pool size of 4. The output of max pooling is fed into the final softmax layer to predict probability distribution over all possible classes (n), which will depend on individual datasets.

Comparison with state of the art. The use of multiple, concatenated CNNs has been widely adopted in existing methods. Gamback et al. [16] used the same model structure, but combined both word and character embeddings as input. Park et al. [36] also used the same idea, but with three different window sizes for word embedding input, and another three different window sizes for character embedding input. For these reasons, the Base CNN model can be considered a good representation of state of the art, and an re-implementation of [16] without using character embeddings.

4.2.2. Base CNN + skipped CNN (sCNN)

With this model, we propose to extend the Base CNN model by adding CNNs that use 'gapped window' to extract features from its input, and we call these CNN layers 'skipped CNNs'. A gapped window is one where inputs at certain (consecutive) positions of the window are ignored, such as those shown in Figure 3. We say that these positions within the window are 'deactivated' while other positions are 'activated'. Specifically, given a window of size j, applying a gap of i consecutive positions will produce multiple shapes of size j windows, as illustrated in Algorithm 1.

As an example, applying a 1-gap to a size 4 window will produce two shapes: [O,X,O,O], [O,O,X,O], where 'O' indicates an activated position and 'X' indicates a deactivated position in the window; while applying a 2-gap to a size 4 window will produce a single shape of [O,X,X,O].

[5] https://gate.ac.uk/. Last accessed: February 2018

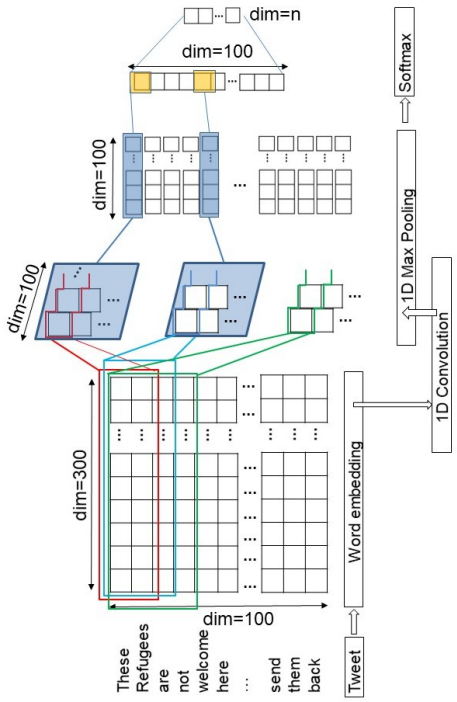

Fig. 2. The Base CNN model uses three different window sizes to extract features. This diagram is best viewed in colour.

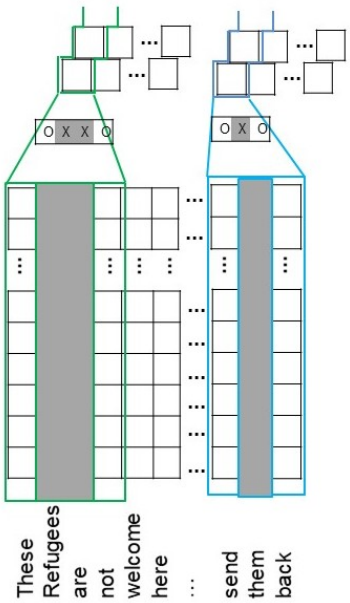

Fig. 3. Example of a 2 gapped size 4 window and a one gapped size 3 window. The 'X' indicates that input for the corresponding position in the window is ignored.

Algorithm 1 Creation of i gapped size j windows. A sequence [O,X,O,O] represents one possible shape of a 1 gapped size 4 window, where the first and the last two positions are activated ('O') and the second position is deactivated ('X').

Input: $i : 0 < i < j,\ j : j > 0,\ w \leftarrow [p_1, ..., p_j]$
Output: $W \leftarrow \emptyset$ a set of j sized window shapes
for all $k \in [2, j)$ and $k \in \mathbb{N}_+$ **do**
  Set p_1 in w to O
  Set p_j in w to O
  for all $x \in [k, k+i]$ and $x \in \mathbb{N}_+$ **do**
    Set p_x to X
    for all $y \in [k+i+1, j)$ and $y \in \mathbb{N}_+$ **do**
      Set p_y in w to O
    end for
    $W \leftarrow W \cup \{w\}$
  end for
end for

To extend the Base CNN model, we add CNNs using 1-gapped size 3 windows, 1-gapped size 4 windows and 2-gapped size 4 windows, keeping the remaining parts of the structure the same. This results in a model illustrated in Figure 4.

Intuitively, the additional CNNs can be considered as extractors of 'skip-gram' features. We expect such features to capture useful clues for detecting hate speech from unbalanced datasets.

Comparison with state of the art. The concept of skip-grams was first introduced in Mikolov et al. [29] and has been extensively used on learning word representations ever since. Such representations are then used in other language-related tasks. This is however, different from directly using skip-gram as features for NLP. To the best of our knowledge, the work by Nguyen et al. [30] pioneered the use of DNN models to extract skip-gram features to be used directly in NLP tasks, i.e., mention detection in this case. And there is no existing work that uses such structures in document classification, especially on very short sentences. Also, strictly speaking, our skipped CNNs do not extract skip-grams as originally defined [29], which would be more expensive to compute[6].

[6]To extract the equivalence of 2-skip quad-gram following the original definition we need to generate 3 different window shapes with a size of 6.

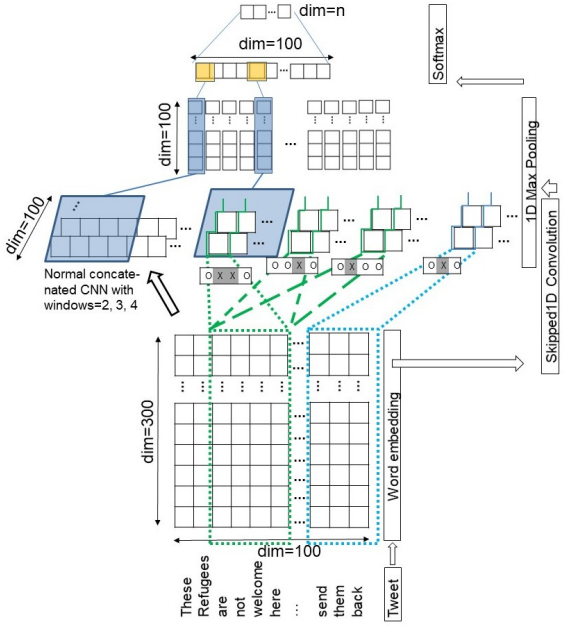

Fig. 4. The Base+sCNN model concatenates features extracted by the normal CNN layers with window sizes of 2, 3, and 4, with features extracted by the four skipped CNN layers. This diagram is best viewed in colour.

4.2.3. *Base CNN + GRU*

With this model, we extend the Base CNN model by adding a GRU layer that takes input from the max pooling layer. This treats the features as timesteps and outputs 100 hidden units per timestep. Compared to LSTM, which is a popular type of RNN, the key difference in a GRU is that it has two gates (reset and update gates) whereas an LSTM has three gates (namely input, output and forget gates). Thus GRU is a simpler structure with fewer parameters to train. In theory, this makes it faster to train and generalise better on small data; while empirically it is shown to achieve comparable results to LSTM [10]. Next, a global max pooling layer 'flattens' the output space by taking the highest value in each timestep dimension, producing a feature vector that is finally fed into the softmax layer. The structure of this model is shown in Figure 7.

Intuitively, the GRU layer captures sequence orders that can be useful for this task. For example, 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 ...'.`

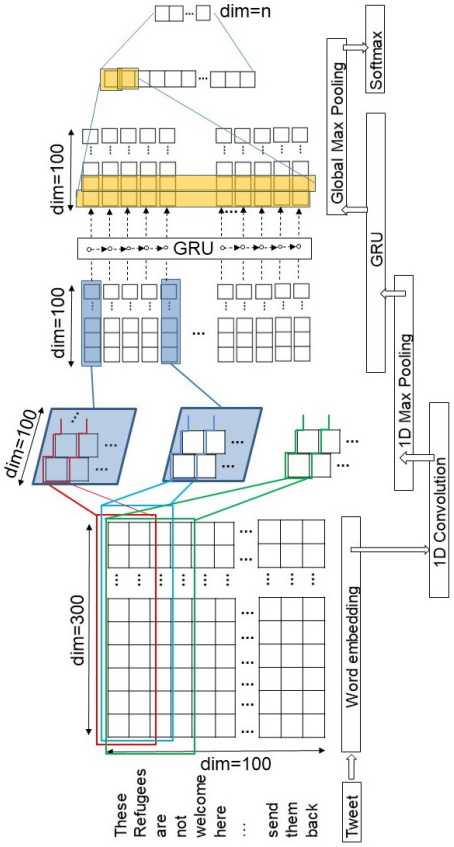

Fig. 5. The CNN+GRU architecture. This diagram is best viewed in colour.

Comparison with state of the art. Our CNN+GRU model is similar to those in [9, 35, 40]. The differences include: 1) we use a GRU instead of an LSTM for the reasons stated before; and 2) we use a global max pooling layer to extract features from the GRU. While previous work used similar models on tasks such as activity and gesture recognition, our work is the first to study classification of very short texts and practically, it remains a question as to what extent such orderly information is present in short texts like Tweets to benefit the classification task.

4.2.4. *Model Parameters*

We use the categorical cross entropy loss function and the Adam optimiser to train the models, as the first is empirically found to be more effective on classification tasks than other commonly used loss functions including classification error and mean squared error

[27], and the second is designed to improve the classic stochastic gradient descent (SGD) optimiser and in theory combines the advantages of two other common extensions of SGD (AdaGrad and RMSProp) [23].

Our choice of parameters described above are largely based on empirical findings reported previously, default values or anecdotal evidence. Arguably, these may not be the best settings for optimal results, which are always data-dependent. However, we show later in experiments that the models already obtain promising results even without extensive data-driven parameter tuning.

4.3. Word embeddings

We also investigate the effect of different word embeddings trained on external corpora on this task.We select three most widely used pre-trained word embeddings, i.e., the Word2Vec embeddings trained on the 3-billion-word Google News corpus with a skip-gram model[7] (**e.w2v**), the 'GloVe' embeddings (**e.glv**) trained on a corpus of 840 billion tokens using a Web crawler [37], and the Twitter embeddings (**e.twt**) trained on 200 million Tweets with spam removed [25][8]. The intuition of using embeddings pre-trained on very large corpora is that we expect them to capture information that is missing from our training data, and hopefully, to some extent encode additional features useful for identifying those long tail cases.

Dataset	e.twt	e.w2v	e.glv	e.all
WZ	4%	13%	7%	3%
WZ-S.amt	3%	11%	5%	2%
WZ-S.exp	3%	11%	5%	2%
WZ-S.amt	3%	11%	5%	2%
WZ.pj	4%	15%	7%	4%
DT	6%	25%	11%	6%
RM	4%	11%	6%	3%

Table 3
Percentage of OOV in each pre-trained embedding model across all datasets.

A potential issue with using pre-trained word embeddings is out-of-vocabulary (OOV) words. Table 3 shows that the three embedding models suffer from different degrees of OOV, with the Twitter embeddings providing the best coverage. We adopt two methods to address this issue. For the first, we use instead, word embeddings learned from the target data on the fly instead of pre-trained embeddings. We initialise the weights in the embedding layer randomly and let our model learn the embeddings from the training data (**e.none**). For the second, we arbitrarily combine (**e.all**) the three pre-trained embeddings in the preference order of e.twt, e.w2v, and e.glv, such that we look up a word in the three embedding models until we find a match.

Following these processes, we obtain a total of five different embedding models to test on the data. All different word embeddings use a dimension of 300.

Comparison with state of the art. Word embeddings have been extensively used in a wide range of downstream NLP tasks since its introduction and their usage in hate speech detection were also previously reported [14, 28, 46]. The evaluation of word embeddings has been the focus in this field, and efforts such as the workshop on evaluating vector space representations for NLP[9] were set up to promote research in this specific area. While existing work in this direction has primarily used intrinsic evaluation by assessing correlation with human judgements of word similarity, it was recently recognised that such kinds of evaluation often fails to predict extrinsic performance, i.e., the performance of downstream NLP tasks that use pre-trained embeddings [8]. Our work makes unique contribution to this problem as our results can be used as a comparison of several state of the art pre-trained word embeddings in the task of hate speech detection from short texts like Tweets. In terms of combining different embeddings, studies such as [17, 19] have used simple concatenation (of word embedding vectors from different models), Principle Component Analysis and re-training using combined corpora, with a goal to improve the performance of tasks using any single embedding model. While our work has a different goal of eliminating OOV and therefore, we do not test these complex methods.

5. Experiment

In this section, we present a series of experiments for evaluation and discuss the results. We firstly explain two baseline methods representing state of the art in Section 5.1, followed by our settings for compar-

[7] https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[8] 'Set1' in [25]

[9] https://sites.google.com/site/repevalacl16/home. Last accessed: February 2018

ison in Section 5.2. We then discuss the general performance of different methods on each dataset in Section 5.3. Next we analyse in details the impact of our proposed methods in capturing hate Tweets in the long tail (Section 5.4).

Performance evaluation metrics. We use the standard Precision (P), Recall (R) and F1 measures for evaluation. However, for the sake of readability, we only present F1 scores in the following sections unless otherwise stated. Detailed Precision and Recall scores can be found in the Appendix. Due to the datasets being very unbalanced and the limitations of micro-averaging as discussed before, we use **macro-average** when calculating average performance across a number of classes in a dataset, unless otherwise stated.

Implementation and cross-fold evaluation. For all methods discussed in this work, we used the Python Keras[10] with Theano backend [11] and the scikit-learn[12] library for implementation[13]. For DNN based methods, we fix the *epoches* to 10 and use a *mini-batch* of 100 on all datasets. These parameters are rather arbitrary and fixed for consistency. We run all experiments in a 5-fold cross validation setting and report the average across all folds.

5.1. Baselines

We use two baseline methods each representing the classic and deep learning based methods used in the literature. **First**, we use the SVM based method described in Davidson et al. [12]. A number of different types of features are used as below:

- Surface features: word unigram, bigram and trigram each weighted by TF-IDF; number of mentions, and hashtags[14]; number of characters, and words;
- Linguistic features: Part-of-Speech (PoS) tag unigrams, bigrams, and trigrams, weighted by their TF-IDF and removing any candidates with a document frequency lower than 5; number of syllables; Flesch-Kincaid Grade Level and Flesch Reading Ease scores that to measure the 'readability' of a document
- Sentiment features: sentiment polarity scores of the tweet, calculated using a public API[15].

Second, we use our Base CNN model described before as another baseline. As discussed, the network structure is the same as that in Gamback et al. [16], and Park et al. [36] except that the latter used different window sizes for CNNs and that both also used character embeddings.

5.2. Comparison Settings

We apply our Base + sCNN and Base + GRU models with each of the five word embedding options, and compare the results against baselines.

5.3. General Results

Range of F1 scores. Table 4 shows the range of F1 obtained by different methods (including baselines) on each class from each dataset. Full details on a per-method basis can be found in the Appendix. The numbers show that, as expected, classifying hate Tweets is a much harder task than non-hate: on the one hand, much lower F1 scores are obtained on hate Tweet classification than non-hate; on the other hand, the wide range of F1 scores on hate Tweet classification also indicate that there is significant difference in the performance of different methods. The poor performance can be seen to largely correlate with the extremely unbalanced training data (see Table 1). For example, on the WZ-S.amt, WZ-S.exp and WZ-S.gb datasets, there are only 13$\sim$33 Tweets belong to the 'both' class, compared to more than 5,000 non-hate. The same observation can be made for racism Tweets on these datasets. As a result, the worst F1 scores are generally found on these two classes.

The extremely unbalanced training data and classifier performance also reinforce our argument that macro-average would be a better option than micro-average when measuring the overall performance of a method across multiple classes on a dataset, as the latter will produce results largely biased towards non-hate.

Base sCNN/GRU v.s. Base CNN models. Figures 6 and 7 compare the average F1 scores obtained by the

[10] https://keras.io/, version 2.0.2

[11] http://deeplearning.net/software/theano/, version 0.9.0

[12] http://scikit-learn.org/, version 0.19.1

[13] Code available at https://github.com/ziqizhang/chase

[14] Extracted from the original tweet before pre-processing which splits hashtags.

[15] https://github.com/cjhutto/vaderSentiment

	Racism	Sexism	Both	Non-hate	Hate
WZ-S.amt	.06∼.4	.66∼.81	0	.88∼.96	-
WZ-S.exp	.23∼.56	.57∼.71	0∼.16	.89∼.95	-
WZ-S.gb	.16∼.58	.66∼.79	0∼.17	.88∼.96	-
WZ.pj	.55∼.72	.54∼.69	0∼.12	.78∼.88	-
WZ	.57∼.74	.53∼.67	-	.78∼.88	-
DT	-	-	-	.89∼.98	.24∼.44
RM	-	-	-	.91∼.94	.61∼.67

Table 4
Range of F1 obtained by different models.

Base+sCNN and Base+GRU models against the two baseline models by showing: 1) the average F1 scores obtained by the two baseline models; and 2) the absolute change in average F1 over the better performing baseline, which is found to be the Base CNN models on all datasets.

Firstly, the Base+sCNN models appear to be able to bring consistent improvement over the Base CNN models, regardless of the word embedding options. In many cases, the improvement can be quite significant, as the maximum attainable improvement in F1 on the seven datasets ranges from 2 (on RM, with e.twt) to 13 (on WZ-S.exp with e.twt) percentage points.

Secondly, the Base+GRU models are also able to bring improvement in F1 in most cases. However, the improvements are much smaller compared to Base+sCNN, as the maximum attainable improvement in F1 ranges from 1 (on WZ.pj, with e.twt/e.glv/e.all) to 7 (on WZ-S.gb with e.twt) percentage points across different datasets. There are also two cases where Base+GRU caused a small decrease in F1 (on WZ.pj and WZ, both with e.none).

These results suggest that the skipped CNNs can be very effective feature extractors for hate speech detection in very short texts such as Tweets. GRU can still extract useful features for the task, but is less promising. This may be due to the short text nature, in which case the really useful orderly features can be sparse.

We also experimented with an architecture that stacks GRU on top of Base+sCNN, which only produced negligible improvements over Base+sCNN, and sometimes minor decrease in performance. Therefore, we do not show the results here. In an analogy, such an architecture expects the GRU layer to extract orderly features from a concatenation of n-gram and skip-grams. However, results seem to suggest that the orderly information from such a concatenation does not appear to be more indicative than the simple presence of the n-grams and skip-grams (i.e., the analogous features extracted by the Base+sCNN models).

Effects of different word embeddings. Comparing results obtained with different word embeddings on each dataset, we cannot observe consistent patterns to determine a 'best performing' option. In terms of the Base CNN models, the best F1 scores are obtained with e.w2v on 3 datasets (WZ-S.amt, WZ-S.exp, RM), with e.none on 2 dataset (WZ-S.gb, WZ.pj), with e.twt on 2 datasets (WZ, DT), with e.glv on 2 datasets (DT, RM), and with e.all on DT. However, with the skipped CNNs or the GRU added, the best performing word embedding on each dataset can also change. For example, on the WZ-S.gb dataset, Base+sCNN obtains the best F1 of 0.60 with e.w2v, while Base CNN obtains the best F1 of 0.54 with e.none.

There is also no strong correlation between the percentage of OOV in each pre-trained word embedding models (See Table 3) and the obtained F1 with that embedding model. For example, despite being the least complete embedding model, e.w2v still contributed to best performing F1 on 3 datasets with Base CNN. On the contrary, despite being the most complete embedding model, e.all only led to one best performing F1 with Base CNN. While e.w2v has the most OOV on the DT dataset, the performance gained with this word embedding model on this dataset is not proportionally worse compared to other more complete word embeddings.

These results are consistent with previous findings that the superiority of one word embedding model is generally non-transferable across tasks, domains, or even datasets. The quality of a word embedding model is possibly more important than its coverage. However, this quality can be relative to different tasks and domains, and very difficult to measure and generalise.

5.4. Effectiveness on Identifying the Long Tail

While the results so far have shown that our enhanced models (i.e., Base CNN+sCNN, Base CNN+ GRU) can obtain better - and in many cases quite significant - average performance across both hate and non-hate classes in the task, it is unclear whether they are particularly effective on identifying hate Tweets, which are typically found in the long tail of such datasets as we have shown before. To understand this, we undertake further two kinds of analyses below.

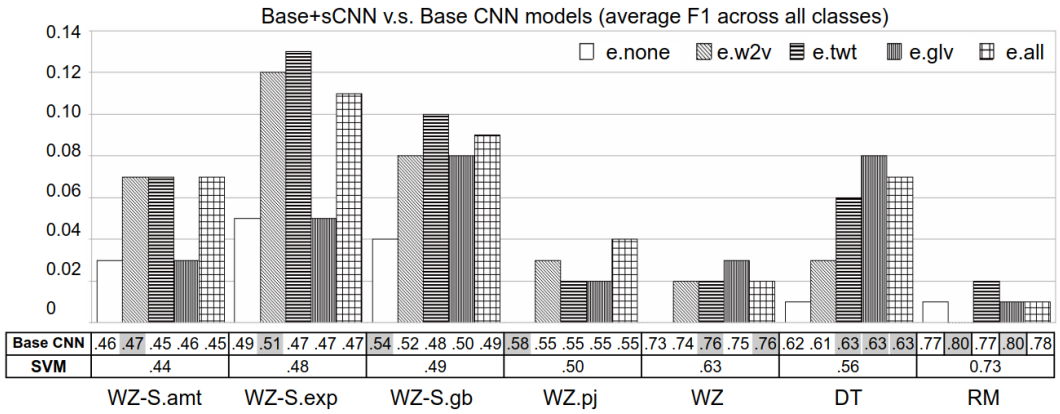

Fig. 6. Comparison of average F1 obtained by Base+sCNN models against the SVM and Base CNN models. **Inside the table**: each column (except the first) corresponds to a separate dataset labelled below. the first row shows results for the Base CNN models. Each column contains 5 scores corresponding to the model using one of the five word embedding options, i.e., 'e.none', 'e.w2v', 'e.twt', 'e.glv' and 'e.all'. The second row shows results for the SVM model. The best F1 on each dataset by the baselines are shaded in grey. **On the chart**: each cluster of (five) columns corresponds to the same dataset represented by the table column below. The height of each column within a cluster indicates the amount of changes in F1 obtained by Base+sCNN over the Base CNN model that is shown immediately below, inside the table. For example, the leftmost white column reads '*Base+sCNN with e.none* improves *Base CNN with e.none* by 0.02 points in F1 (on a scale of [0, 1.0])

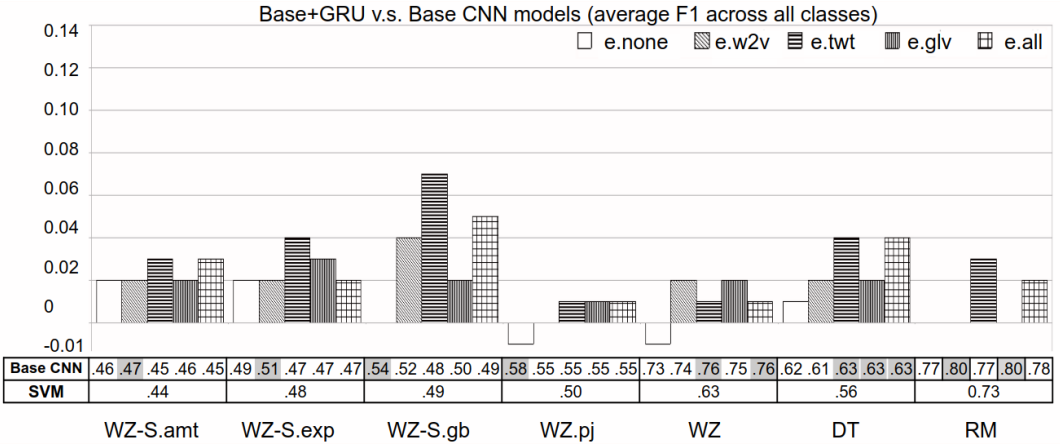

Fig. 7. Comparison of average F1 obtained by Base+GRU models against the SVM and Base CNN models. See Figure 6 caption for an explanation of how to interpret the results.

For the first analysis, on each dataset and for all of our baseline and enhanced models, we calculate and compare their macro-average F1 scores across *hate classes* only. The results are shown in Figures 8 and 9. Comparing these results against those obtained on all classes shown in Figures 6 and 7, it is apparent that the performance gain noticed on our Base+sCNN or Base+GRU models are indeed, largely from their much better performance on recognising hate Tweets. In fact, the improvements in classifying hate Tweets are much more significant than in general. For example, comparing Base+sCNN against Base CNN models, when only hate classes are considered (Figure 8), the maximum improvement obtainable by Base+sCNN models on each datasets are 10 on WZ-S.amt, 13 on WZ-S.exp, 14 on WZ-S.gb, 6 on WZ.pj, 4 on WZ, 16 on DT, and 8 on RM. The Base+sCNN models can achieve $\geqslant 4$ percentage points of improvement on all the seven datasets, and $\geqslant 10$ points on four. When all classes are considered, the maximum obtainable improvements decrease and $\geqslant 10$ points improvements are obtainable on only 2 datasets. Similar but weaker patterns can also be seen when comparing Base+GRU against Base CNN models.

It is also interesting to note that, while one may think that the gains in classifying hate Tweets could, to some extent, be off-traded by loss in classifying non-hate, the fact is that in the majority of cases, the performance in classifying non-hate by these enhanced models remained consistent. On the contrary, we also noticed quite a few cases where the performance of classifying non-hate also benefited by between 1 and 2 percentage points (see Appendix for detailed results). This also makes sense because, as we can see in Figure 1, on many datasets, quite a sizeable fraction of non-hate Tweets are also located in the long tail of the dataset and therefore, could have benefited from the enhanced feature extractors.

For the second analysis, we compare the output of each enhanced model against the output of its corresponding Base CNN model to identify the additional Tweets that are correctly classified by the enhanced model, or in other words, **additional true positives**. Then let T_a denote the set of all such Tweets, we calculate for each Tweet, its $TU2C$ score according to equation 3. As discussed before, this returns a value within the scale [0, 1.0] indicating the level of 'uniqueness' of the features found in that Tweet. We then split the $TU2C$ scores into 11 ranges with an increment of 0.1, and then calculate the percentage of T_a falling under each range. We plot these distributions as heat maps in Figure 10.

In Figure 10, darker colour indicates that a higher percentage of additional true positives identified by an enhanced model (indexed by column) are found to have $TU2C$ scores belong to that range (indexed by row). As it is shown, for all enhanced models on all datasets, there is consistent pattern that the vast majority of additional true positives have low $TU2C$ scores. The pattern is particularly strong on WZ and WZ.pj datasets. The figure shows that, on these datasets, the majority of additional true positives have very low $TU2C$ scores and in a very large number of cases, a substantial fraction of them (between 50 and 60%) have a $TU2C$ of 0, suggesting that these Tweets have no class-unique words at all and therefore, we expect them to potentially have fewer class-unique features. We believe these figures are convincing evidence that our methods of enhancing conventional CNN structures with skipped CNN or GRU on such tasks can significantly improve their capability of classifying Tweets that lack discriminative features, addressing the very challenging case of long tail.

5.5. Comparison Against Previously Reported Results

All the datasets used in this work have been previously used by other research and therefore, in Table 5 we compare the results obtained by our Base CNN, Base+sCNN and Base+GRU models against previously reported results on the same datasets on an 'as-is' basis. All results are calculated as **micro-average over all classes** in a dataset, as this is the predominant case in all previous work.

Firstly, comparing the micro-average results of Base CNN models against the macro-average results for the same models shown previously in Figure 6, once again we notice how the extremely unbalanced datasets can cause dramatic difference between the two ways of averaging performance figures across all classes in a dataset. The significantly higher figures from micro-averaging are due to the bias caused by a classifier's much better performance on classifying non-hate Tweets, which are always the majority in any dataset. This however, masks a classifier's capability in classifying hate Tweets, which are better reflected from Figures 8 and 9. The same observations can be made for the Base+sCNN and Base+GRU models (see detailed results in the Appendix).

Secondly, the difference in micro-average F1 made by Base+sCNN and Base+GRU is rather insignificant in most cases. This is again, due to the bias of micro-averaging on unbalanced datasets. As we have shown in Figures 8 and 9, the two enhanced structures can make significant improvement over the Base CNN models' ability of capturing hate Tweets, which are often in the long tail of a dataset.

Finally, comparing against previously reported results, our Base+sCNN and Base+GRU models can achieve much better results on most cases, except on the WZ.pj dataset where Park et al. [36] combined both word and character embeddings. Due to the lack of complete results on a per-class basis from previous work, we cannot really compare our macro-average results against their reported figures. However, our implementation of the SVM [12] and Base CNN models [16] can be used as good reference of state of the art methods.

5.6. Manual Error Analysis

We manually analysed a sample of 200 tweets each from the WZ-S.amt, DT and RM datasets, covering all classes to identify tweets that are difficult to classify. Consistent with our data analysis, errors due to

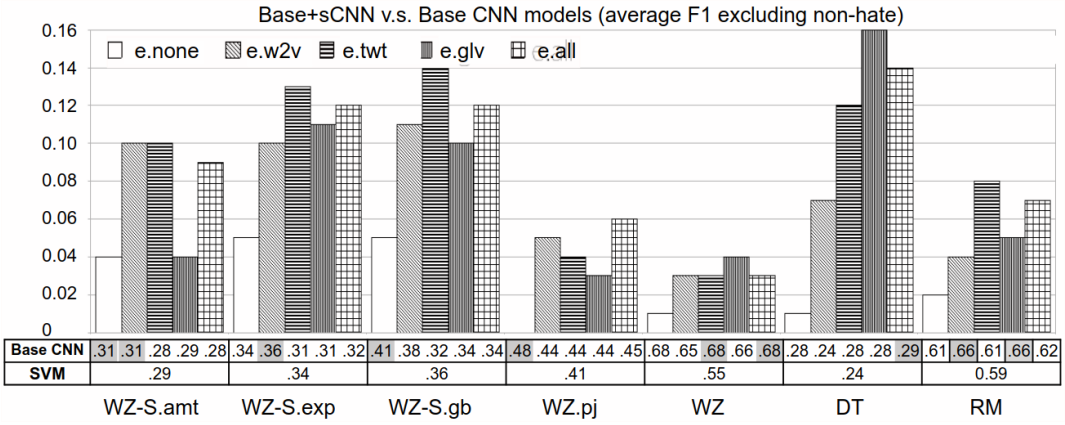

Fig. 8. Comparison of average F1 across hate classes only, as obtained by Base+GRU models against the SVM and Base CNN models. See Figure 6 caption for an explanation of how to interpret the results.

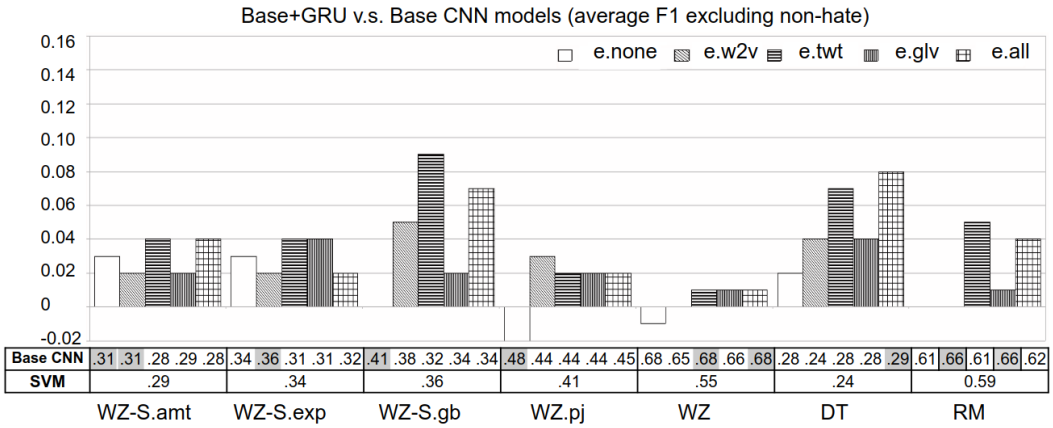

Fig. 9. Comparison of average F1 across hate classes only, as obtained by Base+GRU models against the SVM and Base CNN models. See Figure 6 caption for an explanation of how to interpret the results.

	State of the art		Base CNN					Base+sCNN					Base+GRU				
Dataset	score	source	e.none	e.w2v	e.twt	e.glv	e.all	e.none	e.w2v	e.twt	e.glv	e.all	e.none	e.w2v	e.twt	e.glv	e.all
WZ-S.amt	.839	[43]	.9	.918	.915	**.922**	.916	.9	.917	.915	.918	.915	.9	.919	.921	**.922**	.92
WZ-S.exp	.912	[43]	.903	**.917**	.91	.915	.91	.904	.914	.91	**.917**	.911	.904	.913	.914	.915	.915
WZ-S.gb	.783	[16]	.911	.922	.923	.926	.923	.91	.921	.923	.927	.923	.91	.926	**.928**	.925	.925
WZ.pj	**.827**	[36]	.8	.814	.816	.812	.817	.8	.8	.8	.816	.802	.8	.815	.821	.816	.823
WZ	.739	[44]	.802	.813	.816	.815	**.817**	.8	.804	.807	.817	.807	.804	.81	.812	.82	.815
DT	.8	[12]	.93	.944	.944	.944	.944	.93	.941	.942	.942	.938	.928	.938	.939	**.945**	.939
RM	.805	[12]	.879	.9	.888	.898	.889	.88	.887	.887	**.9**	.887	.876	.896	.899	**.9**	.897

Table 5

Comparing micro-average F1 across all classes on each dataset against previously reported results. The best performing result on each dataset is highlighted in **bold**. For [44] and [43], we used the result reported under their 'Best Feature' setting. For [12], we report results obtained from our re-implementation as the previous work treated the dataset with three classes.

non-distinctive features appear to be the majority of cases. For example, one would assume that the presence of the phrase `'white trash'` or pattern `'* trash'` is more likely to be a strong indicator of hate speech than not, such as in `'White bus drivers are all white trash...'`. How-

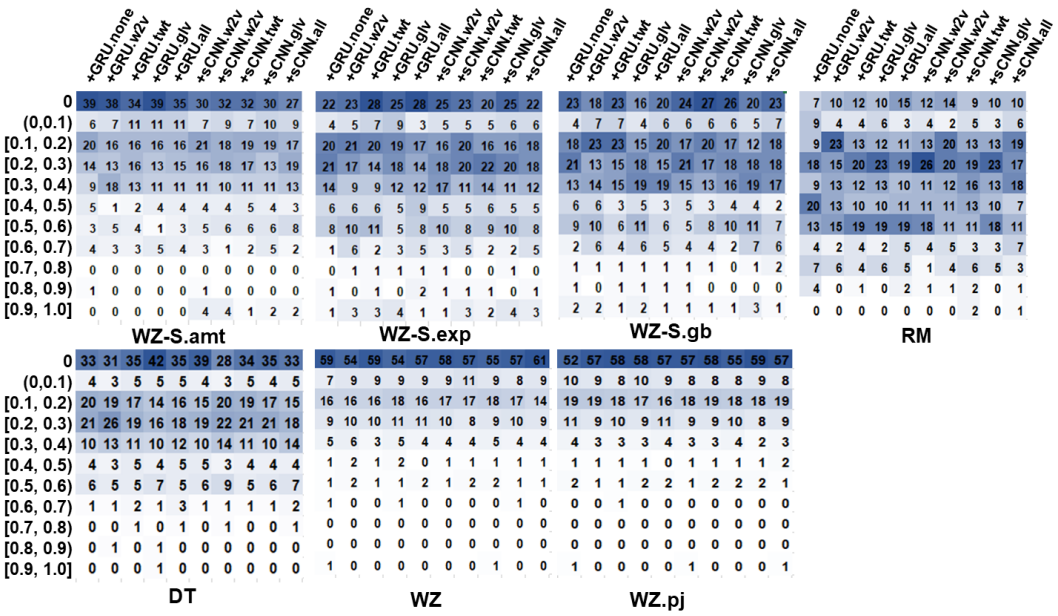

Fig. 10. (This figure is best viewed in colour) Distribution of additional true positives identified by enhanced models compared to their corresponding Base CNN models over different ranges of $TU2C$ scores (see equation 3). Each row in a heatmap corresponds to a $TU2C$ range. Each column corresponds to an enhanced model. The number within each cell is the % of additional true positives that belong to the $TU2C$ score range for the enhanced model. The numbers in each column sum up to 100%. Darker colour indicates a higher % while light colour indicates a lower %. For example, on the WZ.pj dataset, the rightmost column says that among the additional true positives identified by the Base+sCNN model (compared to Base CNN on this dataset), 57% has $TU2C = 0$, i.e., they do not have any class-unique words at all; 19% has $TU2C \in [0.1, 0.2)$, i.e., their class-unique words are between 10 and 20% of all words in the Tweets.

ever, our analysis shows that such seemingly 'obvious' features are also prevalent in non-hate tweets such as `'... I'm a piece of white trash I say it proudly'`. The second example does not qualify as hate speech since it does not 'target individual or groups' or 'has the intention to incite harm', which is indeed often very subtle to identify from lexical or syntactic levels. Similarly, **subtle metaphors** are often commonly found in false negatives such as `'expecting gender equality is the same as genocide'`. Further, expression of **stereotypical views** such as in `'... these same girls ... didn't cook that well and aren't very nice'` is also common in false negative sexism tweets. These are very difficult to capture purely relying on lexical and syntactic patterns in the data, because they require understanding of the implications of the language.

6. Conclusion and Future Work

The propagation of hate speech on social media has been increasing significantly in recent years, both due to the anonymity and mobility of such platforms, as well as the changing political climate from many places in the world. 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 semantic analysis of such content. A crucial task in this direction is classifying hate speech to different types.

This work makes several contributions in terms of: 1) undertaking thorough data analysis to understand the extremely unbalanced nature of the typical datasets one has to deal with in such tasks; 2) investigating methods based on the principles of developing more effective feature extractors in the form of novel DNN architectures, and exploring the incorporation of background information from large unlabelled corpora in the form of pre-trained word embedding models; and 3) empirically evaluate, compare and analyse the performance of several proposed methods against state of the art on the task of hate speech detection.

Lessons learned. *First*, we showed that the very chal-

lenging nature of identifying hate speech from short text such as Tweets is due to the fact that hate Tweets are found in the long tail of a dataset due to their lack of unique, discriminative features. We further showed in experiments that for this very reason, the practice of 'micro-averaging' over both hate and non-hate classes in a dataset adopted for reporting results by most previous related work can be questionable. It can significantly bias the evaluation towards the dominant non-hate class in a dataset, masking a method's ability to identify real hate speech.

Second, built on a state of the art CNN structure, we proposed to add a 'skipped' CNN or GRU structure, each serving as feature extractors to discover implicit features that can be potentially useful for identifying hate Tweets in the long tail. Evaluated extensively on the largest collection of Twitter datasets to date, we conclude that both structures can be useful for detecting hate speech in short texts such as Tweets. Among the two, the skipped CNNs are very powerful as they are able to obtain $\geqslant 4$ percentage points of improvement in F1 on all datasets and $\geqslant 10$ points on four, when considering only hate classes.

Third, our exploration of using different word embeddings trained on large unlabelled corpora could not draw conclusion as to whether one word embedding option is consistently better than others. Unsurprisingly, this is inline with previous findings showing that the superiority of one word embedding model is generally non-transferable across tasks, domains, or even datasets.

Future work. Despite making a substantial effort towards automatic detection of hate speech, we still consider our task largely incomplete, and we aim to explore the following directions of research in the future.

First, as have noted before, compared to non-hate, the number of examples belong to hate classes in a typical dataset is extremely small. Hence we will explore methods that aimed at compensating the lack of training data in a supervised learning tasks. Methods such as transfer learning could be potentially promising, as they study the problem of adapting supervised models trained in a resource-rich context to a resource-scare context. We will investigate, for example, whether features discovered from one hate class can be transferred to another, thus enhancing the training of each other.

Second, as shown in our data analysis as well as error analysis, the presence of abstract concepts such as 'sexism', 'racism' or even 'hate' in general is very difficult to detect if solely based on textual content. Therefore, we see the need to go beyond pure text classification and explore possibilities to model and integrate knowledge about users, social groups and mutual communication.

Finally, our methods prove to be effective for classifying Tweets, a type of short texts. We aim to investigate whether the benefits of such DNN structures can generalise to other short text classification tasks, such as in the context of sentences.

References

[1] I. Awan and I. Zempi. Jo cox 'deserved to die': Cyber hate speech unleashed on twitter, Last accessed: May 2017.

[2] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.

[3] BBCNews. Countering hate speech online, Last accessed: July 2017, http://eeagrants.org/News/2012/.

[4] BBCNews. Finsbury park attack: Son of hire boss held over facebook post, Last accessed: May 2017, http://www.bbc.co.uk/news/uk-wales-40347813.

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

[6] Pete Burnap and Matthew L. Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data Science*, 5(11):1–15, 2016.

[7] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 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, pages 71–80, Washington, DC, USA, 2012. IEEE Computer Society.

[8] Billy Chiu, Anna Korhonen, and Sampo Pyysalo. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, RepEval@ACL 2016, Berlin, Germany, August 2016*, pages 1–6, 2016.

[9] Jason P.C. Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[11] Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. Improving cyberbullying detection with user context. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, pages 693–696, Berlin, Heidelberg, 2013. Springer-Verlag.

[12] Thoams Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th Conference on Web and Social Media*. AAAI, 2017.

[13] Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Trans. Interact. Intell. Syst.*, 2(3):18:1–18:30, September 2012.

[14] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 29–30. ACM, 2015.

[15] Igini Galiardone, Danit Gal, Thiago Alves, and Gabriela Martinez. Countering online hate speech. *UNESCO Series on Internet Freedom*, pages 1–73, 2015.

[16] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90. Association for Computational Linguistics, 2017.

[17] Sahar Ghannay, Yannick Estève, Nathalie Camelin, Camille Dutrey, Fabian Santiago, and Martine Adda-Decker. Combining continuous word representation and prosodic features for asr error prediction. In *Proceedings of the Third International Conference on Statistical Language and Speech Processing - Volume 9449*, SLSP 2015, pages 84–95, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[18] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(10):215–230, 2015.

[19] Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. Single or multiple? combining word representations independently learned from text and wordnet. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2608–2614. AAAI Press, 2016.

[20] Edel Greevy and Alan F. Smeaton. 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, pages 468–469, New York, NY, USA, 2004. ACM.

[21] Guardian. Anti-muslim hate crime surges after manchester and london bridge attacks, Last accessed: July 2017, https://www.theguardian.com.

[22] Guardian. Zuckerberg on refugee crisis: 'hate speech has no place on facebook', Last accessed: July 2017, https://www.theguardian.com.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

[24] Irene Kwok and Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 1621–1622. AAAI Press, 2013.

[25] Quanzhi Li, Sameena Shah, Xiaomo Liu, and Armineh Nourbakhsh. Data sets: Word embeddings learned from tweets and general data. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 428–436, 2017.

[26] Natasha Lomas. Facebook, google, twitter commit to hate speech action in germany, Last accessed: July 2017.

[27] James D. McCaffrey. Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training, Last accessed: Jan 2018, https://jamesmccaffrey.wordpress.com.

[28] Yashar Mehdad and Joel Tetreault. Do characters abuse more than words? In *Proceedings of the SIGDIAL 2016 Conference*, pages 299–303, Los Angeles, USA, 2016. Association for Computational Linguistics.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[30] Thien Huu Nguyen and Ralph Grishman. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 886–891, 2016.

[31] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, 2016.

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

[33] A. Okeowo. Hate on the rise after trump's election, Last accessed: July 2017, http://www.newyorker.com/.

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

[35] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016.

[36] Jo Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on twitter. In *ALW1: 1st Workshop on Abusive Language Online*, Vancouver, Canada, 2017. Association for Computational Linguistics.

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

[38] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *International Workshop on Natural Language Processing for Social Media*, pages 1–10. Association for Computational Linguistics, 2017.

[39] Ellen Spertus. 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, pages 1058–1065. AAAI Press, 1997.

[40] Eleni Tsironi, Pablo Barros, Cornelius Weber, and Stefan Wermter. An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition. *Neurocomput.*, 268(C):76–86, December 2017.

[41] Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity*, pages 86–95, 2017.

[42] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 19–26. Association for Computational Linguistics, 2012.

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

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

[45] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Conference on Information and Knowledge Management*, pages 1980–1984. ACM, 2012.

[46] Shuhan Yuan, Xintao Wu, and Yang Xiang. A two phase deep learning model for identifying discrimination from tweets. In *Proceedings of 19th International Conference on Extending Database Technology*, pages 696–697, 2016.

[47] Haoti Zhong, Hao Li, Anna Squicciarini, Sarah Rajtmajer, Christopher Griffin, David Miller, and Cornelia Caragea. Content-driven detection of cyberbullying on the instagram social network. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3952–3958. AAAI Press, 2016.

Appendix A. Full results

Dataset and classes		SVM			Base CNN with e.none			Base CNN with e.w2v			Base CNN with e.twt			Base CNN with e.glv			Base CNN with e.all		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
WZ-S.amt	racism	.16	.39	.22	.35	.12	.17	.38	.08	.13	.33	.03	.06	.45	.04	.07	.37	.04	.07
	sexism	.58	.78	.66	.79	.7	.74	.85	.75	.80	.86	.71	.78	.87	.74	.80	.87	.71	.78
	both	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	non-hate	.94	.83	.88	.92	.96	.94	.93	.98	.95	.92	.98	.95	.93	.98	.96	.92	.98	.95
WZ-S.exp	racism	.23	.56	.3	.37	.29	.32	.57	.28	.37	.5	.16	.25	.58	.14	.23	.54	.18	.26
	sexism	.46	.74	.57	.71	.56	.63	.77	.60	.67	.74	.57	.64	.76	.59	.66	.74	.58	.64
	both	.29	.23	.16	.17	.04	.07	.25	.02	.04	.25	.02	.04	.25	.02	.04	.25	.02	.04
	non-hate	.96	.83	.89	.93	.96	.95	.94	.98	.95	.93	.97	.95	.93	.98	.95	.93	.97	.95
WZ-S.gb	racism	.26	.67	.35	.51	.36	.42	.53	.24	.33	.37	.10	.16	.56	.13	.22	.43	.13	.2
	sexism	.54	.79	.64	.78	.65	.71	.82	.69	.75	.83	.7	.76	.82	.72	.77	.83	.69	.75
	both	.25	.06	.09	.16	.07	.1	.25	.03	.05	.25	.03	.05	.25	.03	.05	.25	.03	.05
	non-hate	.96	.85	.89	.93	.97	.95	.94	.98	.96	.94	.98	.96	.94	.98	.96	.94	.98	.96
WZ.pj	racism	.49	.63	.55	.71	.65	.68	.74	.62	.68	.75	.63	.69	.75	.61	.67	.75	.64	.69
	sexism	.48	.63	.54	.68	.61	.64	.77	.55	.65	.78	.55	.65	.77	.55	.64	.78	.56	.65
	both	.1	.15	.12	.23	.08	.12	0	0	0	0	0	0	0	0	0	0	0	0
	non-hate	.84	.73	.78	.84	.88	.86	.83	.92	.87	.83	.92	.87	.83	.92	.87	.83	.92	.88
WZ	racism	.52	.64	.57	.7	.7	.7	.75	.65	.69	.74	.67	.7	.75	.64	.69	.74	.66	.7
	sexism	.47	.62	.53	.7	.6	.64	.79	.53	.63	.79	.54	.64	.81	.53	.64	.79	.55	.65
	non-hate	.84	.72	.78	.85	.88	.86	.83	.92	.87	.83	.92	.87	.83	.93	.87	.83	.92	.87
DT	hate	.16	.53	.24	.35	.23	.28	.55	.16	.24	.54	.19	.28	.54	.19	.28	.52	.2	.29
	non-hate	.97	.82	.89	.95	.97	.96	.95	.99	.98	.95	.99	.97	.95	.99	.97	.95	.99	.97
RM	hate	.48	.8	.59	.66	.57	.61	.76	.59	.66	.73	.53	.61	.76	.58	.66	.73	.54	.62
	non-hate	.98	.81	.87	.92	.94	.93	.92	.96	.94	.91	.96	.93	.92	.96	.94	.91	.96	.93

Table 6

Full results obtained by the SVM [12] and Base CNN models [16]) models.

Dataset and classes		**Base CNN+sCNN with e.none**			**Base CNN+sCNN with e.w2v**			**Base CNN+sCNN with e.twt**			**Base CNN+sCNN with e.glv**			**Base CNN+sCNN with e.all**		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
WZ-S.amt	racism	.34	.22	.26	.44	.39	.40	.46	.26	.33	.34	.14	.2	.46	.26	.33
	sexism	.77	.75	.76	.79	.83	.81	.8	.8	.8	.81	.8	.81	.79	.8	.79
	both	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	non-hate	.93	.95	.94	.95	.95	.95	.95	.96	.95	.94	.96	.95	.94	.95	.95
WZ-S.exp	racism	.46	.45	.45	.48	.67	.56	.46	.58	.51	.48	.51	.49	.46	.58	.51
	sexism	.71	.55	.62	.68	.75	.71	.67	.74	.7	.7	.73	.72	.67	.74	.7
	both	.15	.06	.09	.25	.06	.1	.25	.06	.1	.25	.02	.04	.25	.06	.1
	non-hate	.93	.96	.95	.96	.94	.95	.96	.94	.95	.95	.95	.95	.96	.94	.95
WZ-S.gb	racism	.48	.48	.48	.49	.7	.58	.49	.56	.52	.48	.47	.47	.5	.57	.53
	sexism	.76	.68	.71	.75	.8	.77	.77	.78	.77	.79	.78	.79	.77	.78	.78
	both	.37	.13	.17	.15	.08	.11	.25	.06	.09	.17	.06	.08	.25	.03	.05
	non-hate	.94	.96	.95	.96	.95	.96	.96	.96	.96	.96	.96	.96	.96	.96	.96
WZ.pj	racism	.7	.69	.69	.64	.8	.71	.64	.85	.73	.69	.75	.72	.64	.85	.73
	sexism	.66	.64	.65	.65	.69	.67	.66	.69	.68	.71	.66	.69	.67	.69	.68
	both	.11	.07	.09	.25	.05	.08	.25	.03	.05	0	0	0	.19	.08	.11
	non-hate	.85	.86	.86	.88	.83	.85	.88	.83	.85	.87	.87	.87	.88	.93	.86
WZ	racism	.69	.72	.7	.65	.81	.72	.64	.87	.74	.69	.78	.74	.64	.87	.74
	sexism	.67	.63	.65	.70	.63	.66	.71	.63	.67	.74	.62	.67	.71	.63	.67
	non-hate	.85	.86	.86	.86	.85	.86	.87	.85	.86	.86	.88	.87	.87	.85	.86
DT	hate	.37	.24	.29	.48	.24	.32	.49	.35	.41	.49	.4	.44	.46	.4	.43
	non-hate	.95	.97	.96	.95	.98	.97	.96	.98	.97	.96	.97	.97	.96	.97	.97
RM	hate	.66	.61	.63	.64	.78	.7	.64	.76	.69	.7	.71	.7	.64	.76	.69
	non-hate	.92	.94	.93	.95	.91	.93	.95	.91	.93	.94	.94	.94	.95	.91	.93

Table 7

Full results obtained by the Base+sCNN models.

Dataset and classes		Base CNN+GRU with e.none			Base CNN+GRUs with e.w2v			Base CNN+GRU with e.twt			Base CNN+GRU with e.glv			Base CNN+GRU with e.all		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
WZ-S.amt	racism	.28	.21	.24	.36	.12	.18	.5	.1	.17	.34	.08	.13	.4	.11	.17
	sexism	.77	.75	.76	.86	.75	.8	.86	.74	.8	.85	.78	.81	.86	.74	.8
	both	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	non-hate	.93	.95	.94	.93	.97	.95	.93	.98	.95	.94	.97	.96	.93	.98	.95
WZ-S.exp	racism	.48	.37	.42	.53	.44	.48	.47	.23	.31	.66	.27	.38	.46	.21	.29
	sexism	.7	.55	.62	.72	.65	.68	.75	.61	.67	.71	.66	.68	.74	.63	.68
	both	.13	.04	.06	0	0	0	.25	.06	.1	0	0	0	.25	.04	.07
	non-hate	.93	.96	.95	.94	.96	.95	.94	.97	.95	.95	.95	.95	.94	.97	.95
WZ-S.gb	racism	.53	.42	.47	.63	.45	.53	.56	.32	.48	.58	.23	.33	.52	.32	.4
	sexism	.73	.7	.71	.78	.76	.77	.81	.75	.78	.79	.75	.77	.8	.74	.77
	both	.13	.03	.05	0	0	0	.25	.03	.05	0	0	0	.25	.03	.05
	non-hate	.94	.96	.95	.96	.97	.96	.95	.97	.96	.95	.97	.96	.95	.97	.96
WZ.pj	racism	.69	.69	.69	.73	.64	.68	.73	.71	.72	.72	.66	.69	.74	.71	.72
	sexism	.67	.62	.65	.78	.56	.65	.76	.59	.66	.75	.6	.67	.76	.6	.67
	both	.25	.05	.08	0	0	0	0	0	0	0	0	0	0	0	0
	non-hate	.85	.87	.86	.83	.92	.87	.85	.91	.88	.84	.9	.87	.85	.91	.88
WZ	racism	.7	.7	.7	.73	.63	.68	.72	.71	.71	.72	.72	.72	.73	.73	.72
	sexism	.7	.61	.65	.85	.48	.61	.82	.49	.61	.79	.56	.66	.82	.5	.62
	non-hate	.85	.88	.86	.82	.94	.87	.83	.92	.87	.84	.91	.87	.83	.92	.87
DT	hate	.34	.28	.3	.43	.22	.29	.46	.29	.36	.54	.24	.32	.45	.31	.36
	non-hate	.96	.97	.96	.95	.98	.97	.96	.98	.97	.96	.99	.97	.96	.98	.97
RM	hate	.65	.59	.61	.74	.6	.66	.75	.6	.67	.77	.59	.66	.74	.59	.66
	non-hate	.92	.93	.93	.92	.96	.94	.92	.96	.94	.92	.96	.94	.92	.96	.94

Table 8

Full results obtained by the Base+GRU models.