Automated Detection of Online Hate Speech: An Evaluation of the Field

Samuel Payne - samuel.payne.12@ucl.ac.uk[1]

MSc Information Security

Supervised by: Prof. Emiliano De Cristofaro

Submission date: 11th September 2020

[1]**Disclaimer:** This report is submitted as part requirement for the MSc in Information Security at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged

Abstract

In a society dominated by global platforms that facilitate and encourage users to share their views, online hate speech is a growing concern. Whilst hate remains a small proportion of all online content, to begin to manage the problem, we must find effective ways to automatically identify hateful speech amongst the billions of posts across social media. Despite a wide body of research investigating the problem of automatic hate speech detection, verifying and comparing the performance of state of the art solutions is a relatively unexplored area. We reproduce four hate speech classifiers found in the literature and examine their performance on a large hate speech dataset. We observe a range of performance changes when compared to the original claims of each model. We conclude by challenging the suitability of common machine learning metrics for evaluating hate speech classifiers and explore avenues for future work.

Contents

Chapter 1

Introduction

Hate speech is not a new phenomenon, but the advent of the internet and social media has amplified the problem. Those who wish to spread hate now have easy access to global platforms and can quickly disseminate their views to large audiences. Whilst the philosophical and legal discussion around the nature of hate speech is ongoing, online providers are under increased pressure to at minimum, have an understanding of hate content on their platform. Despite free speech counterarguments, many platforms opt to ban hate speech, but often rely on user-driven reporting mechanisms to address the problem.

The increased propagation of hateful content and the need to efficiently manage it, has naturally led to research efforts into programmatically identify this content. Largely focused on the advancement of supervised machine learning techniques, numerous works position their method as *state of the art*, however far fewer works have examined these claims. As we will discuss, there are several use-cases for the practical application of automatic hate speech classification, however for all cases it is important that classification is effective and generalisable. To that end, we review the field of automatic hate speech detection, reproduce several machine learning classifiers found in the literature, and examine how these models perform on new data.

Our work begins with a background on hate speech and machine learning: we examine several attempts to define the term *hate speech*, investigate the legal position of hate speech, and review how the major social networking platforms have responded to the issue. We present several use-cases for automated hate speech classification and provide a brief overview of machine learning terminology and methods.

We then examine the wider literature on the topic. Labelled datasets are an essential component of supervised machine learning, therefore we explore attempts to compile hate speech datasets, discuss the issues these datasets might have, and how these issues impact the problem space. We review attempts to construct hate speech classifiers across various categories and discuss related work that attempts to reproduce and evaluate some of these existing models.

We outline our methodology for reproducing previous works by first discussing the dataset used for experimentation and shortlisting the various classifiers found in the literature. We then detail our process for selecting final choices for reproduction, outline our results format and discuss the implementation of each classifier[1], noting any assumptions or adjustments required to reproduce the original work.

We report our results, comparing them to the originally reported performance of each classifier. We also make some brief observations about our chosen dataset and compare the performance of all our reproductions.

[1]A sample of our source code can be found in **Appendix A**. A copy of the complete code is available at: https://github.com/sampayne/hate-speech

Finally, we compare our results to other evaluations of existing work (e.g [40], [3], [33]) and note any final conclusions and suggestions for future research.

Chapter 2

Background

2.1 Definitions of *hate speech*

Here we will look at the attempts to define *hate speech*, examining definitions from within the field of automatic hate speech detection and also looking at interpretations from philosophy. Legal definitions will be reserved for section **Section 2.2**.

Although noting that the term has no single meaning, Strossen (p. xxiii) [71] provides us with an initial definition of *hate speech* to examine:

> *"Speech that expresses hateful or discriminatory views about certain groups that historically have been subject to discrimination [...] or about certain personal characteristics that have been the basis of discrimination [...]"*

Here Strossen defines personal characteristics to be: race, religion, gender and sexual orientation. However, because this list is non-exhaustive, it limits the definition. Nationality, political belief or economic status are further examples of the types of targets that might incite hate. Strossen's definition therefore, could be considered too restrictive to societal change; groups that have not historically been discriminated against can still be the target of hate speech.

Heinze (pp. 22-23, quoting Weinstein & Hare) [34] provides a broader definition:

> *"expression which articulates hatred for another individual or group, usually based on a characteristic [...] which is perceived to be shared by members of the target group"*

Heinze, praises Weinstein & Hare's definition, noting "it does not distinguish between powerful and powerless groups, nor between numerical majorities and minorities". Heinze also cites others who criticise the term *hate speech* in general, suggesting that it implies a level of overt aggression in the text, which might not be the case. They imply it is possible to construct hateful text that appears calculated and sensible in its composition, whilst still fulfilling other requirements of hate speech.

Heinze also focuses on the case of religious hate speech, questioning the line between satire or legitimate criticism of a religious institution, and hatred directed at members of the religion. This highlights a clear difficulty when defining hate speech and identifying the threshold of hatred amongst legitimate discussion or humour. In their attempts to define hate speech, Fortuna & Nunes [18] address similar grey areas; stating that a nation state cannot be a target of hate speech, but the peoples of that state can, and that religion is not a valid target, but members of that faith as a group are.

Within the literature on automated detection of hate speech, a number of works attempt to arrive at a definition for the concept. Fortuna & Nunes [18], and MacAvaney et al [44] both gather and summarise several definitions from the literature and elsewhere. We compile some of these definitions:

- Nobata et al [53] - "Language which attacks or demeans a group based on race, ethnic origin, religion, disability, gender, age, disability, or sexual orientation/gender identity"
- Silva et al [69] - "any offense motivated, in whole or in part, by the offender's bias against an aspect of a group of people"
- Davidson et al [12] - "Language that is used to expresses [sic] hatred towards a targeted group or is intended to be derogatory, to humiliate or to insult the members of the group"
- de Gilbert et al [25] - "Hate speech is a deliberate attack directed towards a specific group of people motivated by aspects of the group's identity"
- Fortuna & Nunes [18] - "Language that attacks or diminishes, that incites violence or hate against groups, based on specific characteristics such as physical appearance, religion, descent, nation or ethnic origin, sexual orientation, gender identity of others, and it can occur with different linguistic styles, even in subtle forms or when humour is used"

Like Strossen [71], Nobata et al's [53] definition is limited in that it attempts to bound the potential targets of hate speech, precluding changes in demographics and social attitudes. The definitions of both Silva et al [69], and the second condition of Davidson et al [12] include broader descriptions of the hateful act. By including 'insults' in their definition, Davidson et al [12] potentially encompasses speech with any humorous intent as hateful. Though insults and humor directed to a targeted group might still be discriminatory or otherwise 'distasteful', they do not necessarily convey hatred, suggesting the definition is too broad. Similarly, almost any negative comment made about a target would fall within Silva et al's [69] definition with use of "any offense". We therefore argue that any definition should include a clear indication of some heightened level of intent in the action. The definitions from Nobata et al [53], de Gilbert et al [25] and Fortuna & Nunes [18] all include "attack", suggesting a certain level of aggression, though only de Gilbert et al [25] uses this as the singular indicator of intent. Others extend the definition further to include some form of 'degrading' action. De Gilbert et al [25] is also the only definition to specify that acts must be "deliberate". This could suggest that such speech must be understood to be hateful by the author, significantly limiting the definition. Finally, Fortuna & Nunes [18] is the only definition to include incitement of others to hate as a form of hate speech, showing a limitation in other definitions. Interestingly, no definition explicitly describes acts that include the sharing of existing hateful content, an important aspect of online hate speech.

Though we do not try to form a comprehensive definition here, we conclude that any attempt to define hate speech should include the following:

- A description of the targets of hate speech that allows for societal change.
- A clear indicator of the increased hateful or aggressive intent of the speech.
- A definition of the types of speech that account for humour and non-hateful criticism or offense.
- An acknowledgment that hate speech might not be directly authored by an individual, but instead shared or echoed from another, and that speech might not explicitly attack a particular group but instead incite others to do so.

It is clear there is no single definition of hate speech, but trends in the way it has been defined are certainly apparent. Whilst our ability to define the concept does not directly affect a machine learning classifier's performance, it does impact the methods by which we collect examples of hate to train classifiers [64] [80] and the effectiveness of legislation and policy that aims to deal with the problem of online hate speech.

2.2 Hate Speech and the Law

Hateful acts have been regulated by law for decades, long before the advent of social media [56]. However, international attempts to legislate against hate speech have been fraught with issues around the preservation of free speech and, more recently, jurisdictions over global online platforms. We present an overview of hate speech legislation at three levels: global international level, localised international level and at national level.

Global International View - United Nations

The International Covenant on Civil and Political Rights (ICCPR) is a United Nations convention introduced in 1966 that includes clauses to implicitly outlaw hate speech [56] with Article 20 stating:

> *"Any advocacy of national, racial or religious hatred that constitutes incitement to discrimination, hostility or violence shall be prohibited by law."*

O'Regan [54] notes that several nations, including the United Kingdom and United States issued reservations to their adoption of the convention, with the US ensuring that the convention did not override the freedoms granted by its constitution. The International Convention on the Elimination of All Forms of Racial Discrimination (ICERD) is a similar convention introduced in 1965 that focuses on racial discrimination. [55] O'Regan [54] notes similar reservations from signatories regarding the Convention's impact on national legislation, making clear that even early attempts to regulate hate speech were inconsistently received and adopted.

Local International View - European Union

Moving to an international, but localised framework, the European Union is limited in its ability to practically regulate online hate speech.

The Electronic Commerce Directive 2000 [74] imposes requirements on platforms to manage illegal content on their network, however it mandates only that they respond swiftly to flagged posts and makes no demand that the platform is actively monitored for qualifying content. The Directive also prevents member states from imposing monitoring requirements on platforms, limiting the regulation that nations could introduce to block or remove hate speech content. However, in order to mitigate any liability for the content on their network, thereby removing any motivation for active monitoring, the Directive requires a platform to act as a "mere conduit" for content and must not "select the receiver of the transmission". We argue that some platforms do not meet this definition; the dissemination of content between users on a platform is often not precisely defined. Not only do platforms algorithmically distribute content, but users can also pay for their content to gain a wider reach based on various metrics defined by the platform, not by the author, therefore falling short of the definition within the Directive.

A framework later introduced in 2008 [75] requires EU member states to "take measures" to punish:

> *"publicly inciting to violence or hatred directed against a group of persons or a member of such a group defined by reference to race, colour, religion, descent or national or ethnic origin;"*

The framework explicitly mentions "dissemination or distribution of tracts, pictures or other material", arguably encompassing online distributed content. However it starkly omits any references to sexuality or gender identity, therefore falling short of comprehensively regulating hate speech. The framework also allows member states flexibility on content that does not "disturb public order or which is threatening, abusive or insulting.", again falling short of some definitions of hate speech.

Most recently, the EU took a direct step in dealing with online hate speech. In 2016 the Code of Conduct on Countering Illegal Hate Speech Online was agreed with several major social media platforms [76]. Like the ECD 2000, the Code only imposes obligations around removing reported content and does little to encourage platforms to actively monitor for illegal content. Since its inception only five additional platforms agreed to abide by the code.

National View - United Kingdom

Finally, at the national level we look at the United Kingdom. A mix of statutes encompass hateful actions as the targets of hate, as deemed unacceptable to discriminate against by society have changed, and the mediums by which a hateful act can be carried out have evolved. The statute covering most hateful acts is the Public Order Act 1986 [31] and its various amendments. The original Act criminalised various actions taken on the basis of racial bias but lacked some specificity on the use of hate speech. Amendments within the Criminal Justice and the Public Order Act 1994 better codifies the use of words or writing as a qualifying action that might cause distress. The POA 1986 was further extended in 2006 by the Racial and Religious Hatred Act [32] to include actions committed on the basis of religious bias and specifies words and distributed written material as a possible medium for the act. The Crime and Disorder Act 1998 [28] and its amendments introduced the concept of an "aggravated" act committed on the basis of religious or racial bias that included acts intended to cause "harassment, alarm or distress". Only with the introduction of the Criminal Justice and Immigration Act 2008 [29] did crimes covered by the POA 1986 extend to those committed on the basis of discrimination against sexuality. O'Regan [54] notes various state attempts to review UK hate crime law but without outcome.

Online hate speech, though arguably encompassed by the previously discussed laws, also falls under two additional acts: the Malicious Communications Act 1988 [30] and the Communications Act 2003 [27]. The former regulates hate speech online implicitly, but the latter codifies the use of a "public electronic communications network" in sending "grossly offensive or of an indecent, obscene or menacing character". Neither Act explicitly references hate speech, but guidance from the Crown Prosecution Service (CPS) make several relevant references to hate crime on social media [68]. Earlier guidance from the CPS [9] also suggested that the sharing of content (e.g retweeting - the act of quoting a post on *Twitter*) can make the 'sharer' liable for the same offence as the original author. This is particularly relevant to the context of hate speech where participants may passively engage in sharing hateful content without authoring it directly.

We would argue that the lack of specificity regarding online hate speech in UK law, in combination with the jurisdictional issues arising from the nature of international online communities, makes it ineffective at either prosecuting individual instances of hate speech or regulating it more widely, or inciting change from online platforms.

2.3 Hate Speech and Social Media

The major social media platforms: *Facebook* [15], *Twitter* [73], *Snapchat* [35], *Instagram* [36], *LinkedIn* [41], *Pinterest* [58], *Reddit* [61], *YouTube* [83], all have policies directly banning the posting of hateful content on their platforms. *Reddit* clarified this policy only in June 2020 [62], demonstrating the divergence in platform providers' attitudes to hate speech, particularly given the previously discussed EU Code of Conduct was signed by some platforms several years prior.

Although platforms almost universally ban hate speech in their policies, disparity lies in how and in what level of detail platforms choose to define it. At the time of writing, *Facebook* and *Twitter's* policies on hate speech comprise of 1000-1200 words, with *Facebook* outlining a very specific three-tier system of categorising hateful content. In contrast, *Snapchat's* policy on hate speech is less than 60 words. Although the length of a policy is not an indication of its quality or the effectiveness of a platform in enforcing it, the stark difference does demonstrate the inconsistency in how platforms approach the issue of hate speech.

What is not clear in most cases is what is actively being done to detect hateful content and enforce these policies. *Twitter's* policy explicitly states that hateful content must be manually reported by users, presumably for review by an employee. An overview of social media policies [70] suggests that only *Facebook*, *Instagram* and *YouTube* employ automatic identification of hate speech.

What is clear is that if policies, as written, are to be enforced effectively, further implementation of automatic detection methods is needed.

It is important to note that here we only discuss platforms largely intended for wider sharing of content. Peer-to-peer or peer-to-group messaging platforms such as *WhatsApp*, have far fewer restrictions or policies in-place [70] with the founder of *WhatsApp* being quoted as being actively against any policing of users [52].

2.4 The Case for Hate Speech Classification

Although the development of new natural language analysis techniques is a valid scientific goal in of itself, that can be applied outside of the context of hate speech, much of the wider literature elects to forgo a practical justification for the work, instead simply asserting the growth of hate speech online and positing a need for understanding. They also fail to acknowledge the possible negative implications of the work and in particular, work that over estimates its effectiveness. We therefore discuss three applications for effective hate speech classifiers.

Censorship and Moderation

Although not often overtly discussed in the literature, perhaps the most obvious outcomes from the wider deployment of automated hate speech classification are forms of censorship. If implemented bluntly, classifiers could be utilised to entirely censor hate speech, either preventing users from posting content, or entirely blocking other users from accessing it. Total censorship is a controversial concept however, assuming that total censorship of hate speech is the intended outcome, the deployment of ineffective tools for classification is still problematic. In the extreme, if platforms are unable to reliably differentiate between a post attacking a particular group and a post overtly supporting that group, the result is the censorship of groups that their policies intend to protect. Although this would seem an obvious error to avoid, *false positives* are a concern across the existing work. As we will see in our review, many works pre-sample texts based on particular keywords and topics (e.g [79], [12], [85]). In addition to the possibility of censoring overt support, there is also the possibility of censoring genuine discourse about a particular targeted group or even hate speech itself. Ineffective classifiers may be unable to determine the difference between quoting hate speech in order to criticise it and the original intent of the post. There are however a few contexts where total censorship is arguably desirable and false positives less of a concern, for example on platforms used by children. A 2016 report by the UK communications regulator Ofcom [10] suggested that 34% of teenage children observed online hate speech in the previous year, prompting concerns over children's exposure to hateful content [37].

Rather than outright censor content, platforms might instead elect to limit the reach of a post identified as hateful. For example, by limiting the number of users that see a particular post, or restricting how many times a post might be shared or propagated more widely. Platforms utilise similar controls for other types of content; notably *WhatsApp* introduced measures that limit the sharing of posts to large groups [43], in an attempt to prevent the spread of misinformation. A less restrictive measure might be to simply occlude a particular post and warn the user of hateful content, allowing them to opt-in to read it. As *Facebook* has implemented these types of warnings for image and video content it deems inappropriate for immediate consumption [16], it is reasonable therefore to suggest that such measures could be put in place for hate speech.

Measurement, Sentiment and Event Analysis

Some works argue that the automatic classification of hate speech has its application in the measurement of the phenomenon.

Burnap & Williams [7] make the case for automatic detection as a means of measuring the response to a particular event that is likely to induce hateful comments. They argue that, by identifying trends in immediate aftermath of an event, these trends can contribute to how authorities and governments choose to respond.

Works by Silva et al [69] and Mondal et al [49] make the case that measurement of hate speech allows for both an understanding of the targets of hate at any given time, as well as an opportunity to detect new and previously unknown targets, that might be emerging in society. They also make the case that by having an accurate measurement of hate speech online, this can be used to inform on the occurrences of hate outside of the online space. Schmidt & Wiegand [67] go as far to suggest the concept of *anticipatory governance*; the idea that, by examining the change in frequency of online hate speech, significant hateful incidents (racial violence, terror attacks etc.) could be predicted and perhaps circumvented before they happen.

As well as response to individual events, (as in [7]) we argue that wider analysis and measurement of hate speech online can also inform long term policy. By identifying current trends and new targets of hate, education and social policy can be adjusted to counteract these new patterns. If necessary it might also inform judicial policy as new lines for what constitutes a hateful act are defined and change over time.

Deployment of *Counterspeech*

Perhaps considered an alternative to censorship, *counterspeech* might be defined as content that attempts to directly counter hate speech, denigrate those who produce hateful content and/or show support for groups and communities targeted by hate. The content that forms counterspeech can be varied. Mathew et al [47] groups it into eight sub-types: presenting facts, pointing out hypocrisy or contradictions, warning of offline or online consequences, affiliation, denouncing hateful or dangerous speech, humor, positive tone, hostile.

Schieb & Preuss [66] examine the effectiveness of counterspeech in online communities and suggest that a relatively small amount of counter-content can prevent an audience from developing more extreme views and even steer them towards changing their opinion. Social networks have already begun to identify misinformation and rather then simply censor it, they instead highlight to users that the content might be factually inaccurate (though *Twitter's* implementation of such measures is a manual process) [11].

We argue that similar measures could be applied to hate speech and rather than turning to automated censorship of hateful content, online platforms could utilise effective hate speech classification techniques to supplement hateful content with fact-based counterspeech or promote user-generated counterspeech alongside the hateful content. Mathew et al [47] notes however, that different forms of counterspeech show varying levels of acceptance within each targeted community. This suggests that distributing counter-content on an automated basis may not be as simple as identifying the hateful content to counter, but instead rely on the automated identification of effective counterspeech for a given hate target.

2.5 Machine Learning Overview

Next we briefly define and outline important terminology and concepts on the topic of machine learning:

Supervised Learning

Our work and much of the existing literature on the automated detection of hate speech, is concerned with various supervised learning methods. In short, this machine learning method uses examples of a particular subject (text, images etc.), each assigned a particular category, to learn the characteristics of each category and categorise new, previously unseen examples. In the context of hate speech detection (and other text-analysis problems) the subjects being classified are referred to as *texts* and could include *tweets* (text posts from *Twitter*), news comments, forum posts or web pages.

Labels & Annotation

The available categories of *text* in a given dataset are referred to as *labels*. Each text is assigned a label in an *annotation* process. Within the context of hate speech, texts are generally labelled as some variation on a *normal* label or one of a number of labels that might cover a scale of hatefulness (e.g *offensive*, *hateful*) or various sub-categories of hatefulness (e.g *racism*, *sexism* etc.). In general, the body of research around hate speech classification deals with single-class datasets, that is, each text is assigned only one of the available labels. *Multi-class* annotation refers to the, assigning of more than one label to a particular text.

Features

Texts in isolation are not useful for supervised learning. Instead, characteristics or *features* of each text are derived and used as input to a classifier. These could be simplistic *surface features* such as the length of the text, number of words, number of sentences etc. One of the most common features used for hate speech detection are *n-grams*; the frequency of n-length contiguous words or characters in a text. Often these are derived in a range e.g 1-3, which determines the frequency of individual words or characters, as well as pairs and triplets of words or characters. For hate speech detection, features might also include a sentiment score for the text or a count of the number of offensive words from some external dictionary (e.g [1]).

Algorithms, Classifiers & Models

The terms *classifier* and *model* are used interchangeably in the field and refer to the combination of a selection of features, with an underlying classification algorithm or chain of algorithms. These algorithms tend to fall into one of two categories. *Traditional* machine learning algorithms that could be considered predictable in their behaviour and are based on well understood statistical methods including Logistic Regression (LR) and Support Vector Machines (SVMs). *Deep-learning* methods are the contrast and are based on *neural networks*, layers of nodes that, starting with the input features, assign a weighting to each output of the previous layer. Importantly the weights used in each layer are, in general, not known. Recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are examples of deep learning methods.

Performance and Evaluation

Classifiers are evaluated by splitting any given dataset into training and testing segments. The classifier is first trained using the training data, then predictions for each testing text are obtained. The model is evaluated based on the accuracy of these predictions according to a number of common metrics. Using an example *hateful* label:

- *Precision* - the percentage of correctly predicted *hateful* texts among all texts predicted as *hateful.*
- *Recall* - the percentage of correctly predicted *hateful* texts among all texts actually labelled *hateful.*
- *F1* - a combination of Precision and Recall, specifically the *harmonic mean.* Often considered an overall accuracy score.

These metrics can be calculated for each label in a dataset or the results across labels combined to determine the performance of the model in aggregate. These metrics are usually combined using one of three methods:

- *Micro Average* - also referred to as the model's *Accuracy*, computes a single value based on the global number of true positives, false negatives and false positives.
- *Macro Average* - the unweighted mean of all label scores. This aggregate does not account for frequency labels in the dataset. An infrequently occurring label with an extreme performance score has the potential to skew results.
- *Weighted Macro Average* - the mean of all label scores accounting for the distribution of labels in the dataset.

An additional consideration in the evaluation of a classifier is the selection of the training / testing segments from the dataset. In general, segments are obtained randomly and do not overlap. However a totally random sample can lead to infrequently occurring labels being omitted entirely from either segment. To resolve this a *stratified* split can be used to maintain the distribution of labels in each segment.

Finally, to overcome any potential bias in a randomly obtained training / testing split, classifiers are evaluated using a *k-fold* cross validation method. The dataset is split into a training / testing split *k*-times, with each text appearing in the testing segment exactly once. The model is evaluated against each of the *k* splits of the data and the results of each split are aggregated.

Chapter 3

Existing Work

The field of work on the automated detection of hate speech is broad, therefore before reviewing the literature a suitable scope must be established. We limit our review to works produced from 2015 onward allowing for a comprehensive overview of the field, focusing on modern contributions, whilst setting a cut-off point for brevity. We also limit our review to those works that deal specifically with the classification of hate speech or hateful texts. Works that only implicitly encompass hate speech, addressing topics like cyberbullying, are disregarded.

3.1 Existing Surveys

We provide an overview of several aspects of the field most relevant to our later experimentation, however two works attempt to review the literature more generally. Schmidt & Wiegand [67] focus on the type of classifier features used, and include an overview of datasets and annotation. Fortuna & Nunes [18] perform a systematic review of the field, quantifying trends in the literature.

3.2 Hate Speech Corpa

There are numerous works in the literature that attempt to assemble labelled datasets on the topic of hate speech. Although a range of labelling schemes are used, the source of data is somewhat narrow, with a large subset of works gathering texts from the social network *Twitter*.

We argue that the work focuses on *Twitter* as a source of data because in general, posts are not just publicly available for anyone to read, but also 'broadcast' on the platform with no particular recipient in mind. *Twitter* is also used frequently as a platform by public figures, and promotes a culture of sharing opinions on current affairs. From a technical perspective, posts are easily accessed programmatically using their *Twitter's* API[1], making it trivial to sample texts from the platform, either randomly or focusing on a particular set of users. This is perhaps in contrast to a platform like *Facebook* that generally limits the availability of posts to a more narrow audience. In addition, posts on *Twitter* are not explicitly categorised, allowing for a more random distribution of the types of speech that can be collected. This is in contrast to a platform like *Reddit* where posts are explicitly grouped by a category or topic.

Waseem & Hovy [79] compile ≈ 17000 tweets labelled as *sexist*, *racist* or *neither*. The authors first conduct a manual search for common offensive terms and queried random tweets containing these terms. They note this ensured the final dataset included non-hateful tweets that used terms that would be considered hateful in other contexts. However, this limits the dataset to texts that are relevant to topics that incite a certain

[1]https://developer.twitter.com/en/docs/twitter-api

degree of hate speech, and therefore the sample does not contain any texts that could be considered random. This has the potential to both bias the annotation process in favour of hateful and offensive labels because of the presence of slurs and offensive terms [65] and also the generalisation of any classifier trained on the dataset by not providing examples of ‘random noise‘.

Wasem [78] builds upon this initial work, extending it by ≈ 4000 tweets and adding an additional *both* label for the new tweets and ≈ 3000 of the tweets that overlap with the initial dataset.

Davidson et al [12] provides a larger dataset of ≈ 25000 tweets. Like Waseem & Hovy [79], they first gather a list of offensive terms from the online lexicon *Hatebase*[2], and query *Twitter* for texts containing these terms, randomly sampling the results. The final set of texts are labelled as *hateful*, *offensive (but not hateful)* or *neither*, with annotators instructed to focus on the context of the texts. Like Waseem & Hovy's [79] dataset, we argue that by pre-sampling on the existence of offensive terms, texts are predisposed to hateful content. However, this is acknowledged in the work's conclusion.

Zhang et al [85], using largely similar methodologies, compile a corpus of ≈ 2500 tweets labelled as *hate* or *non-hate*, but limited to texts contain terms related to Islam or refugees. We argue that bias in the dataset arises from the possible implicit support or prejudice annotators might have, relating to the specifically sampled target groups [80].

Malmasi & Zampieri [45] present a corpus of ≈ 14500 tweets labelled *hate*, *offensive* or *ok*, but they do not provide further detail on the methodology used.

Founta et al [20] produce a dataset of ≈ 100000 tweets labelled as *abusive*, *hateful*, *spam*, *normal*. Unlike other works, the final texts are a mix of randomly sampled public tweets with the distributions of the minority labels artificially increased by filtering on offensive terms. The result is a dataset that contains enough examples of each label to remain useful, whilst still containing texts that are truly random in their origin.

Most recently, Basile et al [5] collate a corpus of ≈ 13000 tweets, labelled as *hateful* or *not*. They also add annotations if *hateful* texts are *aggressive* or not and if the text targets an *individual* or *generic group*. Similar to Zhang et al [85], the dataset is focused on specific targets of hate: women and immigrants. As in other works, Basile et al [5] uses a keyword sampling approach to gather tweets.

Though *Twitter* is clearly the focus of much of the literature, a few works use other sources for corpa.

Although not directly relevant to hate speech, Nobata et al [53] builds a dataset of several hundred thousand comments from *Yahoo! Finance and News*. This work labels texts as *clean* or *abusive* and uses a near-random sampling of comments.

Gibert et al [25] take forum posts from *Stormfront*, a white supremacist forum, and rather than annotating entire texts, annotate each component sentence. Texts are gathered by randomly sampling posts from across the platform. Around 10000 sentences are labelled as *hate*, *no hate* or *relation*. *Relation* here is used to identify sentences that, when used in isolation, are not considered hateful but when combined with the context of surrounding sentences within the same post, should be. Unlike most other works, these texts are not pre-sampled based on the existence of hateful or offensive terms, but given the nature of the platform we would argue a similar bias is introduced.

Finally, Gao & Huang [22] offer a corpus of ≈ 1500 comment from the *Fox News* website, annotated as *hateful* or *non-hateful*. The comments are gathered from a manually selected set of news articles that the

[2]https://hatebase.org/

authors claim were popular at the time of writing. They also include the title of the original article along with the comment texts, to allow for the context of the parent article to be accounted for in classification experiments.

As Davidson et al [12] discuss in the conclusions about their dataset, we would conclude on the basis that generally texts are pre-sampled for certain search terms, existing hate speech corpa generally suffer from two weaknesses; they fail to include truly ‘ordinary’ texts that are not in some way prone to instances of hate speech, and they fail to include hateful texts that do not use slurs or offensive terms. An additional conclusion could be drawn about the limitations of the field in exploring different sources of hate speech beyond the primary focus of *Twitter*. However, rather than bluntly encouraging diversity in datasets, future work might more closely explore the drawbacks of using *Twitter* as the primary source of hate speech corpa and explore what additional sources would be beneficial to the field.

Criticism of Corpa Annotation

Following our overview of existing corpa and the patterns identified in the pre-sampling of texts and the potential for annotator and classifier bias, we look at several works that specifically evaluate datasets and their associated methodologies.

Work by Ross et al [64] begins to identify possible problems with hate speech annotation, concluding that an improvement in definitions and guidelines for the coding process is needed to reduce disagreement between annotators. They suggest collecting multiple labels for each text and aggregating results to arrive at a final label. They also suggest that hate speech classification might not be suited to a binary or discrete labelling scheme and instead a scale of hatefulness might be more useful.

Waseem [78] builds on this work, examining the relationship between hate speech labels assigned by experts in the topic and amateurs. They conclude that although datasets annotated by experts improve performance in classifier experiments, using unanimous decisions by amateurs combined with expert opinion to determine split decisions can lead to good performance, without the need to excessively burden experts to compile large corpa.

Waseem et al [80] reiterate criticisms that many labelling schemes are too broad and not necessarily best-suited to hate speech coding. They outline a topology for future work aiming to act as a framework for separating abusive language in to quadrants: *directed/generalised* and *implicit/explicit*. They provide examples, noting that most prior work fails to account for this topology in their annotation process and that annotation guidelines are too broad. They also highlight the potential for annotators to be biased in their evaluation of texts as hateful/offensive, depending on their implicit bias relating to the hate target group. We would argue that there is potential for this problem to be magnified when corpa are already pre-sampled using methods previously discussed.

Klubicka & Fernandez [38] analyse the dataset by Waseeem & Hovy [79]. They discuss the possible biases, particularly given the narrow set of pre-sampling terms used to produce the dataset and the very small set of users responsible for the texts in the corpus. They also point out the frequency of retweets in the dataset and the impact these texts have when considering the distribution of hate speech. They argue that retweets do not necessarily carry the same sentiment as the original post. It could be a neutral response or possibly a type of counterspeech. Whilst this does not alter the dataset’s usefulness in text-only classification (most classifiers remove any extraneous syntax from the retweet in their pre-processing), it does have implications for the various ‘contextual classifiers’ discussed later that use relationships between users and their content as features for classification. Arango et al [3] examines more fully the impact of small user distribution in datasets, arguing that, when a small number of users are responsible for significant proportions of hateful texts, classifiers might actually be learning to identify the hateful users, rather than hateful texts; thereby reducing the utility of such datasets and overestimating the performance of classifiers evaluated on them.

Sap et al [65] aim to quantify the types of bias discussed by Waseem et al [80], specifically examining racial bias in the Davidson et al [12] and Founta et al [20] corpa, finding that texts written in African American English were significantly more likely to be labelled as *offensive* than other texts. The work mentions the concept of communities (that are the targets of hate speech) reclaiming phrases and slurs for use by members of those communities. Their results show the potential bias introduced by annotators not understanding the use of reclaimed slurs within those communities and mis-label them as *hateful* or *offensive*. We reiterate the potential for compounding bias arising from the tendency for works to pre-sample texts based on particular topics and search terms.

3.3 Hate Speech Classifiers

There are numerous attempts to develop and improve upon models to classify hate speech and abusive language. We provide an overview of the literature, dividing the work into three:

- **Text Classifiers** - models that use only the text to derive features. We explore both traditional and deep learning-based methods.
- **Contextual Classifiers** - models that utilise data in addition to the text as a basis of features. This might include data about the author; gender, ages etc. or including data about the text's popularity; number of interactions, 'likes' etc.
- **Weakly Supervised Classifiers** - models intended to overcome some the limitations of supervised learning methods for classifying hate speech.

Text Classifiers

Agarwal & Sureka [1] explore two classification techniques: SVM and K-Nearest Neighbour (KNN). They experiment with both algorithms on a *Twitter* corpus using predominantly surface textual features including the frequency of punctuation, emoticons and hashtags, along with n-grams. They also include some rudimentary measurement of positivity/negativity using keywords matches. They report an Accuracy of 0.90 for the KNN model and 0.97 for the SVM model, with the KNN model's Precision score falling significantly behind the SVM model. They conclude that the user of religious and war related keywords and negative emotions as part of the feature set contributed significantly to the performance of both models, and that features based on internet slang, emoticons and punctuation were far less significant.

Nandhini & Sheeba [51] implement a Naive Bayes (NB) based model to detect cyberbullying, with word frequency and part-of-speech tagging driven features. They report Accuracy values as high as 0.945 for a binary classification. They do not make any comparisons of the efficacy of their model or the features chosen. Liu & Forss [42] also experiment with a NB model, including Latent-Dirichlet-Allocation topic models as part of their comparisons. Although precise results are vague in the work, they conclude NB models are still competitive in comparison to SVM and neural network models.

Djuric et al [14] using a LR model to classifier *Yahoo Finance* comments as *hateful* or *clean*. They aim to improve upon 'bag-of-words' (BOW) based features commonly used, and instead use a continuous-bag-of-words (CBOW) feature set that assumes that the order of words and their place within a sentence are statistically significant. They report an 'area under the curve' (AUC) result and cite improved performance of the CBOW-based model versus two variations of a BOW model. Using AUC as a metric however, makes it difficult to compare to most other works that generally report the metrics outlined in **Section 2.5**.

Gitari et al [26] implement a rule-based classifier and explore the impact of accounting for hateful sentences and the theme of sentences as well as the subjectivity of sentences. They report relatively poor results overall, with F1 scores of 0.708 and 0.698.

Burnap & Williams [8] experiment with SVM and Random Forest Decision Tree (RFDT) algorithms. Like Agarwal & Sureka [1], they report that SVM consistently outperformed the alternative. Features used in their experiments include n-grams (n = 1-5), n-gram typed dependencies and the existence of hateful terms based on an external lexicon. They conclude that the inclusion of typed dependencies in the feature set can improve performance. Interestingly, accounting for hateful terms along with standard n-grams performed worse than n-grams alone in half the categories of hate they examined. We suggest that this might be due to the relative use of slurs and offensive terms within hate speech targeting different groups. When collapsing all types of hate speech together into a single *hate*, *non-hate* dataset, the model achieved promising results combining all types of feature, achieving Macro results of 0.96, 0.97, 0.96 for Precision, Recall and F1 respectively. Tulkens et al [72] also experimented with an SVM model, with features based on dictionaries of Dutch words grouped into various categories (e.g skin colour, nationality etc.). Several variations of the dictionaries were used, with a peak F1 score of 0.46 for the classification of racist comments.

Nobata et al [53] experiment with an alternative regression model, comparing and combining several features, including character n-grams and three types of word-embedding feature. They experiment with several datasets, reporting an F1-score as high as 0.817 when combining all features on a *Yahoo News* dataset. They conclude that the feature set can improve performance combined with 'standard' features, but note that the use of character n-grams alone can lead to good results. They also compare their model on the same dataset as Djuric et al [14] and report an improved AUC value of 0.905, versus 0.8007 (in [14]). In follow-up work, Mehdad & Tetreault [48] further investigate the use of character n-grams as features, reporting a result using a combination NB + SVM model that, on the same dataset (as [14], [53]), improves on the F1 score of the original work. Malmasi & Zampieri [46] again confirm the performance of character n-grams. Using an SVM algorithm they compare various 'n' sizes as well as word n-grams and word skip-grams. They report character 4-grams as the best performing feature, with results comparable to Mehdad & Tetreault [48]. They also report that a combination of features does not yield vastly improved results. Malmasi & Zampieri [45] continue investigations into SVM models, combining various n-gram features and confirming that character 4-grams were the superior performing feature. They attempt to improve on these results by implementing a meta-learning model that feeds the results of individual classifiers as input to a second SVM layer. Although they see improved results, the additional complexity yields only a 0.018 absolute increase in model Accuracy compared with character 4-grams. More experimentation on SVM models by Robinson et al [63], demonstrates improved results from the SVM model proposed by Davidson et al [12], when additional feature optimisation is applied. Like prior work, MacAvaney et al also experiment with a meta-learning approach utilising SVM based models. They only report a subset of their results and only one of these results improves on the basic character 4-gram approach reported in earlier work.

Davidson et al [12] test several models, but elect to tune and report on a LR model utilising various features including term frequency–inverse document frequency weightings (TF-IDF), part-of-speech tags and a sentiment score assigned to each text. This is in addition to features counting the frequency of hashtags, mentions retweets etc. They report scores of 0.91, 0.90 and 0.90 for Precision, Recall and F1 respectively but note that for the *hate* label in their experiment the model achieved Precision and Recall scores of 0.44 and 0.61, highlighting that the model tended to underestimate the offensiveness of texts. Experimenting with several traditional models, Gaydhani et al [24] also report an LR model as having the better performance. Their aggregate results slightly improve on Davidson et al [12], with scores of 0.96, 0.96, 0.96 for Precision, Recall and F1. The performance for the *hateful* label however is notably higher; 0.94 and 0.96 for Precision and Recall.

Gambäck & Sikdar [21] construct a CNN-based classifier and experiment with several methods for constructing features from embeddings and character n-grams. They report a peak F1 and Recall score for a *word2vec* based embedding, but the best Precision score was achieved using a random embedding model. Vigna et al [13] also explore neural network-based methods, constructing a RNN model and comparing against an SVM model. In a binary classification experiment, predicting *hate* and *no hate*, neither model vastly outperforms the other. Badjatiya et al [4] also propose a RNN based approach and report an aggregate score of 0.930 across Precision, Recall and F1. Their results support Gambäck & Sikdar [21] showing that a randomised embedding model can achieve improved results. Further evidence of the efficacy of CNN models is provided by Park & Fung [57]. They implement a CNN model utilising a combination of character and word level embeddings. They report performance of 0.827 for Precision, Recall and F1. This represented only a small increase in performance versus word or character embeddings used separately, where word embeddings performed better than character; an interesting result considering the performance of character n-grams in other works. Zhang et al [85] propose a another CNN model, that makes use of a gated recurrent unit network (GRU). They perform experiments comparable to both Gambäck & Sikdar [21] and Park & Fung [57], reporting a 0.03 increase in F1 score over the former, and a 0.01 decrease compared to the latter. They do however report a peak F1 of 0.94 when run against Davidson et al's [12] dataset. Zimmerman et al [86] also report results for a CNN based method, reporting an F1 score that falls short of previously discussed models. Kshirsgar et al [39] and Al-Makhadmeh & Tolba both construct novel neural network models, the former reporting an aggregate F1 result of up-to 0.924, but still performing poorly on the *hate* label in Davidson et al's [12] dataset. The latter reports an Accuracy of 0.987, but does not provide any label-level results. Finally, Mozafair et al [50] experiment with using neural networks in combination with the BERT model; a model pre-trained on English *Wikipedia* and *BookCorpus*. Although the use of BERT alone demonstrates good results, the best performing model combines BERT with a CNN and scores up to 0.92 for Precision, Recall and F1.

Watanabe et al [81] construct a model using the *J48Graft* algorithm, based on decision trees, utilising a large number of features. The performance of the model using all features was reasonable, with aggregate scores of 0.793, 0.784, and 0.784 for Precision, Recall and F1 on a ternary classification including a *hateful* label. Scores for the *hateful* label in isolation were also promising: 0.770, 0.699 and 0.732. However, the authors do not acknowledge that their experiments with simple unigram features yield reduced but comparable results, with the Precision score for the *hateful* label actually improving.

Contextual Classifiers

Several works in the literature attempt to extend the detection of hate speech beyond simply a text analysis problem.

Waseem & Hovy [79] use the gender and location of *Twitter* users as features, in addition to character n-grams of the text. Despite reporting a heavy gender bias in the frequency of hateful labels, they report only a slight increase in model performance when including gender, and a minor detriment to performance when also including the author's approximate location. They do note however that accurate coverage of both features is limited.

Gao & Huang [22] also experiment with including adding features from outside the text, examining comments posted on news articles. They include the author's username, on the basis that an offensive username could be indicative of hateful content, and the title of the associated article on the basis that articles focusing on particular topics are more likely to attract hateful comments. They note a performance improvement of 4% when the contextual features are included with features based only on the content of the text.

Founta et al [19] go further than Waseem & Hovy [79] in experimenting with *Twitter* user meta-data, examining the effects of user popularity (e.g number of followers), their activity on the network and also the user's wider follower/following network. They conclude that the inclusion of such metrics substantially increases performance over textual features though combining text and user data yields a slightly higher accuracy than text and network data alone. Though they make it clear that given a performance improvement when all features are combined, there is no duplication of information across the types of features used.

Building on Waseem & Hovy [79] and incorporating similar network features as Founta et al [19], Unsvåg & Gambäck [17] further investigate the correlations of user meta-data with hateful content and the effects of these features on classification. Unlike Waseem & Hovy [79], who report a strong bias in hateful content when compared with gender, Unsvåg & Gambäck find no particular correlations of user characteristics with hateful tweets. Like Founta et al [19], they find that user and network features do improve classifier performance, concluding that network features, specifically the number of followers, led to the biggest improvement. However like previous work, they find that some user-features are largely ineffective.

Pitsilis et al [59] examine the impact of analysing a user's propensity to post hateful content as a useful feature for detecting future instances of hate speech. Their results show that including features based on this propensity improves performance, but when compared to a model with no additional data used, the best case only demonstrates an absolute improvement in F1 score of 0.0124. Qian et al [60] explore a similar technique, examining both a *Twitter* user's previous posts and posts from other users that are semantically similar to aid in classification. Although they conclude these features help, they fail to achieve the best result in Precision or Recall and only achieve a 0.035 absolute increase in F1 performance over baseline models not utilising this data.

Weakly Supervised Classifiers

Building on the difficulty in obtaining large datasets, a few works attempt to improve upon the numerous supervised learning approache by utilising a weakly-supervised method.

Yuan et al [84] use the presence of certain hashtags within tweets to weakly identify them as discriminatory or not. These weak labels are used to prepare an embedding layer that is then combined with a LR model and training on a well-labelled dataset. The authors report an Accuracy of up to 0.910, however we would argue that the premise on which the weak labels are derived is faulty. The authors suggest that tweets with the hashtags *sexism* and *racism* are more likely to be discriminatory. This claim is not substantiated, and given results in other fields have show that those with prejudices are unlikely to identify as such [6], we would argue against the claim. Gao et al [23] adopt a similar approach, instead using a corpus of known hateful terms and slurs to weakly label tweets.

3.4 Evaluations

Our work attempts to replicate existing classifiers in the literature and evaluate performance on a new dataset. We summarise the few comparable works:

Wang [77] reproduces Zhang et al [85] but not with the specific intention of evaluating its performance in aggregate or against some benchmark. Instead the work analyses the decision making process of Zhang et al's [85] model, offering some insights for improving performance.

Klubicka & Fernandez [38] perform a similar reproduction of Waseem & Hovy [79] and compare with the original result citing similar results. They also evaluate performance of the same algorithm with different features concluding, upon achieving worse results, that n-gram features were most suitable for the classification of hate speech. It is worth noting however that their replication is an approximation of the original work based on several assumptions.

Although they do not reproduce any prior hate speech classifiers directly, Lee et al [40] examine the performance of several types of classifier against Founta et al's [20] dataset. Though no single model outperforms in all metrics, they note the improved performance of neural network based models over traditional methods and conclude that a GRU based RNN network with latent topic clustering (LTC) is their best performing model. We note that for the *hateful* label, performance in Lee et al [40] is universally poor for Recall and F1 scores, with an SVM based model dramatically outperforming others in Precision.

Gröndahl et al [33] reproduce four existing works ([4], [12], [82], [85]) and evaluates them against a combination of datasets. They find that when trained and tested on a new dataset, model performance is equivalent to each result, however when trained on one dataset and tested on another, performance drops significantly. They also conclude that in general, these models often conflate hateful and offensive (but not hateful) texts. Finally, they conduct various 'attacks' on existing models, attempting to reduce model performance by making small transformations to texts. They conclude that models are fragile to such attacks. Arango et al [3] also focus on precisely reproducing existing works; Agrawal & Awekar [2] and Badjatiya et al [4]. They evaluate the works on new datasets, using Wasseem & Hovy [79] to train both models and testing them on a dataset by Basile et al [5]. They cite reduced performance for both models, concluding that models do not generalise well, even when datasets are drawn from the same platform. This echoes Gröndahl et al's [33] earlier results. Arango et al [3] also identify important methodological issues with both reproduced works and report new results accounting for these issues.

3.5 Overview

From our review of the literature we can identify several trends in the work. Datasets focus on gathering texts from the platform *Twitter* (e.g [79], [12], [20]), which is not inherently limited but many of these corpa suffer from some form of bias. Follow up works have examined these biases (e.g [38], [65]) and some recent datasets attempt to account for some of the issues (e.g [20]). The range of classifiers and algorithms used is varied, with numerous attempts to improve and optimise results. Though not necessarily suffering from methodological issues, works can be inconsistent in the results they report (e.g [14]), and common metrics can be limited in communicating the true performance of hate speech classifiers. Although some works evaluate the claims of prior work (e.g [33]), with some identifying clear methodological issues (e.g [3]), there is a clear gap in reproducing results and verifying performance across new datasets. We hope our experiments will begin to fill this gap.

Chapter 4

Methodology

4.1 Dataset

From the available datasets in the field, we elect to conduct our experiments using a modified version of the dataset provided by Founta et al [20]. The dataset set is among the larger of the datasets in the literature and, as previously noted, is one of the few works that attempts to account of the methodological issues of pre-sampling.

The original dataset set consists of ≈100,000 tweet IDs from *Twitter*. Each ID, based on the original text of the tweet, is labelled as *one* of:

normal, spam, abusive, hateful

As we are concerned primarily with analysis of hate speech and differentiating it from offensive or abusive text, we condense the *normal* and *spam* labels into a single label: *neither*.

The final the dataset consists of 99996 texts labelled:

hateful (4965 - 4.97%), *abusive* (27150 - 27.15%), *neither* (67881 - 67.88%)

We apply no additional processing to the texts. For the remainder of this work, we will refer to this modified dataset as **HAN**.

4.2 Classifiers

The large collection of classifiers identified in the literature are reduced to a shortlist based on several exclusion criteria:

- **Produced Before 2015** - To ensure we are experimenting with modern, solutions, we exclude any works produced before 2015.
- **Non-English or Closed-Source Lexicons** - several works use lexicons and dictionaries that were either specific to non-English languages or are not otherwise precisely specified.
- **Non-Textual Features** - as previously discussed, several works use meta-data and non-text-based features to construct classifiers. Some works also use relations between the texts in the datasets. We deem any classifier utilising these features as out of scope.
- **Clarity of Results** - some works, when presenting multiple classifiers, are vague in their conclusion about the relative performance of the classifiers. These works were disregarded.

- **Clarity of Methodology -** some works are vague about the features or underlying algorithm used in the classifier, making it difficult to reproduce the work.

Where works report the results of more than one classifier (e.g [12]), we generally consider only the best performing model.

Baseline

Two works by Silva et al [69] and Mondal et al [49] utilise a simple syntactic model to attempt identify hate speech in a deterministic fashion. Though the model can evolve to incorporate new entries in its lexicons, the texts themselves do not alter performance overtime. Is it however, an interesting benchmark to compare advanced classifiers against and so we elected to reproduce this model.

Supervised Classifiers

Based on the outlined exclusion criteria, classifiers are shortlisted to 18 possibilities. **Table 4.1** groups and orders classifiers by their underlying algorithm, and presents the oldest papers in each group first:

Year	Authors	Classifier Type	Ref
2019	Mozafari et al	BERT	[50]
2017	Gambäck & Sikdar	CNN	[21]
2017	Park & Fung	CNN	[57]
2018	Zhang et al	CNN	[85]
2018	Zimmerman et al	CNN	[86]
2018	Watanabe et al	J48Graft	[81]
2015	Djuric et al	Regression (Logistic)	[14]
2016	Nobata et al	Regression (Vowpal Wabbit)	[53]
2017	Davidson et al	Regression (Logistic)	[12]
2018	Gaydhani et al	Regression (Logistic)	[24]
2017	Badjatiya et al	RNN	[4]
2015	Agarwal & Sureka	SVM	[1]
2016	Burnap & Williams	SVM	[8]
2016	Mehdad & Tetreault	SVM	[48]
2017	Malmasi & Zampieri	SVM	[46]
2018	Malmasi & Zampieri	SVM	[45]
2018	Robinson et al	SVM	[63]
2019	MacAvaney et al	SVM	[44]

Table 4.1: Shortlist of Works to Evaluate

From this shortlist, we filter down to a final collection of works to reproduce and evaluate against **HAN**. We do this on the basis of difficulty in reproducing; helped primarily by the provision of source code and we aim to use more recent works where possible and to select from a range of classifier types.

Regression

From the available Regression models, we reject Djuric et al [14] on the basis that it is the oldest of the selection and has no available source code. We reject Nobata et al [53] due to the complexity of its feature set and, due to lack of source code, difficulty in reproducing. Whilst source code was available for Davidson et al [12], we elected to reproduce Gaydhani et al [24] as the newer work. Although no source code is made available, the work describes the model in enough detail to adequately reproduce.

SVM

We reject Robinson et al [63] due to the difficulty in reproducing the features used without the original source code. It also directly compares itself to prior work by Zhang et al [85], and is out-performed in all experiments. We reject both Burnap & Williams [8], Mehdad & Tetreault [48] and MacAvaney et al [44] on the grounds of lack of source code and advanced features. We reject Agarwal & Sureka [1] due to the relative age of the paper and lack of source code. We reject Malmasi & Zampieri (2017) [46] on the basis that Malmasi & Zampieri (2018) [45] builds directly upon the earlier work and better details the data pre-processing steps. Malmasi & Zampieri (2018) [45] is also one of the newest of the identified SVM works and, although the work lacks source code, has a relatively simple set of features that can be easily reproduced.

CNN

Of the CNN-based works, Park & Fung [57] and Gambäck & Sikdar [21] are rejected on the basis of lack of source code. Zhang et al [85] and Zimmerman et al [86] both provide source code and were produced in the same year. We electe to use Zhang et al [85] on the basis that Gröndahl et al [33] also reproduce the work and therefore provide a point of comparison.

RNN

We only identified a single RNN-based paper, Badjatiya et al [4], and so elected to reproduce.

Others

Watanabe et al [81], Mozafari et al [50] are rejected on the basis that they did not provide source code and implement somewhat esoteric model types.

We arrive at a final set of works to reproduce:

Year	Authors	Classifier Type	Ref
2017	Mondal et al	Template / Syntactic	[49]
2018	Gaydhani et al	Regression (Logistic)	[24]
2018	Malmasi & Zampieri	SVM	[45]
2018	Zhang et al	CNN	[85]
2017	Badjatiya et al	RNN	[4]

Table 4.2: Final List of Works to Evaluate

4.3 Experiment Parameters

Template Baseline

For our template-based baseline we implement pre-processing steps that we believe best supports the model and report the following results:

- Precision, Recall and F1 scores of each label.
- Precision, Recall and F1 of the classifier calculated using Macro and Weighted Macro averages.
- A Micro / Accuracy score for the classifier.
- A confusion matrix for all labels.

Supervised Classifiers

For each of the four selected machine learning classifiers, we apply the following steps:

- Pass the classifier all texts in raw form and apply pre-processing as detailed in each individual work.
- Run a 10-fold cross validation with a random seed of 314. Due to the reduced frequency of the *hateful* label, we apply a stratified folding method to ensure representative distributions in each fold.
- Report the following results, averaged across the 10-folds using the mean:
  – Precision, Recall and F1 scores for each label.
  – Precision, Recall and F1 scores for the classifier calculated using Macro and Weighted Macro averages.
  – The Micro / Accuracy score for the classifier.
  – A confusion matrix for all labels.
  – Standard deviations for all values.

4.4 Template Baseline - Mondal et al

The simple syntax based model proposed by Mondal et al [49] functions by comparing sub-texts to one of two token patterns:

I <intensity> <user intent> <one word> people

OR

I <intensity> <user intent> <hate target>

The work notes that the *<intensity>* is optional. e.g *"I hate ... people"* is just as valid as *"I really hate ... people"*

Lexicons

The works uses four lexicons to determine the structures described. Three are detailed in the work, however the lexicon for the *<hate target>* token was scraped from *Hatebase*. We perform a similar scrape and as in the original work, only use vocabulary where the *average offensiveness* is greater than 50 (scale 0-100). The original work notes using 116 words from a total of 1078. A scrape of *Hatebase* at the time of writing, yields 784 words from a total of 1536, though their use in the original work is not clear, we include plurals and variants in these counts. The number of completely unique terms is likely to reduce after the stemming process, we do not report these final counts.

Pre-Processing

The original work does not detail any pre-processing steps, so we apply steps that we believe best support the model:

- Text is lowercased.
- Apostrophes, quotes, common HTML entities, emojis and other common punctuation are removed.
- Hashtags are expanded into individual words (as in [85]). Values not found in English dictionaries are preserved (without the hashtag symbol)
- White-space is normalised to single spaces and the texts tokenised.

- Specific tokens are filtered out or transformed (e.g "cant" is transformed into "can", "not").
- Tokens are stemmed using the default Porter Stemmer implementation in the *NLTK* Python library[1]. Both the library and the English stop word list are commonly used in the existing work.

We also lowercase, stem and tokenise all values in the four lexicons to maximise matches.

Model

The model is detailed well in the original work, however we still make some assumptions about its operation and implement a model we believe is mostly likely to detect hateful content.

Our implementation first searches each text for any instance of the token *"i"* and then search for each token type in the associated lexicon.

We make the assumption that for the <*intensity*> and <*user intent*> tokens, multiple instances of each type is valid. e.g *"I really really hate, loath and despise..."* (2 <*intensity*>, 3 <*user intent*>) is as equally valid as *"I really hate..."* (1 <*intensity*>, 1 <*user intent*>).

We also make an assumption about the filtering of *exclusion words* when identifying a <*hate target*> token. We assume that the existence of these words should be ignored at the start of a possible <*hate target*> token rather causing the match to fail. For example, our implementation considers *"I hate when ... people"* valid. There may be some small cases when this is not a incorrect assumption. As with the other tokens, we also ignore sequences of multiple words.

Texts with a matching sub-structure are labelled *hateful*, others *not*.

4.5 LR - Gaydhani et al

The work here reports an LR model as its best performing result. TF-IDF n-grams is used as the only feature set.

Whilst no source code was available to precisely reproduce the work, the authors do note that they utilise the *scikit-learn* Python library[2] and we believe sufficient detail is provided to reproduce the model.

Pre-Processing

The work details several pre-processing steps that we implement in the following order:

- Lowercase texts and remove URLs, mentions and the retweet prefix.
- Standardised white-space (inferred from *"remove space pattern"* in the original work).
- Removed stop words. We utilise the English corpora from *NLTK*.
- Stem texts using the default *NLTK* Porter Stemmer implementation.

Model

The paper details several points of configuration for the model. Using an example of a *scikit-learn* TF-IDF LR implementation presented in other work [40] we configure the TF-IDF vector layer with an n-gram range of 1-3 and a L2 normalisation. The LR layer is configured with an regularisation parameter (C) of 100, and the *liblinear* optimisation algorithm. We use the *scikit-learn* defaults for all other configuration.

[1]https://www.nltk.org/
[2]https://scikit-learn.org

4.6 SVM - Malmasi & Zampieri

The original work reports results of several stand-alone classifiers using a range of features, as well as the results of an ensemble method. We aim to reproduce the best performing stand-alone classifier: a SVM model utilising character based n-grams.

No source code is available for the work, so we build the model using *scikit-learn*. We acknowledge that due to lack of detail about exact implementation in the original work, our reproduction is only an approximation.

Pre-Processing

The work details some basic pre-processing steps that we reproduce: lowercasing, and removing mentions, URLs and emoji.

Model

We configure the feature vector with character based n-grams with a range of 2-4. We ensure these are extracted across word boundaries, as specified in the work. We approximate the classifier using the standard *scikit-learn* implementation of a linear SVM model.

4.7 CNN - Zhang et al

This work presents a CNN combined with a GRU network as its best performing model.

Source code was available for this work, however no precise examples are provided to run the best performing version of their classifier. Several assumptions therefore had to be made to reproduce the model and we note these below.

Pre-Processing

All pre-processing is performed by the source code. We ensure the *enchant* Python library[3] was installed with UK and US English dictionaries and we use the standard *NLTK* English stop word list.

Model

We configured the model exactly as detailed in the work, with the following assumptions:

- In the code it is possible to assemble the overall model in three ways. The configuration of the classifier as detailed in the work appears to only be supported by the method we utilise.
- For the GRU layer, we had to assume the setting of *return_sequences* and set it to *True*
- For the final softmax layer the paper details a regularisation step. Within the code it is possible to regularise both the kernel and the activity of the layer. Based on the *Tensorflow* documentation[4] we have assumed only the activity is regularised. The work does not specify the factor used for the regularisation, therefore we assume 0.01, which is the *Tensorflow* default.
- The source code provides an option to configure the model with a word distribution file, given there is no mention of this in the original work, this is excluded
- The source code allows the word embedding layer to be randomised, it is not clear in the work which setting was used, so we configure the setting to not apply any randomisation.

[3]https://pypi.org/project/pyenchant/
[4]https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/L1L2

- Finally, in the original version of the source code, tweets were tokenised on each alphabetic character. (e.g 'h','e','l','l','o'), we notice this makes both the filtering of stop words and the stemming of words, entirely redundant. We make a small alteration to the code to tokenise on contiguous alphabetic strings instead.

4.8 RNN - Badjatiya et al

This work presents a Long Short-Term Memory-based (LSTM) RNN, combined with Gradient Boosted Decision Trees (GBDTs) as its best performing model.

Full source code was available and we use other existing work as a guide to reproducing the model [3]. We apply some minor fixes and remove much of the unnecessary code.

Pre-Processing

All pre-processing is performed by the existing source code. We use the *NLTK* English stop word list.

Model

Although source code was available, some configuration was still necessary to reproduce the best performing model. Some of these were not detailed in paper, but an assumption based on examples in source code:

- We obtain and utilise the same pre-trained embedding
- The training batch size is set to 128
- The number of training epochs is set to 10
- Loss function is set to *categorical_crossentropy*
- Embedding size is set to 200
- Embedding weights are initialised randomly
- Optimiser algorithm is set to *adam*
- Learning of embeddings is enabled

Chapter 5

Results & Discussion

5.1 Dataset Observations

As an extension to our approximation of Mondal et al [49], we use the scraped *Hatebase* terms to make some observations about **HAN**. As previously discussed, we identify 784 *very* offensive terms in the *Hatebase* vocabulary. We examine **HAN** to identify the number of texts containing at least one of these across each label. Results are shown in **Table 5.1**.

Label	Contains Term	% of Label Freq.
hateful	856	17.24%
abusive	1436	5.29%
neither	1877	2.77%
Total	4169	4.17%

Table 5.1: Texts in **HAN** containing one of 784 *Hatebase* terms (average offensiveness > 50)

Here we can observe that less than 20% of the *hateful* texts in **HAN** contain words that could be considered *very* offensive hateful terms, and that $\approx 3.5\%$ of non-hateful texts do. We suggest this could contribute to the performance when classifying the *hateful* label and the confusion rates with other labels.

When we examine the dataset against the full list of 1536 *Hatebase* terms, without filtering based on the severity of the term (a 95.9% increase), we see a growth of around 55% in matching *hateful* and *neither* texts but a 210% increase in matching *abusive* texts (**Table 5.2**).

Label	Contains Term	% of Label Freq.	Relative Growth
hateful	1330	26.79%	55.37%
abusive	4460	16.43%	210.58%
neither	2942	4.33%	56.74%
Total	8732	8.73%	109.45%

Table 5.2: Texts in **HAN** containing one of 1536 *Hatebase* terms

This could be an intuitive result; we might assert that as the average offensiveness of a term decreases, the likelihood of it being considered *abusive* rather that *hateful* increases. However, this might be faulty assumption. Of the 1536 *Hatebase* terms, 619 do not have a valid average offensiveness rating, preventing them from appearing in the filtered list. Despite the lack of a rating these terms may still be deemed very offensive.

Overall we observe an imbalance of *Hatebase* terms within the dataset. This might suggest there are either labelling issues in the dataset, or that the content of *hateful* labels is more nuanced than simply using very offensive language. Irrespective, we posit that this imbalance might contribute to the accuracy issues consistently shown in classifier results.

5.2 Template Baseline - Mondal et al

The results for our approximation of Mondal et al [49] are shown in **Table 5.3** and **Table 5.4**.

In raw terms our approximation detects only 8 *hateful* tweets, 3 of which were false positives. Against our total dataset this represents a 0.008% match. Whilst a disappointing result nominally, this is approximately double the matching rate found in the original work which matched $\approx$ 20000 tweets on a sample of $\approx$ 500 million; a 0.004% rate. This increase in matches can, in part, be explained by the growth in the number of hate target terms available on *Hatebase*. Despite the very small Precision score for *hateful*, the algorithm still achieves good aggregate scores overall. This is indicative of the low false positive rate of *hateful* predictions, but is largely the result of the *hateful* label forming a relatively small component of the dataset. This result demonstrates a weakness in the way classifier performance is assessed, a Weighted Macro score is thought to account for an uneven label distribution in the dataset, but in this instance only the unweighted Macro aggregate hints at the overall performance of the model.

Due to the mis-representation of the true performance of our approximation in aggregate results, we will not include these results in our comparison of reproduced classifiers.

Although the intended goal of Silva et al [69] and Mondal et al [49], is not to identify all hate speech but to measure certain aspects of the hate speech phenomenon, we would begin to question the validity of the methodology. Although the **HAN** dataset was considerably smaller, the classification algorithm positively matched only 5 of 4965 *hateful* texts. If we assume conservatively that 50% of the *hateful* texts in **HAN** are mis-classified, it would still suggest that the trends identified in the original work are based on less than 0.1% of all the hate speech in their sample. We would therefore question any inferences based on these results about the comparative prevalence of hate speech and its targets.

		Precision	**Recall**	**F1**
Labels	*hateful*	0.625	0.001	0.002
	abusive ∪ *neither*	0.950	1.000	0.975
Averages	Macro	0.788	0.500	0.488
	Weighted Macro	0.934	0.950	0.926
	Micro / Accuracy	0.950	0.950	0.950

Table 5.3: Mondal et al [49] - Approximation metrics

T ↓ **P** →	*hateful*	%	*abusive* ∪ *neither*	%
hateful	5.000	0.001	4960.000	0.999
abusive ∪ *neither*	3.000	0.000	95028.000	1.000

Table 5.4: Mondal et al [49] - Approximation confusion matrix

5.3 LR - Gaydhani et al

The results for our reproduction of Gaydhani et al [24] are shown in **Table 5.5** and **Table 5.6**. Results for the original work are shown in **Table 5.7** and **Table 5.8**.

The original work is also evaluated on a *Twitter* dataset with labels:

hateful, offensive, clean

Exact distributions and the total number of texts is not detailed in the work. Whilst there may be subtle semantic differences between *offensive* texts in the original dataset, and *abusive* texts in the dataset we utilise [20], we will assume for the purposes of comparison that they are equivalent.

In general, the model performs poorly on **HAN**, with dips in performance in every metric compared to the original results. Classification of the *hateful* label shows the largest drop in performance with 40.31%, 68.95%, 58.62% relative reductions across Precision, Recall and F1 respectively. Examining the model as a whole, the Accuracy suffers a 5.94% fall in performance on **HAN**.

The original work suggests that performance of the model could be improved by obtaining more examples of *offensive* texts without hateful words. The decreased performance against **HAN**, in combination with our observations of the frequency of hateful terms across each label, supports this assertion. However, examining the confusion matrices of the original work (**Table 5.8**) and our reproduction (**Table 5.6**), we can see *abusive* texts are confused with *hateful* texts less frequently than *offensive* with *hateful* in the original work. Instead, confusion occurs significantly more frequently for *hateful* texts that in the original.

		Precision	σ	**Recall**	σ	**F1**	σ
Labels	*hateful*	0.561	0.024	0.298	0.021	0.389	0.020
	abusive	0.837	0.003	0.896	0.007	0.866	0.003
	neither	0.945	0.002	0.951	0.002	0.948	0.001
Averages	Macro	0.781	0.008	0.715	0.006	0.734	0.006
	Weighted Macro	0.897	0.002	0.903	0.001	0.898	0.001
	Micro/Accuracy	0.903	0.001	0.903	0.001	0.903	0.001

Table 5.5: Gaydhani et al [49] - Reproduction metrics

T ↓ **P** →	*hateful*	σ	%	σ	*abusive*	σ	%	σ	*neither*	σ	%	σ
hateful	148.000	10.440	0.298	0.021	186.000	11.662	0.376	0.024	161.000	13.077	0.326	0.026
abusive	66.000	10.440	0.025	0.004	2433.000	17.664	0.896	0.007	214.000	10.817	0.079	0.004
neither	49.000	6.481	0.007	0.001	285.000	12.042	0.042	0.002	6452.000	13.191	0.951	0.002

Table 5.6: Gaydhani et al [49] - Reproduction confusion matrix

	Precision	**Recall**	**F1**
hateful	0.94	0.96	0.94
offensive	0.96	0.93	0.94
clean	0.96	0.98	0.97
Micro / Accuracy	0.96	0.96	0.96

Table 5.7: Gaydhani et al [49] - Original metrics

Truth ↓ Prediction →	*hateful*	*offensive*	*neither*
hateful	0.965	0.021	0.014
offensive	0.048	0.926	0.026
neither	0.010	0.013	0.977

Table 5.8: Gaydhani et al [49] - Original confusion matrix

5.4 SVM - Malmasi & Zampieri

The results for our reproduction of Malmasi & Zampieri [45] are shown in **Table 5.9** and **Table 5.10**.

We note that when running our implementation of the classifier, *scikit-learn* reports that for the 2nd fold of the cross-validation, the *liblinear* algorithm fails to converge, recommending that the number of possible iterations be increased. Given that the original work does not specify an iteration limit, we report results with this anomaly included.

The original work is also evaluated on a *Twitter* dataset of 14509 texts with labels:

hateful (2399 - 16.5%), *offensive* (4836 - 33.3%), *ok* (7274 - 50.1%)

Again, we assume that *offensive* is equivalent to *abusive*.

The original work only reports an Accuracy / Micro result for the classifier of 0.78. Against this metric, our reproduction performs significantly better; reporting 0.869 - a 11.46% relative increase.

The original work notes that Accuracy consistently increases as the training set size increases. They note the increase slows as they approach their maximal data size, but did not plateau. Our results are consistent with this observation. The original work reports an Accuracy increase of 20% from 0.65 - 0.78, with a data increase of approximately 1500 - 14000, a 933% growth. Our with our results demonstrate an increase from 0.78 - 0.86 with a data increase of approximately 14000 - 100000, a 714% growth. However, whilst **HAN** is significantly larger, the ratios of each label are not consistent with the original work, so the scale in Accuracy can not be expected to correlate completely with the original results.

		Precision	σ	**Recall**	σ	**F1**	σ
Labels	*hateful*	0.357	0.015	0.298	0.019	0.325	0.016
	abusive	0.808	0.004	0.828	0.006	0.818	0.004
	neither	0.925	0.002	0.927	0.003	0.926	0.002
Averages	Macro	0.697	0.005	0.684	0.006	0.690	0.006
	Weighted Macro	0.865	0.002	0.869	0.002	0.867	0.001
	Micro/Accuracy	0.869	0.002	0.869	0.002	0.869	0.002

Table 5.9: Malmasi & Zampieri [45] - Reproduction metrics

T ↓ **P** →	*hateful*	σ	%	σ	*abusive*	σ	%	σ	*neither*	σ	%	σ
hateful	147.000	9.274	0.298	0.019	170.000	11.916	0.343	0.024	178.000	13.000	0.359	0.026
abusive	135.000	10.100	0.050	0.004	2248.000	15.652	0.828	0.006	330.000	13.964	0.122	0.005
neither	130.000	12.767	0.019	0.002	365.000	13.379	0.054	0.002	6292.000	22.316	0.927	0.003

Table 5.10: Malmasi & Zampieri [45] - Reproduction confusion matrix

5.5 CNN - Zhang et al

The results for our reproduction of Zhang et al [85] are shown in **Table 5.11** and **Table 5.12**.

We note that in running out implementation, *scikit-learn* reports an error that the stop words used are inconsistent with the pre-processing applied to the texts. This is because the code applies the stemming algorithm prior to the filtering of stop words and so many of the stop words are incompatible with the stemmed texts. Stemming the stop words before use would resolve this issue, but we assume this error is also present in the original work.

The original work evaluates the classifier against 7 datasets, none directly comparable to ours, however the best performance of the model is run against 24783 tweets labelled:

hate (1430 - 5.77%), *non-hate* (23353 - 94.22%)

This dataset is based on Davidson et al [12]. As the authors (of [85]) note, the dataset was originally labelled with an additional *abusive* label that is condensed into the *non-hate* label for the purposes of their experimentation.

The work reports an Accuracy score of 0.940, against which our reproduction suffers only a 2.66% relative performance drop to 0.915.

The authors also report that of the errors their classifier make, 94% are made predicting *hate* texts. Compared with the confusion rates of our reproduction, the ratio of errors reduces slightly, with false predictions for *hateful* texts accounting for 75% of the total errors. This implies that for hate speech in **HAN**, the model has more difficulty identifying non-hateful texts. Our observations about the distribution of hateful terms in the dataset would support this. Interestingly, our reproduction on **HAN**, performs similarly to original experiments on an alternative *hate* / *non-hate* dataset where authors report an Accuracy of 0.92 and a *hate* error ratio of 70%.

		Precision	σ	**Recall**	σ	**F1**	σ
Labels	*hateful*	0.677	0.067	0.253	0.051	0.363	0.047
	abusive	0.848	0.011	0.928	0.014	0.886	0.004
	neither	0.952	0.004	0.959	0.005	0.955	0.002
	Macro	0.825	0.021	0.713	0.014	0.735	0.016
	Weighted Macro	0.910	0.003	0.915	0.003	0.907	0.003
	Micro/Accuracy	0.915	0.003	0.915	0.003	0.915	0.003

Table 5.11: Zhang et al [85] - Reproduction metrics

T $\downarrow$ **P** $\rightarrow$	*hateful*	σ	%	σ	*abusive*	σ	%	σ	*neither*	σ	%	σ
hateful	125.000	25.318	0.253	0.051	194.000	24.940	0.393	0.050	175.000	11.446	0.354	0.023
abusive	40.000	23.917	0.015	0.009	2518.000	37.616	0.928	0.014	155.000	20.322	0.057	0.007
neither	23.000	8.426	0.003	0.001	258.000	28.601	0.038	0.004	6506.000	33.061	0.959	0.005

Table 5.12: Zhang et al [85] - Reproduction confusion matrix

5.6 RNN - Badjatiya et al

The results for our reproduction of Badjatiya et al [4] are shown in **Table 5.13** and **Table 5.14**.

Experiments in the original work are conducted using a dataset of 16000 tweets labelled as:

sexist (3383 - 21.14%), *racist* (1972 - 12.32%), *neither* (10645 - 66.53%)

The work reports only the Weighted Macro metric so, although the datasets are incompatible for direct comparison, metrics per-label are not available. The work reports a Weighted Macro result of 0.93 for Precision, Recall and F1. Our reproduction against **HAN** results in an 3.44% relative increase in performance.

		Precision	σ	**Recall**	σ	**F1**	σ
Labels	*hateful*	0.851	0.015	0.747	0.024	0.795	0.016
	abusive	0.929	0.004	0.956	0.004	0.942	0.003
	neither	0.983	0.001	0.980	0.001	0.982	0.001
Averages	Macro	0.921	0.006	0.894	0.008	0.906	0.006
	Weighted Macro	0.962	0.002	0.962	0.002	0.962	0.002
	Micro/Accuracy	0.962	0.002	0.962	0.002	0.962	0.002

Table 5.13: Badjatiya et al [4] - Reproduction metrics

T $\downarrow$ **P** $\rightarrow$	*hateful*	σ	%	σ	*abusive*	σ	%	σ	*neither*	σ	%	σ
hateful	370.000	11.489	0.747	0.024	82.000	10.000	0.167	0.020	42.000	6.633	0.086	0.013
abusive	46.000	5.831	0.017	0.002	2594.000	9.950	0.956	0.004	73.000	6.245	0.027	0.002
neither	19.000	2.646	0.003	0.000	114.000	6.403	0.017	0.001	6653.000	6.557	0.980	0.001

Table 5.14: Badjatiya et al [4] - Reproduction confusion matrix

We acknowledge that Arango et al [3] observe methodological issues in Badjatiya et al [4]. In order to account for these issues in our evaluation, we scale the performance of our reproduction according to the same performance drop observed by Arango et al [3].

We did note an inconsistency in the Micro / Accuracy results reported by Arango et al [3] and, upon examining the source code for the work, conclude these results are actually Weighted Macro metrics. **Table 5.15** shows the results reported by Arango et al [3] and a performance scaling factor for Precision, Recall, and F1 calculated by taking the mean of the reductions in the two metrics provided in Arango et al. We apply this scaling factor to our results (**Table 5.16**).

Even when accounting for the methodological issues, our reproduction still performs favourably on **HAN**. Against the original work we see gains in all Weighted Macro scores. Further comparison and discussion of our results will refer to this scaled result.

		Precision	**Recall**	**F1**
Arango et al Reproduction	Weighted Macro	0.946	0.946	0.946
	Macro	0.937	0.926	0.931
Arango et al Adjustment	Weighted Macro	0.823	0.821	0.807
	Macro	0.823	0.821	0.807
	Mean Performance Ratio	0.870	0.806	0.819
	Percentage Change	-12.96%	-19.40%	-18.09%
Badjatiya et al Orignal	Weighted Macro	0.930	0.930	0.930
	Weighted Macro Scaled	0.809	0.750	0.762

Table 5.15: Results for Arango et al's [3] reproduction of Badjatiya et al [4], including the adjustment for methodological issues. We produce a scaling factor for the performance loss and apply it to Badjatiya et al's original result.

		Precision	**Recall**	**F1**
Labels	*hateful*	0.740	0.602	0.651
	abusive	0.809	0.770	0.772
	neither	0.855	0.790	0.804
Averages	Macro	0.802	0.721	0.742
	Weighted Macro	0.837	0.775	0.788
	Micro/Accuracy	0.837	0.775	0.788

Table 5.16: Badjatiya et al [4] - Reproduction metrics, scaled according to **Table 5.15**

5.7 Classifier Comparison

Of the four classifies we reproduce we observe drops in aggregate performance over their original results for Zhang et al [85] and Gaydhani et al [24], though only the latter saw a significant drop (> 5%). Malmasi & Zampieri [45] and Badjatiya et al [4] saw performance gains, but only the former saw a significant improvement.

Table 5.17 shows that despite a small drop in performance compared to original results, Zhang et al [85] is the best performing model in 5 of 7 aggregate metrics, only falling behind Badjatiya et al [4] in Macro

Recall and F1.

Turning to our focus of hate speech, no model performs exceptionally well. Our results show that only Badjatiya et al [4] achieves a Recall or F1 score above 0.5 for the *hateful* label, With three of four models achieving a Precision score above 0.5.

Comparing the confusion rates for each model, **Table 5.17** shows very similar confusion rates across models when the *hateful* label is the true annotation. False *hateful* predictions when *abusive* and *neither* are the truth diverge more substantially, but given there are considerably more instances of these texts, the confusion for *hateful* is not order of magnitudes different across each model. It is clear that for **HAN**, the *hateful* texts are not consistently mistaken for *abusive*, as one might expect, but instead the confusion is fairly well distributed between *abusive* and *neither*. Note, we exclude Badjatiya et al [4] from our results here due to the lack of confusion data after accounting for methodological issues.

Metric	**Model**	**Precision**	**Recall**	**F1**
***hateful* Label**	Gaydhani et al	0.561	0.298	0.389
	Malmasi & Zampieri	0.357	0.298	0.325
	Zhang et al	0.677	0.253	0.363
	Badjatiya et al (Scaled)	**0.740**	**0.602**	**0.651**
Macro	Gaydhani et al	0.781	0.715	0.734
	Malmasi & Zampieri	0.697	0.684	0.690
	Zhang et al	**0.825**	0.713	0.735
	Badjatiya et al (Scaled)	0.802	**0.721**	**0.742**
Weighted Macro	Gaydhani et al	0.897	0.903	0.898
	Malmasi & Zampieri	0.865	0.869	0.867
	Zhang et al	**0.910**	**0.915**	**0.907**
	Badjatiya et al (Scaled)	0.837	0.775	0.788
Micro/Accuracy	Gaydhani et al	0.903	0.903	0.903
	Malmasi & Zampieri	0.869	0.869	0.869
	Zhang et al	**0.915**	**0.915**	**0.915**
	Badjatiya et al (Scaled)	0.837	0.775	0.788

Table 5.17: Comparison of reproduced classifiers across all metrics. Best result within each metric is bold.

Truth	*hateful*	*hateful*	*hateful*	*abusive*	*neither*
Prediction	*hateful*	*abusive*	*neither*	*hateful*	*hateful*
Gaydahani et al	148	186	161	66	49
Malmasi & Zampieri	147	170	178	135	130
Zhang et al	125	194	175	40	23

Table 5.18: Comparison of *hateful* label confusion. Excludes Badjatiya et al.

Chapter 6

Conclusions & Future Work

From our results, we can conclude that performance of models is, at best, inconsistent when trained against new datasets. Two of the works we reproduce, Gaydahani et al [24] and Zhang et al [85], explicitly report tuning parameters of their model. This we suggest, contributes to changes in performances when replicated with new datasets. Badjatiya et al [4] also use some values that may have been derived as a result of tuning on their original dataset, though we do report improved performance here. Our results differ somewhat to results from Gröndahl et al [33] where they find original results are largely equivalent to performance on a new dataset. Instead, our results begin to suggest issues around the generalisation of models and their potential for practical application. Following from Gröndahl et al [33], future work might aim to examine generalisation of models and their original results by training models on their originally reported datasets and then testing against a new dataset (assuming label compatibility). This would continue to explore how applicable models would be in practice with data gathered from new sources and demographics. In cases where the tuning methodology was outlined, as in Gaydahani et al [24], future work might also examine if improved results can be achieved with new datasets when applying the same tuning methodology. This would suggest whether the drops in performance web observe are fundamental to the combination of features and algorithms applied to the dataset, or if it is simply a lack of fine adjustment for the data.

When comparing aggregate metrics, our results are relatively consistent with those found by Lee et al [40], who experimented with the Founta et al [20] dataset. We also conclude that neural models tend toward better performance than traditional methods, and that LR outperforms SVM.

Examining only the *hateful* label, most of the works we reproduce outperform the equivalent model from Lee et al [40] in Recall and F1, but equivalent Precision scores are far lower. Most notably for their character-based SVM model where they report a Precision of 0.805 for the *hateful* label, in contrast to 0.298 for our approximation of Malmasi & Zampieri [45]. This would suggest there is not something fundamentally difficult about the detection of hate speech in the **HAN** dataset but rather our reproduced models do not perform well.

We also observe inconsistencies in the reporting of results across works, this causes difficulty in making like-for-like comparisons of performance between models. Future work and perhaps the wider field should attempt to establish a fixed framework for reporting results, to aid in comparison. We find, in agreement with earlier work (e.g [64], [78]), that some aggregate statistics are not useful for the purposes for analysing the performance of a hate speech classifier. As previously discussed, the Micro/Accuracy measurement for our approximation of Mondal et al reports an Accuracy of 0.950 outperforming all the reproduced supervised learning methods. Clearly this metric fails to reflect that the algorithm only identifies 0.1% of all hate speech texts; an unacceptable failure rate for practical implementation. We acknowledge that this imbalance in the metric is due to the relative small percentage of *hateful* texts in **HAN** but this generally reflects the real-world instances of hateful content [20]. Waseem [78] makes a recommendation of to use Weighted

Macro scores to compare classifiers but, with the exception of Badjatiya et al [4], the works we evaluate do not use this metric. In addition, the Weighted Macro metric is intended to prevent poor performance on a small frequency label from unduly impacting the performance of the whole classifier. We argue that an unweighted Macro is a more suitable appropriate as it more clearly reflects the importance of hateful texts despite them appearing relatively infrequently in **HAN**. However, we would still suggest that future work should look to develop a new metric that takes into account the importance of identifying particular labels and better reflect this when reporting the aggregate performance a model.

Appendix A

Source Code

A subset of our source code is detailed below. The listing contains the code used to run and evaluate each classifier, but does not include the code for the classifiers themselves. The source code in its entirety is available from:

https://github.com/sampayne/hate-speech

```
#[..REMOVED IMPORT STATEMENTS FOR PURPOSE OF LISTING HERE..]

INDENT = '    '

def main():

    #print(api.load("word2vec-google-news-300", return_path=True))
    #print(api.load("glove-twitter-25", return_path=True))
    #return

    print('Started:', datetime.now())

    folds = 10
    seed = 314

    hateful_label = 'hateful'
    abusive_label = 'abusive'

    labels = [
        hateful_label,
        abusive_label
    ]

    default_label = 'neither'

    raw_data = load_tweets('founta-dataset.csv', labels, default_label)

    labels.append(default_label)

    total_length = len(raw_data)

    print('dataset distribution')

    for label in labels:

        count = len([t for t, l in raw_data if l == label])

        print('{' + label + '},' + str(count) + ',' + percent((count / total_length) * 100,
                                                              dp=2))
```

```
print('total,' + str(total_length))

baseline_combined_label = abusive_label + ' or ' + default_label

mondal_model = mondal(raw_data=raw_data, target_label=hateful_label, fallback_label=
                                              baseline_combined_label,
                                              all_original_labels=labels)

mondal_results = evaluate_baseline(model=mondal_model, labels=[hateful_label,
                                              baseline_combined_label])

mondal_results.print_all()

print('Number of k-folds:', folds)
print('Random seed:', seed)

gaydhani_model= Model(gaydhani(raw_data))

gaydhani_results = evaluate_model(model=gaydhani_model, labels=labels, folds=folds, seed
                                              =seed)

gaydhani_results.print_all()

malmasi_model = Model(malmasi(raw_data))

malmasi_results = evaluate_model(model=malmasi_model, labels=labels, folds=folds, seed=
                                              seed)

malmasi_results.print_all()

zhang_model = Model(zhang(raw_data))

zhang_results = evaluate_model(model=zhang_model, labels=labels, folds=folds, seed=seed)

zhang_results.print_all()

badjatiya_model = Model(badjatiya(raw_data=raw_data, labels=labels))

badjatiya_results = evaluate_model(model=badjatiya_model, labels=labels, folds=folds,
                                              seed=seed)

badjatiya_results.print_all()

#Scaled results based on performance drop observed in Arango et al
precision_scale_factor = mean([82.3/94.6, 81.6/93.7])

recall_scale_factor = mean([82.1/94.6, 68.9/92.6])

f1_scale_factor = mean([80.7/94.6, 73.1/93.1])

badjatiya_original = 0.930

print(badjatiya_model.name, 'original weighted-macro results + scaled')
print('precision,recall,f1')
print('original,' + percent(badjatiya_original) + ',' + percent(badjatiya_original) + ',
                                              ' + percent(badjatiya_original))
print('scaled,' + percent(badjatiya_original * precision_scale_factor) + ',' + percent(
                                              badjatiya_original * recall_scale_factor)
                                              + ',' + percent(badjatiya_original *
                                              f1_scale_factor))

scaled_badjatiya_results = badjatiya_results.scale(precision_scale_factor,
                                              recall_scale_factor,f1_scale_factor)
```

```
    scaled_badjatiya_results.print_all()

    print('aggregate comparison')
    print('model,metric,precision,recall,f1')
    gaydhani_results.print_aggregate()
    malmasi_results.print_aggregate()
    zhang_results.print_aggregate()
    scaled_badjatiya_results.print_aggregate()

    print('{' + hateful_label + '} comparison')
    print('model,precision,recall,f1')
    gaydhani_results.print_label(hateful_label)
    malmasi_results.print_label(hateful_label)
    zhang_results.print_label(hateful_label)
    scaled_badjatiya_results.print_label(hateful_label)

    print('Completed:', datetime.now())

class PrecisionRecallF1:

    def __init__(self, precision, recall, f1):

        self.precision = MeanStandardDeviation(precision)
        self.recall = MeanStandardDeviation(recall)
        self.f1 = MeanStandardDeviation(f1)

    def scale(self, precision_scale_factor, recall_scale_factor, f1_scale_factor):

        return PrecisionRecallF1([self.precision.scale(precision_scale_factor)],
                                 [self.recall.scale(recall_scale_factor)],
                                 [self.f1.scale(f1_scale_factor)])

    def __repr__(self):

        return str(self.precision) + ',' + str(self.recall) + ',' + str(self.f1)


class MeanStandardDeviation:

    def __init__(self, list):

        if len(list) == 1:
            self.mean = list[0]
            self.standard_deviation = None
        else:
            self.mean = mean(list)
            self.standard_deviation = stdev(list)

    def scale(self, factor):

        return self.mean * factor

    def __repr__(self):
        if self.standard_deviation is None:
            return str(percent(self.mean)) + ',{}'
        else:
            return str(percent(self.mean)) + ',{' + str(percent(self.standard_deviation)) +
                                                  '}'

class Evaluation:

    def __init__(self,
                 name,
                 labels,
                 per_label_metrics,
                 micro_mean:PrecisionRecallF1,
                 macro_mean:PrecisionRecallF1,
                 weighted_mean:PrecisionRecallF1,
                 confusion_matrix_mean,
                 confusion_matrix_ratios):

        self.labels = labels
        self.name = name
        self.per_label_metrics = per_label_metrics
        self.micro_mean = micro_mean
        self.macro_mean = macro_mean
        self.weighted_mean = weighted_mean
        self.confusion_matrix_mean = confusion_matrix_mean
        self.confusion_matrix_ratios = confusion_matrix_ratios

    def print_label(self, target_label):

        for label, result in self.per_label_metrics:

            if target_label == label:
                print(self.name + ',' + str(result))

    def print_aggregate(self):

        print(self.name + ',micro/accuracy,' + str(self.micro_mean))

        print(self.name + ',macro,' + str(self.macro_mean))

        print(self.name + ',weighted macro,' + str(self.weighted_mean))


    def print_all(self):

        print(self.name, '- metrics')

        print('metric,precision,sig,recall,sig,f1,sig')

        for label, result in self.per_label_metrics:

            print('{' + label + '},' + str(result))

        print('macro,' + str(self.macro_mean))

        print('weighted macro,' + str(self.weighted_mean))

        print('micro/accuracy,' + str(self.micro_mean))

        self.print_confusion_matrix(mean=self.confusion_matrix_mean, ratios=self.
                                              confusion_matrix_ratios, labels=self.
                                              labels)

    def print_confusion_matrix(self, mean, ratios, labels):

        if len(mean) == 0 or len(ratios) == 0:
            return

        print(self.name, '- confusion matrix')

        print(',' + ' #,sig,%,sig,'.join(labels) + ' #,sig,%,sig')

        for i, label in enumerate(labels):

            mean_values = mean[i]
            ratio_values = ratios[i]
            line = []

            for j in range(len(labels)):

                line.append(str(mean_values[j]))
                line.append(str(ratio_values[j]))

            print(label + ',' + ','.join(line))

    def scale(self, precision_scale_factor, recall_scale_factor, f1_scale_factor):

        print('scaling', self.name)
        print(',precision,recall,f1')
        print('ratio,' + percent(precision_scale_factor), percent(recall_scale_factor),
                                                    percent(f1_scale_factor))
        print('%change,' + percent((precision_scale_factor-1)*100, dp=2), percent((
                                                    recall_scale_factor-1) * 100, dp=2),
                                                    percent((f1_scale_factor-1) * 100, dp=
                                                    2))

        scaled_label_metrics = []

        for label, metric in self.per_label_metrics:
            scaled_label_metrics.append((label, metric.scale(precision_scale_factor,
                                                        recall_scale_factor,
                                                        f1_scale_factor)))

        return Evaluation(name= self.name + ' Scaled',
                          labels=self.labels,
                          per_label_metrics=scaled_label_metrics,
                          micro_mean=self.micro_mean.scale(precision_scale_factor,
                                                            recall_scale_factor,
                                                             f1_scale_factor),
                          macro_mean=self.macro_mean.scale(precision_scale_factor,
                                                            recall_scale_factor,
                                                             f1_scale_factor),
                          weighted_mean=self.weighted_mean.scale(precision_scale_factor,
                                                            recall_scale_factor,
                                                             f1_scale_factor),
                          confusion_matrix_mean=[],
                          confusion_matrix_ratios=[])

class Model:

    def __init__(self, model):

        name, classifier, tweets = model

        self.name = name
        self.classifier = classifier
        self.tweets = tweets

def evaluate_baseline(model, labels):

    name, predictions, truth = model

    print('Evaluating', name + '...')

    all, micro, macro, weighted, confusion_matrix, confusion_ratios = evaluate_fold(
                                                    predictions=predictions, truth=truth,
                                                    labels=labels)

    results = calculate_means(name=name, all=[all], micro=[micro], macro=[macro], weighted=[
                                                    weighted], confusion_matricies=[
                                                    confusion_matrix], confusion_ratios=[
                                                  confusion_ratios], labels=labels)

    print('Finished evaluating', name)

    return results

def evaluate_model(model:Model, labels, folds, seed):

    print('Evaluating', model.name + '...')

    splitter = sklearn.model_selection.StratifiedKFold(n_splits=folds, shuffle=True,
                                                  random_state=seed)

    all = []
    micro = []
    macro = []
    weighted = []
    confusion_matricies = []
    confusion_ratios = []

    X = np.array([x for x, y in model.tweets])
    y = np.array([y for x, y in model.tweets])

    i = 1

    classifier = model.classifier

    for train_index, test_index in splitter.split(X, y):
        print(INDENT, str(i) + 'th fold...')

        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        print(INDENT, INDENT, 'Fitting...')

        classifier.fit(X_train, y_train)

        print(INDENT, INDENT, 'Testing...')

        predictions = classifier.predict(X_test)

        print(INDENT, INDENT, 'Evaluating...')

        fold_metrics = evaluate_fold(predictions=predictions, truth=y_test, labels=labels)

        fold_all, fold_micro, fold_macro, fold_weighted, fold_cm, fold_cr = fold_metrics

        all.append(fold_all)
        micro.append(fold_micro)
        macro.append(fold_macro)
        weighted.append(fold_weighted)
        confusion_matricies.append(fold_cm)
        confusion_ratios.append(fold_cr)

        i = i + 1

    model_results = calculate_means(name=model.name, all=all, micro=micro, macro=macro,
                                                  weighted=weighted, confusion_matricies=
                                                  confusion_matricies, confusion_ratios=
                                                  confusion_ratios, labels=labels)

    print('Finished evaluating', model.name)

    return model_results
```

```
def evaluate_fold(predictions, truth, labels):

    all = evaluate_metrics_for_average(predictions, truth, labels, 'all')
    micro = evaluate_metrics_for_average(predictions, truth, labels, 'micro')
    macro = evaluate_metrics_for_average(predictions, truth, labels, 'macro')
    weighted = evaluate_metrics_for_average(predictions, truth, labels, 'weighted')
    confusion_matrix = metrics.confusion_matrix(y_true=truth, y_pred=predictions, labels=
                                              labels)
    confusion_ratios= metrics.confusion_matrix(y_true=truth, y_pred=predictions, labels=
                                              labels, normalize='true')

    return all, micro, macro, weighted, confusion_matrix, confusion_ratios

def calculate_means(name, all, micro, macro, weighted, confusion_matricies, confusion_ratios
                                              , labels):

    per_label = calculate_mean_for_label_fold_results(all, labels)

    micro_mean = calculate_mean_for_fold_results(micro)

    macro_mean = calculate_mean_for_fold_results(macro)

    weighted_mean = calculate_mean_for_fold_results(weighted)

    mean = calculate_mean_confusion_matrix(confusion_matricies=confusion_matricies, labels=
                                              labels)

    ratios = calculate_mean_confusion_matrix(confusion_matricies=confusion_ratios, labels=
                                              labels)

    return Evaluation(name= name,
                      labels=labels,
                      per_label_metrics=per_label,
                      micro_mean=micro_mean,
                      macro_mean=macro_mean,
                      weighted_mean=weighted_mean,
                      confusion_matrix_mean=mean,
                      confusion_matrix_ratios=ratios)


def calculate_mean_confusion_matrix(confusion_matricies, labels):

    final_matrix = []

    for _ in labels:

        inner = []

        for _ in labels:

            inner.append([])

        final_matrix.append(inner)

    for fold in confusion_matricies:

        for outer in range(len(fold)):

            for inner in range(len(fold[outer])):

                final_matrix[outer][inner].append(fold[outer][inner])

    for outer in range(len(final_matrix)):

        for inner in range(len(final_matrix[outer])):
            final_matrix[outer][inner] = MeanStandardDeviation(final_matrix[outer][inner])

    return final_matrix

def percent(value, dp = 3):

    parts = str(round(value, dp)).split('.')

    int = parts[0]
    dec = '0'

    if len(parts) == 2:
        dec = parts[1]

    return int  + '.' + dec.ljust(dp, '0')

def calculate_mean_for_label_fold_results(kfold_results, labels):

    all_precision = {}
    all_recall = {}
    all_f1 = {}

    for label in labels:
        all_precision[label] = []
        all_f1[label] = []
        all_recall[label] = []

    for precision, recall, f1 in kfold_results:

        for i, label in enumerate(labels):

            all_precision[label].append(precision[i])
            all_recall[label].append(recall[i])
            all_f1[label].append(f1[i])

    final_results = []

    for label in labels:

        final_results.append((label, PrecisionRecallF1(all_precision[label], all_recall[
                                                         label], all_f1[label])))

    return final_results

def calculate_mean_for_fold_results(kfold_results):

    precision = [precision for precision, _, _ in kfold_results]
    recall = [recall for _, recall, _ in kfold_results]
    f1 = [f1 for _, _, f1 in kfold_results]

    return PrecisionRecallF1(precision, recall, f1)

def evaluate_metrics_for_average(predictions, test_labels, all_labels, metric):

    final_metric = metric;

    if metric == 'all':
        final_metric = None

    precision = metrics.precision_score(y_true=test_labels, y_pred=predictions, labels=
                                          all_labels, average=final_metric)
    recall = metrics.recall_score(y_true=test_labels, y_pred=predictions, labels=all_labels,
                                           average=final_metric)
    f1 = metrics.f1_score(y_true=test_labels, y_pred=predictions, labels=all_labels, average
                                                                =final_metric)

    return precision, recall, f1

def load_tweets(data_set_name, target_labels, default_label):

    raw_data = []

    with open(data_set_name, newline='') as csvfile:

        reader = csv.reader(csvfile, delimiter='\t', quotechar='"')

        for row in reader:

            assert(len(row) == 3)

            text, label, confidence = row

            if text == 'text':
                #Skip the header row
                continue;

            assert (len(text.strip()) > 0)
            assert (len(label.strip()) > 0)

            label = label.lower()

            if not label in target_labels:
                raw_data.append([text, default_label])
            else:
                raw_data.append([text, label])

    return raw_data



if __name__ == "__main__":
    main()
```

Bibliography

[1] Swati Agarwal and Ashish Sureka. “Using KNN and SVM Based One-Class Classifier for Detecting Online Radicalization on Twitter”. In: *Distributed Computing and Internet Technology*. Ed. by Raja Natarajan, Gautam Barua, and Manas Ranjan Patra. Cham: Springer International Publishing, 2015, pp. 431–442.

[2] Sweta Agrawal and Amit Awekar. “Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms”. In: *CoRR* abs/1801.06482 (2018).

[3] Aymé Arango, Jorge Pérez, and Barbara Poblete. “Hate Speech Detection is Not as Easy as You May Think: A Closer Look at Model Validation”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Sigir’19. Paris, France: Association for Computing Machinery, 2019, pp. 45–54.

[4] Pinkesh Badjatiya et al. “Deep Learning for Hate Speech Detection in Tweets”. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW ’17 Companion. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 759–760.

[5] Valerio Basile et al. “SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter”. In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 54–63.

[6] Eduardo Bonilla-Silva and Tyrone A. Forman. “"I Am Not a Racist But...": Mapping White College Students’ Racial Ideology in the USA”. In: *Discourse & Society* 11.1 (2000), pp. 50–85.

[7] Pete Burnap and Matthew L. Williams. “Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making”. In: *Policy & Internet* 7.2 (2015), pp. 223–242.

[8] Pete Burnap and Matthew L. Williams. “Us and them: identifying cyber hate on Twitter across multiple protected characteristics”. In: *EPJ Data Science* 5.1 (2016), p. 11.

[9] Dominic Casciani. *Prosecutors clarify offensive online posts law*. Dec. 2012. URL: https://www.bbc.co.uk/news/uk-20777002. [Accessed: 10th September 2020].

[10] Office of Communications. *Children and parents: media use and attitudes report 2016*. Nov. 2016. URL: https://www.ofcom.org.uk/research-and-data/media-literacy-research/childrens/children-parents-nov16. [Accessed: 10th September 2020].

[11] Elizabeth Culliford and Katie Paul. *With fact-checks, Twitter takes on a new kind of task*. May 2020. URL: https://www.reuters.com/article/us-twitter-factcheck/with-fact-checks-twitter-takes-on-a-new-kind-of-task-idUSKBN2360U0. [Accessed: 10th September 2020].

[12] Thomas Davidson et al. *Automated Hate Speech Detection and the Problem of Offensive Language*. 2017.

[13] Fabio Del Vigna et al. “Hate me, hate me not: Hate speech detection on facebook”. In: *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*. 2017, pp. 86–95.

[14] Nemanja Djuric et al. "Hate Speech Detection with Comment Embeddings". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15 Companion. Florence, Italy: Association for Computing Machinery, 2015, pp. 29–30.

[15] Facebook. *Facebook Community Standards - Hate Speech*. URL: https://en-gb.facebook.com/communitystandards/hate_speech. [Accessed: 10th September 2020].

[16] Facebook. *Why am I seeing a warning before I can view a photo or video on Facebook?* URL: https://www.facebook.com/help/814083248683500. [Accessed: 10th September 2020].

[17] Elise Fehn Unsvåg and Björn Gambäck. "The Effects of User Features on Twitter Hate Speech Detection". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 75–85.

[18] Paula Fortuna and Sérgio Nunes. "A Survey on Automatic Detection of Hate Speech in Text". In: *ACM Comput. Surv.* 51.4 (July 2018).

[19] Antigoni Maria Founta et al. "A Unified Deep Learning Architecture for Abuse Detection". In: *Proceedings of the 10th ACM Conference on Web Science*. WebSci '19. Boston, Massachusetts, USA: Association for Computing Machinery, 2019, pp. 105–114.

[20] Antigoni-Maria Founta et al. "Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior". In: *CoRR* abs/1802.00393 (2018).

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

[22] Lei Gao and Ruihong Huang. "Detecting Online Hate Speech Using Context Aware Models". In: *CoRR* abs/1710.07395 (2017).

[23] Lei Gao, Alexis Kuppersmith, and Ruihong Huang. "Recognizing Explicit and Implicit Hate Speech Using a Weakly Supervised Two-path Bootstrapping Approach". In: *CoRR* abs/1710.07394 (2017).

[24] Aditya Gaydhani et al. "Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach". In: *CoRR* abs/1809.08651 (2018).

[25] Ona de Gibert et al. "Hate Speech Dataset from a White Supremacy Forum". In: *CoRR* abs/1809.04444 (2018).

[26] Njagi Dennis Gitari et al. "A lexicon-based approach for hate speech detection". In: *International Journal of Multimedia and Ubiquitous Engineering* 10.4 (2015), pp. 215–230.

[27] HM Government. *Communications Act 2003*. July 2003. URL: https://www.legislation.gov.uk/ukpga/2003/21. [Accessed: 10th September 2020].

[28] HM Government. *Crime and Disorder Act 1998*. July 1998. URL: https://www.legislation.gov.uk/ukpga/1998/37. [Accessed: 10th September 2020].

[29] HM Government. *Criminal Justice and Immigration Act 2008*. Feb. 2008. URL: https://www.legislation.gov.uk/ukpga/2008/4. [Accessed: 10th September 2020].

[30] HM Government. *Malicious Communications Act 1988*. July 1988. URL: https://www.legislation.gov.uk/ukpga/1988/27. [Accessed: 10th September 2020].

[31] HM Government. *Public Order Act 1986*. Nov. 1986. URL: https://www.legislation.gov.uk/ukpga/1986/64. [Accessed: 10th September 2020].

[32] HM Government. *Racial and Religious Hatred Act 2006*. Feb. 2006. URL: https://www.legislation.gov.uk/ukpga/2006/1. [Accessed: 10th September 2020].

[33] Tommi Gröndahl et al. "All You Need is "Love": Evading Hate Speech Detection". In: *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. AISec '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2–12.

[34] Eric Heinze. *Hate speech and democratic citizenship.* Oxford University Press, 2017.

[35] Snap Inc. *Community Guidelines.* URL: https://www.snap.com/en-GB/community-guidelines. [Accessed: 10th September 2020].

[36] Instagram. *Instagram Community Guidelines.* URL: https://help.instagram.com/477434105621119/. [Accessed: 10th September 2020].

[37] Leo Kelion. *Children see 'worrying' amount of hate speech online.* Nov. 2016. URL: https://www.bbc.co.uk/news/technology-37989475. [Accessed: 10th September 2020].

[38] Filip Klubicka and Raquel Fernández. "Examining a hate speech corpus for hate speech detection and popularity prediction". In: *CoRR* abs/1805.04661 (2018).

[39] Rohan Kshirsagar et al. "Predictive Embeddings for Hate Speech Detection on Twitter". In: *CoRR* abs/1809.10644 (2018).

[40] Younghun Lee, Seunghyun Yoon, and Kyomin Jung. "Comparative Studies of Detecting Abusive Language on Twitter". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2).* Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 101–106.

[41] LinkedIn. *LinkedIn Professional Community Policies.* May 2020. URL: https://www.linkedin.com/help/linkedin/suggested/34593/linkedin-professional-community-policies. [Accessed: 10th September 2020].

[42] S. Liu and T. Forss. "New classification models for detecting Hate and Violence web content". In: *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K).* Vol. 01. 2015, pp. 487–495.

[43] Donna Lu. *WhatsApp restrictions slow the spread of fake news - but don't stop it.* Sept. 2019. URL: https://www.newscientist.com/article/2217937-whatsapp-restrictions-slow-the-spread-of-fake-news-but-dont-stop-it/. [Accessed: 10th September 2020].

[44] Sean MacAvaney et al. "Hate speech detection: Challenges and solutions". In: *Plos One* 14.8 (Aug. 2019), pp. 1–16.

[45] Shervin Malmasi and Marcos Zampieri. "Challenges in discriminating profanity from hate speech". In: *Journal of Experimental & Theoretical Artificial Intelligence* 30.2 (2018), pp. 187–202.

[46] Shervin Malmasi and Marcos Zampieri. "Detecting Hate Speech in Social Media". In: *CoRR* abs/1712.06427 (2017).

[47] Binny Mathew et al. "Thou Shalt Not Hate: Countering Online Hate Speech". In: *Proceedings of the International AAAI Conference on Web and Social Media* 13.01 (July 2019), pp. 369–380.

[48] Yashar Mehdad and Joel Tetreault. "Do Characters Abuse More Than Words?" In: *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue.* Los Angeles: Association for Computational Linguistics, Sept. 2016, pp. 299–303.

[49] Mainack Mondal, Leandro Araújo Silva, and Fabrício Benevenuto. "A Measurement Study of Hate Speech in Social Media". In: *Proceedings of the 28th ACM Conference on Hypertext and Social Media.* Ht '17. Prague, Czech Republic: Association for Computing Machinery, 2017, pp. 85–94.

[50] Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. *A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media.* 2019.

[51] B. Sri Nandhini and J. I. Sheeba. "Cyberbullying Detection and Classification Using Information Retrieval Algorithm". In: *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015).* Icarcset '15. Unnao, India: Association for Computing Machinery, 2015.

[52] Casey Newton. *Facebook once built a phone that could only be used by right-handed people.* Feb. 2020. URL: https://www.theverge.com/facebook/2020/2/27/21154578/facebook-phone-gfk-yves-behar-steven-levy-inside-story-highlights. [Accessed: 10th September 2020].

[53] Chikashi Nobata et al. "Abusive Language Detection in Online User Content". In: *Proceedings of the 25th International Conference on World Wide Web*. Www '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 145–153.

[54] Catherine O'Regan. "Hate Speech Online: an (Intractable) Contemporary Challenge?" In: *Current Legal Problems* 71.1 (Dec. 2018), pp. 403–429.

[55] United Nations: OHCHR. *International Convention on the Elimination of All Forms of Racial Discrimination*. Dec. 1965. URL: https://www.ohchr.org/en/professionalinterest/pages/cerd.aspx. [Accessed: 10th September 2020].

[56] United Nations: OHCHR. *International Covenant on Civil and Political Rights*. Dec. 1966. URL: https://www.ohchr.org/en/professionalinterest/pages/ccpr.aspx. [Accessed: 10th September 2020].

[57] Ji Ho Park and Pascale Fung. "One-step and Two-step Classification for Abusive Language Detection on Twitter". In: *CoRR* abs/1706.01206 (2017).

[58] Pinterest. *Pinterest Community Guidelines*. URL: https://policy.pinterest.com/en-gb/community-guidelines. [Accessed: 10th September 2020].

[59] Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. "Effective hate-speech detection in Twitter data using recurrent neural networks". In: *Applied Intelligence* 48.12 (2018), pp. 4730–4742.

[60] Jing Qian et al. "Leveraging Intra-User and Inter-User Representation Learning for Automated Hate Speech Detection". In: *CoRR* abs/1804.03124 (2018).

[61] Reddit. *Reddit Content Policy*. URL: https://www.redditinc.com/policies/content-policy. [Accessed: 10th September 2020].

[62] Reddit. *Update to Our Content Policy*. June 2020. URL: https://www.reddit.com/r/announcements/comments/hi3oht/update_to_our_content_policy/. [Accessed: 10th September 2020].

[63] David Robinson, Ziqi Zhang, and Jonathan Tepper. "Hate Speech Detection on Twitter: Feature Engineering v.s. Feature Selection". In: *The Semantic Web: ESWC 2018 Satellite Events*. Ed. by Aldo Gangemi et al. Cham: Springer International Publishing, 2018, pp. 46–49.

[64] Björn Ross et al. "Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis". In: *CoRR* abs/1701.08118 (2017).

[65] Maarten Sap et al. "The risk of racial bias in hate speech detection". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1668–1678.

[66] Carla Schieb and Mike Preuss. "Governing hate speech by means of counterspeech on Facebook". In: June 2016.

[67] Anna Schmidt and Michael Wiegand. "A Survey on Hate Speech Detection using Natural Language Processing". In: *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1–10.

[68] Crown Prosecution Service. *Guidelines on prosecuting cases involving communications sent via social media*. Aug. 2018. URL: https://www.cps.gov.uk/legal-guidance/social-media-guidelines-prosecuting-cases-involving-communications-sent-social-media. [Accessed: 10th September 2020].

[69] Leandro Silva et al. *Analyzing the Targets of Hate in Online Social Media*. 2016.

[70] Adriana Stephan. *Comparing Platform Hate Speech Policies: Reddit's Inevitable Evolution*. July 2020. URL: https://fsi.stanford.edu/news/reddit-hate-speech. [Accessed: 10th September 2020].

[71] Nadine Strossen. *Hate: why we should resist it with free speech, not censorship*. Oxford University Press, 2020.

[72] Stéphan Tulkens et al. "A Dictionary-based Approach to Racism Detection in Dutch Social Media". In: *CoRR* abs/1608.08738 (2016).

[73] Twitter. *Twitter Hateful conduct policy.* URL: https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy. [Accessed: 10th September 2020].

[74] European Union. *Electronic Commerce Directive 200.* June 2000. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32000L0031&from=EN. [Accessed: 10th September 2020].

[75] European Union. *Framework Decision on combating certain forms and expressions of racism and xenophobia by means of criminal law.* Nov. 2008. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32008F0913&from=en. [Accessed: 10th September 2020].

[76] European Union. *The EU Code of conduct on countering illegal hate speech online.* May 2016. URL: https://ec.europa.eu/info/policies/justice-and-fundamental-rights/combatting-discrimination/racism-and-xenophobia/eu-code-conduct-countering-illegal-hate-speech-online_en. [Accessed: 10th September 2020].

[77] Cindy Wang. "Interpreting Neural Network Hate Speech Classifiers". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2).* Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 86–92.

[78] Zeerak Waseem. "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.* Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 138–142.

[79] 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.* San Diego, California: Association for Computational Linguistics, June 2016, pp. 88–93.

[80] Zeerak Waseem et al. "Understanding Abuse: A Typology of Abusive Language Detection Subtasks". In: *CoRR* abs/1705.09899 (2017).

[81] H. Watanabe, M. Bouazizi, and T. Ohtsuki. "Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection". In: *IEEE Access* 6 (2018), pp. 13825–13835.

[82] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. "Ex Machina: Personal Attacks Seen at Scale". In: *CoRR* abs/1610.08914 (2016).

[83] YouTube. *Hate Speech Policy.* URL: https://support.google.com/youtube/answer/2801939. [Accessed: 10th September 2020].

[84] Shuhan Yuan, Xintao Wu, and Yang Xiang. "A Two Phase Deep Learning Model for Identifying Discrimination from Tweets." In: *EDBT.* 2016, pp. 696–697.

[85] Ziqi Zhang, David Robinson, and Jonathan Tepper. "Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network". In: *The Semantic Web.* Ed. by Aldo Gangemi et al. Cham: Springer International Publishing, 2018, pp. 745–760.

[86] Steven Zimmerman, Udo Kruschwitz, and Chris Fox. "Improving hate speech detection with deep learning ensembles". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* 2018.