Automatic detection of hate speech on Twitter

Ben Stevenson (2227777s)

April 23, 2020

NB: This paper uses terms that some readers may find offensive. Please read at your own discretion.

ABSTRACT

Online hate speech is an on-going issue to social media sites and can have negative impacts on peoples mental health. There are many challenges with regards to detecting hate speech online, with the main problem being the small discrepancy between defining hate speech and language that is deemed offensive but not hate. Automatic text classification can be used to address this issue. This paper addresses the issue of hate speech by using a handcrafted feature approach and a BERT model approach. We chose to use a dataset containing approximately 25,000 tweets which were pre-labelled as hate, offensive and neither hate nor offensive. We propose four different approaches; Text-based, Handcrafted features, BERT embeddings fed into a classical model and BERT fine-tuning. The Text-based approach is used as a baseline performance for text classification due to a lack of novel implementation. The handcrafted feature approach involves a novel implementation of 13 feature engineered vectors. To target hate speech, the handcrafted features were specifically designed to improve the detection of hate speech, such as syntactic structure and counting the occurrence of hate uni-grams and bi-grams. Using the handcrafted features on the logistic regression model, we improved the hate class prediction by 12% from the text-based approach. Therefore we deem the feature implementation successful. However, overall the best performing model was the BERT fine-tuning approach which achieved a 0.77 macro F1 score and a hate class F1 score of 0.45.

1. INTRODUCTION

Social media has become the most important hub for business, communication and marketing available in today's society. Over 55% of the world's population have constant access to internet use, with North America and Europe having a penetration rate (percentage of population) of 89.4% and 87.7% [28] respectively. Furthermore, there are over 2.3 billion monthly internet users in Asia, where consequently, the majority of social media content circulates. Statistics collected in 2019 show that nearly 3.5 billion internet users have signed up to at least one social media platform [15], meaning that people from all around the world have access to these platforms.

Social media platforms follow a simple structure of allowing users to post, react and message privately, therefore, permitting freedom to users to post and message what they wish. With such a global user base, there is inevitably a widespread opinion in political, racial, religious, sexual and ethical views. Unfortunately, this leads to harmful and abusive content posted publicly and privately online. We argue that Twitter, for example, has issues with users creating multiple new accounts quickly and easily and with little or no personal information. Many users continually make multiple accounts, and therefore, it is hard to reduce the prevalence of harmful and abusive content, i.e. hate speech. This project will approach the issue of automatically detecting hate speech on Twitter using text classification methods.

1.1 Motivation

The motivation for this project stems from the stigma surrounding social media use, especially the rise in mental health issues. Hate speech on social media needs to be regulated more effectively to help reduce the negative impacts of being online.

A study, conducted in the United States by Hinduja et al. [10], found that 60% of students in the study had been a victim of cyberbullying. Therefore, having an immediate impact on their ability to learn and to feel safe while at school. According to research by Hinduja and Patchin [11], students exposed to cyberbullying, as a direct result of the hate-speech they read on social media, are statistically more likely to develop a mental health illness in their lifetime and two times more likely to attempt suicide. John et al. [14] produced a systematic review of results and data from 25 articles related to the impact of cyberbullying. The data contained 115,056 children or young adult participants. Within the study, they noted that 44,526 experienced a negative influence from social media use. Furthermore, John et al. also concluded that victims of cyberbullying were over two times more likely to self-harm than non-victims and two times more likely to exhibit suicidal behaviours.

1.2 Statement of problem

Twitter uses an automatic detection system for hate speech to be manually reviewed. As stated in April 2019, out of all tweets reported as hate, Twitter proactively sourced 38% [9] using their detection software. Therefore 62% of abusive tweets that are detected require Twitter users to report the misconduct manually [24], potentially exposing vulnerable people to harmful content and on occasion, extreme opinions. Hence, the problem exists that to combat the prevalence of hate speech online there is still a long way to go. This paper will detail our approach to improving the text classification of hate speech.

Davidson et al. [4] discussed a common issue with automatic hate speech detection in prior work. They outlined that hate speech was being incorrectly conflated with offensive language. However, to correctly separate hate speech from offensive language, a universal definition is required to which one does not currently exist. This is a problem as law in countries such as the United Kingdom and Germany state that the act of hate speech is illegal, but offensive language is not.

The difference between the two definitions is minute, and therefore detecting the two as individual classes is a complex classification task. It is essential for social media sites that enforce strict hate speech regulations to be able to differentiate. Twitter defines their hate speech policy to be:
"You may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease." [13].

Davidson et al. defined hate to be *"language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group"*. The definition was used to create a Twitter dataset containing 24,783 tweets of hate, offensive but not hate and neither hate nor offensive language. We refer to the dataset as HateOffensive from this point on. They constructed a classification model to make predictions on the HateOffensive dataset. The results showed that their model achieved a 61% rate in predicting hate speech, therefore indicating that a further 39% were incorrectly classified. This is the best performing classification model for this multi-class dataset to date.

The HateOffensive dataset has been widely used, and many attempts have been made to improve on the results obtained by Davidson et al.. Fahr et al. [1], MacAvaney et al. [19] and Founta et al. [7] have all reported results similar to Davidson et al.'s. The results reported by [1, 7, 19] provide detail for the overall performance of their models on the HateOffensive dataset. Given the imbalance of the HateOffensive data, classification models are heavily skewed to predicting non-hate speech. This produces results that can often mask the performance of an individual class. For example, Davidson et al. state the precision and recall scores, 0.44 and 0.61, respectively, of their model to emphasise the difficulty in detecting hate speech. They also report the F1 score, a harmonic metric for the overall performance, to be 0.9. The F1 score does not represent that the hate class in the dataset is only correctly predicting 61% of tweets.

1.3 Contribution statement

This paper positions itself to focus on the detection of hate speech. We approach hate speech text classification using four classification approaches. These models include classic models such as Logistic Regression, Support Vector machine and a Neural Network, as standard models and also in combination with handcrafted features in a feature union [20] and a more modern model called BERT [6]. We propose a novel handcrafted feature approach with 13 features which have not previously been used together in order to improve on the detection of hate speech, from the classical approaches. We focus the features around how hate can be understood from text.

Using BERT, we implement two approaches: BERT embeddings fed into a classic model and BERT fine-tuning. BERT produces state-of-the-art classification performances, and we optimise its performance to obtain the best metrics for the hate class. We show how BERT embeddings compare against a classical approach for text classification tasks and how BERT fine-tuning approach can be used in conjunction with a classification layer to detect hate speech.

1.4 Outline

- Background: Section 2 - Explains the technologies and algorithms used throughout the research to provide the user with a better understanding.

- Related Work: Section 3 - Discusses how previous research has approached the problems in detecting online hate while addressing limitations to current work.

- Hypothesis: Section 4 - List of hypothesis and aims that were set out during the research. We make a hypothesis on the syntactic structure of hate speech and discuss how to achieve the hypothesis. We also set out overall goals for our models based on the performance of previous work.

- Model Frameworks: Section 5 - Explains the four text classification approaches used throughout the research.

- Experimental Setup: Section 6 - Details the overall process for how the research projected was conducted. We talk about the extraction and manipulation of the data and the frameworks used to combine the classification models.

- Results: Section 7 - Reports the performance scores for each model and makes an analysis of the performance scores. We discuss the reason for particular results.

- Conclusion & Future Work: Section 8 - Summarise the research paper process and results and provide possible directions for future work.

2. BACKGROUND

Section 2.1 begins by discussing the current industry standard for detecting hate speech online. Section 2.2 lists three datasets commonly used in related work. Section 2.3 discusses the types of models that we used for classification and also state-of-the-art models using the HateOffensive dataset. Section 2.4 provides an overview of BERT. Finally, Section 2.5 describes the process of part-of-speech tagging which is used as part of our handcrafted feature approach.

2.1 Current Online Procedures

Facebook and Twitter (Twitter was discussed in Section 1.2) have previously announced that they use text classification tools to help prevent abuse online. Facebook has recently opened its abuse data to the public. This data contains information about the amount of content that their detection algorithm finds and how much of the found content was detected by themselves or reported by Facebook users. In July, August and September of 2019, Facebook acted on 3.2 million items under the classification of "Bullying and Harassment". However, 84% of the content was reported by users before being flagged by Facebook's detection system [26]. Facebook states it needs people to report bad behaviour online due to the challenges of detecting abuse online [26].

2.2 Hate Speech Datasets

This section lists three commonly used datasets containing hate-related tweets and comments that are publicly available. We used the HateOffensive dataset as we believed it provided the best real-life representation of the ratio of hate to offensive tweets. It also did not specify the type of hate, which we felt was important due to the wide variety of hate found online. Other datasets commonly used are:

1. Waseem and Hovy [27] - This dataset contains 20,947 tweets labelled as: Sexist (13%), Racist (1%), Neither (84%) and Both Racist and Sexist (1%). They used feminists and anti-racism activists to annotate the dataset. The annotators were allowed the option to skip tweets if they were non-English or considered noise.

2. Stormfront [5] - The Stormfront dataset contains posts from the Stormfront forum, which was a popular racial hate site. The dataset contained 10,568 sentences labelled as; Hate (11%), Non-hate (86%), Relation (2%), Skip (1%). The 'relation' represents sentences that could be classified as hate when provided with a relational context.

3. TRAC [16] - This dataset contains 1,253 tweets labelled as; Non-aggressive (38%), overtly aggressive (29%) and Covertly aggressive (33%). The tweets were either English and Hindi.

These datasets contain more than two classes to represent hate speech and non-hate speech. Hate speech text classification is a task that can contain multiple facets, therefore it is called multi-class classification. The HateOffensive dataset is a multi-class dataset.

2.3 Classic Text Classifiers

Before the rise of machine learning algorithms, text classification was computed manually. The process involved assigning a label from a set of categories to an array of text documents. This process was expensive due to labour costs and highly time-consuming. When computing power and machine learning algorithms advanced, text classification became automated, drastically speeding up the classification process. Now, models can make trained predictions on thousands to millions of documents with zero human cost. Text classification plays an essential role in Natural Language Processing (NLP). This section goes into further detail about the four text classification models used throughout the research. We use the following classifiers for multiple approaches in our methodology as these approaches have been used in state-of-the-art classification tasks.

Support Vector Machine

The support vector machine (SVM) [3] is a supervised learning algorithm frequently used in machine learning tasks. An SVM is used for classification and regression tasks; therefore, classification specific variants of the SVM have been developed: Support Vector Classification (SVC). According to Cortes and Vapnik [3], the SVM finds a hyperplane that best separates the labelled data into different sectors. Moreover, whilst the SVM was designed for binary classification, it has since been developed to handle multi-class classification tasks. We will use the support vector classification model called LinearSVC, which approaches multi-class text classification using a one-vs-the-rest algorithm [29]. The LinearSVC can be optimised on parameters such as *penalty, loss, random_state, max_iter and C.*

Logistic Regression

Logistic Regression [8] is a classification model based on the statistical model of the logistic function. It estimates the relationship between an independent variable and a dependent variable, usually binary. Similar to the SVM the logistic regression is a classification tool optimised for binary classification and to handle multi-class classification, the model also applies the one-vs-the-rest algorithm.

The logistic function is classed as a sigmoid function. A sigmoid function (Equation 1) takes any real value between 0 and 1. The formula follows with equation 2 where t is set to a linear function in a univariate regression model. This is substituted into the original sigmoid equation (Equation 1) to create the logisitic equation (Equation 3)

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{1}$$

$$t = \beta_0 + \beta_1 x \tag{2}$$

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{3}$$

Logistic regression has 15 parameters that can be optimised for the training data. This is a task specific process.

Multi-layered perceptron

The third classic text classification model we used in our approach was the Multi-layered perceptron (MLP); it is a form of the artificial neural network developed in the 1980s. A perceptron is a neural network algorithm popular in supervised classification which consists of three layers of non-linearly activating nodes [18]; input, hidden and output layer. Each layer is linked to the next, feeding decision-based predictions until an output is made. The model was chosen to demonstrate how neural networks perform in text classification, as they are becoming increasingly effective. We use this classifier from the sklearn python library, which lists 23 parameters to train the MLP.

2.4 BERT

BERT, **B**i-directional **E**ncoder **R**epresentations of **T**ransformers, is a contextual based neural network model [6]. BERT is pre-trained on a large corpus of Wikipedia and BookCorpus (3,300 million words) which contains words, common subwords such as prefix's and suffixes and individual characters. Previous models, such as Continuous bag-of-words, used a uni-direction function that only reads text from left-to-right. The bi-directionality of BERT enables the model to learn the context in two directions. This is especially beneficial when BERT encounters two words that are spelt the same but have different meanings such as "Bank" (Riverbank or money bank).

The text input to BERT is tokenised, which is the process of separating input into individual word pieces, prior to training. BERT tokenising is unique to itself because of the way it splits words into subwords. A word is broken down into subwords when it does not exist within the vocabulary.

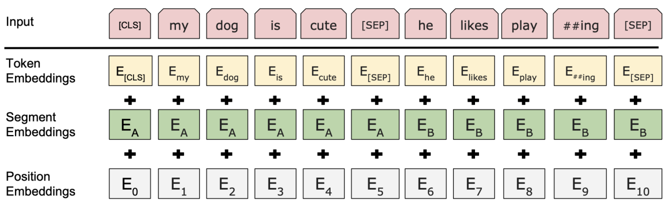

Figure 1: A representation of the input to BERT following the pre-trained tasks

It will then look for the largest subword within the vocabulary. For example BERT applies a wordpiece tokenisation such that "playing" → "play" + "##ing" and "going" → "go" + "##ing". This process allows for unseen words to be broken down into smaller word pieces. This will addresses a word mismatch between the pre-trained data BERT is trained on and the hate speech corpus for our fine-tuning approach.

Since the development of BERT, architectures have been developed to aid BERT classification. The most popular architectures used in NLP are the Pytorch Huggingface and Tensorflow libraries. These architectures enable a variety of different language modelling approaches., such as embedding extraction and fine-tuning. Figure 1 shows how the BERT embeddings are formed from the input text. The input layer can also be fed into a predictive BERT classification model, which is called BERT fine-tuning. BERT can be fine-tuned by altering its hundreds of layers to match the dataset. We will approach the problem using BERT embeddings and BERT fine-tuning.

2.5 Part-of-speech Tagging

Part-of-Speech (POS) tagging [21] is a software library which takes a text input and outputs a part-of-speech tag. These tags include word types such as adverbs, verbs, pronouns, nouns, adjectives and many others—the part of speech tagging used in this paper uses a set of 38 different tags. The tags are specific to the context of the text. For example, a verb can be defined as "third-person singular present", "past tense", "present participant" or "base form". POS tagging is useful for text classification models as it improves the understanding of words with contextual tagging. If one word appears twice in a sentence with two different meanings, the POS tagging will acknowledge this difference.

3. RELATED WORK

Section 3.1 details models designed to improve the detection of hate speech or obfuscated text on social media. Section 3.2 describes current classification approaches which have been tested against the HateOffensive dataset.

3.1 Detection algorithms for common social media problems

Automatic detection of abusive content is a difficult task for many reasons. Online content comes in many different forms with informal or formal language being the most inclusive categorisation. Within the subject of informal comments, language obfuscation is a commonly used method of making text informal. Language obfuscation is a method used on raw text to make it more complicated to human readers and machine detection to identify its meaning correctly. Obfuscation has many forms, which are continually adapting. A common obfuscation trick is a purposeful misspelling; such as "you" spelt as "u", or the "N" word spelt with different suffix endings such as "er", "a", "ah" which are all used in different contexts and may also confuse classification methods. Another language obfuscation trick is the use of numbers in place of "look-a-like" letters, the letter "l" is often replaced with the number "1" or using the number "8" to represent the string "ate", for words such as "hate". Previous work by Suk Lee et al. [17] looked to detect obfuscated abusive text using unsupervised learning of abusive terms. They approach the issue using word2vec's skip-gram, cosine similarity and gadgets used to filter abusive text such as abbreviations, mixed-languages, and words with special characters. We expect BERT to detect abusive words that have been obfuscated, using its wordpiece tokenisation, However, BERT is trained on a set amount of vocabulary, and therefore some uncommon abusive terms may be undetectable **(Limitation 1)**.

Another reason text is hard to detect correctly is due to its context. In today's society, companions use abusive terms to show humour and to have banter. Sarcasm also creates a context issue. Sarcasm has many forms, and it can even be tricky for manual detection as people's understanding of sarcasm varies, especially in written form. Frequently used hate terms and phrases can be used in a sarcastic manner with no harmful intent but just a dark sense of humour. Previous work by Poria et al. [22] proposed a deep convolutional neural network using sentiment, emotion and personality-based features for sarcasm detection. They used sentiment detection to find contradictions within the text such as "I loved the movie, I left during the first interval". This can clearly be split into a positive sentiment followed by a negative sentiment; therefore, they classify that it is likely to be sarcastic.

The African American community frequently uses the "N" word to show love or appreciation for each other, but the same word can be used to show hatred towards the community. To be able to correctly classify content containing such language would require a deep understanding of the relationship between the users. Understanding where the relationship between two parties lies is a complex problem. A detection algorithm would not be able to differentiate between a social-media friend and a "real-life" friend. Céceillon et al. [12] used a graphical method of finding the relationship between the original poster and the commenter to understand the semantics. They found that the social graph feature was useful for detecting cyberbullying. We argue that the ability to understand the relationship of "real-life" friends correctly is an impossible task and therefore 100% accuracy could not truly occur. Additionally, the HateOffensive dataset does not contain any metadata regarding Twitter followers, so therefore we are unable to investigate social media relationships.

3.2 Classification models on the HateOffensive dataset

Silva et al. [25] investigated the langague pattern in detecting targeted hate speech. They proposed that targeted hate speech had a particular syntactic structure. The structure followed four basic rules; I <intensity> <user intent> <hate target>. For example, "I really (*intensity*) hate (*user intent*) black people (*hate target*)". This research and prior knowledge of social media abuse formed the foundation for one of the hypothesis.

Fohr et al. [1] implemented two versions of BERT; BERT embeddings fed into a neural network and the BERT fine-tuning approach. The process involved the data being stripped of all punctuation, except ",",'.","?","!" , along with usernames, retweet meta-data and hashtags split into separate words, such as "#Dontberude" to "Dont be rude". They detail the parameters used for fine-tuning as *epochs = 3, learning rate =* $2e^5$*,max sequence length = 256, batch size = 16*. For the word embedding approach, they feed the BERT embeddings into a bi-directional Long Short-Term Memory (Neural network) and a convolutional neural network. MacAvaney et al. [19] used multiple SVM's in a multiple-view stacked method to approach hate speech detection. To create the view-classifier, each feature was fitted with a Linear SVM classifier, and then the multiple view-classifiers were combined into another linear SVM classifier. They applied this model to the HateOffensive dataset and received a precision score for each class; hate was 49.6%, offensive was 95.9%, neither was 82.5%. MacAvaney also recorded a 0.77 score for macro F1.

In 2017, Davidson et al. used a Logistic Regression model to achieve a state-of-the-art prediction for the hate class. The model included multiple features such as text n-gram, POS-tag n-gram, readability metrics and a binary count for Twitter data such as hashtags, emojis, retweet and mention data and URLs to aid detection. This model yielded a micro F1 average of 0.90. Figure 2 presents the recall score for each class in the diagonal boxes. Within this paper, our aim is to improve on results from this model.

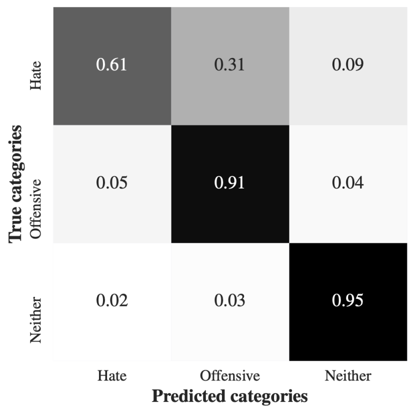

Figure 2: Davidson et al. results in a confusion matrix

The precision score presented by MacAvaney et al. for the hate class is over 45% lower than the precision for the offensive class. Furthermore, the 61% recall for the hate class produced by Davidson et al. was 30% worse than for the offensive class. This is an indication of how the imbalance in the HateOffensive dataset impacts the detection of hate **(Limitation 2)**.

In this paper, we discuss how there is little standardisation of performance metrics used throughout previous work and that some metrics are chosen so that performance appears better, particularly micro average F1 vs macro average F1. This causes issues with comparability. **(Limitation 3)**

We argue that the importance of detecting the hate class has been undervalued in previous work (**Limitation 4**). The research by Founta et al. shows no indication that specific attention was given to improving the detection of hate speech by providing no performance scores for the individual classes. MacAvaney et al., detail their individual class results in the form of precision in their appendix, whilst detailing the best overall performing models in the results. Fahr et al. state the performance of predicting the hate class, but throughout the paper there is little emphasis on ensuring hate speech detection is improved from Davidson et al.'s performance, although they note that future work would look into improving the hate metrics.

3.3 Summary of Limitations

From the overview on the related work, we identified four limitations of hate speech text classification on social media:
Limitation 1: BERT model is pre-trained on data that is unlikely to contain hate terms.
Limitation 2: The imbalance of the dataset causes a disadvantage to the approaches predicting hate.
Limitation 3: The lack of standardised metrics used in hate speech classification makes comparing results a confusing task.
Limitation 4: There is an inherent lack of importance to detecting the hate class
Limitation 5: There is a lack of transparency within some research papers when detailing hate class metrics.

In Section 5 and 6 we discuss how this paper proposes to address these limitations. Section 5.3 addresses **Limitation 1** and Section 6 address **Limitation 2** and **Limitation 3**. **Limitation 4** is addressed throughout the paper and Section 7 discusses **Limitation 5**.

4. HYPOTHESIS & AIMS

Before conducting the research, a hypothesis and a set of aims were laid out. Initially, two aims were set out for the overall performance of the research. Aim 1 was chosen as we believed the implementation of handcrafted features would improve on Macaveney et al.'s approach and aim 2 is what we set out to achieve in our research.

1. **Aim**: Improve on the state-of-the-art macro average F1 score achieved by a classical model created by Macaveney et al. [19] (macro average F1: 0.77).

2. **Aim**: Achieve a better F1 score for the hate class than the score achieved by Davidson et al. [4] (Hate F1: 0.51).

Hypothesis 1 took inspiration from Silva et al. and a preconceived idea of the syntactic structure of hate. Hypothesis 2 and 3 were assumptions made about the format of hate speech. We detail our hypothesis and how we plan to address them:

1. **Hypothesis**: Hate speech, especially when targeted at individuals or small groups has a [Pronoun][Verb] structure throughout the sentence.
   Aim: Create a handcrafted feature that detects the presence of pronouns and counts the occurrence of a verb following a pronoun. Patterns found include "I hate" and "You should".

2. **Hypothesis**: Hate speech is commonly found in a short sentence, with a higher number of uncommon words used.
   Aim: Create handcrafted features which address these assumptions, such as word uniqueness and sentence length.

3. **Hypothesis**: Hate speech contains more offensive words (uni-gram), phrases (bi-gram/tri-gram) per sentences than offensive and neither class.
   Aim: Implement handcrafted feature that counts occurrences of common hate terms from Hatebase.org [1] list of lexicons.

Hypothesis for the performance of models was also set out:

4. **Hypothesis**: The implementation of the handcrafted features to the classification models will improve on the detection rate from the text-based approach, as the features we implement are targeted specifically at improving hate speech prediction.

5. **Hypothesis**: That BERT fine-tuning will yield the best results, this is based on related work by Fohr et al. [1].
   Aim: To achieve results equal to or better than Fohr et al. (Hate Recall: 0.53, Macro Average F1: 0.84).

6. **Hypothesis**: The performance of BERT embeddings fed into classical models should outperform the basic classical models. This is believed due to BERT's special tokenisation and large vocabulary.

5. MODEL FRAMEWORKS

We conduct the research using four supervised learning model strategies. The first two approaches (Section 5.1 and 5.2) are classical approaches: basic text classification approach and hand-crafted feature approach; The two BERT approaches (Section 5.3 and 5.4) are BERT embedding based model and a BERT fine-tuning process.

5.1 Text-Based Approach

The text-based model is the baseline approach used for text classification in the paper, as there were no novel implementations. To gather more information on the effectiveness of a basic text approach, three different models were used, as mentioned in Section 2.3.

We used a gridserachCV method to find the best parameters for each approach. The best performing parameters were hinge loss for LinearSVC and 1.6 C and 3500 iterations for Logistic Regression. The multi-layered perceptron uses the 'adam' solver. For LinearSVC and Logistic Regression approach, a one-versus-rest feature is used to combat the dataset being multi-class. This feature trains a separate classifier on each class and the classifier returns a label based on the highest predicted probability of the three classes [4]. This approach was analysed for multi-class and binary-class classification.

5.2 Handcrafted feature (HCF) Approach

We propose a novel handcrafted approach of 13 chosen features in a feature union. The features provide the classifier with more information to add clarity in order to achieve better results. This approach is novel in the sense that these features have not been used together before. Figure 3 provides an overview of the architecture of the handcrafted feature approach.

We address **Limitation 4** with this approach as the features we design specifically target the detection of hate speech. Each feature was chosen based on the hypothesis set out, and the performance of features is determined individually for evaluation, for example, one feature is left out to assess its effect on the overall performance. To initially deem the features useful to the classification performance, a mean value was computed for select numerical features, see Table 1 in Section 6.3. This approach was implemented using the three models also used in the text-based approach.

The loss parameters were altered for LinearSVC from Squared-hinge to hinge, and for Logistic Regression the penalty and max iteration parameters were "none" penalty and 4500 iterations.

We hypothesised (**Hypothesis 1**) that hate speech is commonly structured containing pronouns (e.g. PRP) followed by a verb (e.g. VRB). We approached this hypothesis by implementing three relevant features:

1. **Part-of-speech (POS)**: POS tagging, which translates all text into their POS type. E.g. "You have" into "PRP VRB". This was stored in a sentence format.

2. **Pronoun Count**: This feature counted all first, second and third-person pronouns found within the text and returned the value as a single count for all three categories.

3. **Pronoun Verb**: This feature used POS and pronoun count to calculate the occurrences of a verb following a pronoun in each tweet. We also include modal verbs, which include "should", "must".

The second set of features were based on the hypothesis (**Hypothesis 2**) that hate speech uses a smaller and less intellectual set of words and smaller sentences.

1. **Word count**: Returns the length of the text input.

2. **Word vs Unique**: Calculates the ratio of unique words to the number of words in text input.

3. **Average Word Length**: The assumption about the intellectual level of words used means an average length for each tweet was calculated. The assumption is made that hate speech does not commonly contain long words.

Following on from the second hypothesis (**Hypothesis 2**) that hate speech uses less intellectual words than non-hate speech we assumpte that the text would be harder to read, with the grammar and spelling more likely being incorrect. We used a python library called TextStat[2] which calculates the readability of text. The packages used in this project were:

1. **Flesch Kincaid Grade**: A score for readability of the text represented as a U.S grade level (school year). It is calculated using a formula which weighs up the number of words per sentence, and syllables per word. In theory, the higher the score, the more intellectual the

[1]https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/lexicons

[2]https://pypi.org/project/textstat/

text is. Therefore, we hypothesised that hate speech would have a lower Kincaid grade score than offensive and neither.

2. **Flesch Reading Ease**: The reading ease score uses a similar formula to the Kincaid grade, but rather represents the grade as an integer in the range 0.00 to 100.00. The higher the number, the easier it is to read. Numbers between 0.00 and 30.00 are classified as being university standard reading.

3. **Number of Difficult Words**: The package Textstat predefined a list of 2949 "easy" words which were used to count the number of difficult, i.e. non-easy, terms in the text.

Furthermore, we investigated text n-grams, up to 3-gram. This was implemented as three different features, where, for example, a text input such as "Hello where are you from" is the uni-gram and "hello where where are are you you from" is the bi-gram.

In line with **Hypothesis 3**, the three features were implemented using a list of 13284 hate uni-grams, bi-grams and tri-grams. Every uni-gram, bi-gram and tri-gram in a tweet is cross-checked with the list. The occurrence of each n-gram lexicon is counted within the text input and stored the count as a feature value.

5.3 BERT Fine-Tuning Approach

This approach uses an already tested solution to the HateOffensive dataset except due to different pre-processing steps; our approach is trained on different parameters. We used a standard implementation called BertForSequenceClassification, from the PyTorch huggingface library, which is produced by adding a neural network sentence classifier as a top layer to the standard BertModel, also a PyTorch implementation [6]. The pre-trained BERT model and the new classification layer are then trained further on our untrained classification task, providing our fine-tune approach with both the large BERT vocabulary and our task-specific data, hence addressing **Limitation 1**.

The parameters were chosen using a brute force approach of testing all of the parameters suggested by BERT developers. The model is optimised using AdamW, which sets the learning rate at $2e^{-5}$. The other parameters were three epochs and a batch size of 16. The longest sequence in the HateOffensive dataset was 57 tokens long, and therefore the max sequence length was defined to be 64.

5.4 BERT Embedding Approaches

The process of extracting the word-piece embeddings uses the PyTorch huggingface implementation of the BertModel on the pre-trained bert-based-uncased English vocabulary and the BertTokeniser. We encode each tweet entry using the tokeniser, adding the special tokens [SEP] and [CLS]. We calculate the max length of the longest sequence and pad each entry to the longest length. Padding is the process of adding null data to the end of the tokenised vector to make all vectors the same length. The tokenised data is fed into the BertModel, where the last hidden layer is extracted to create the embedding.

We feed the embeddings into the LinearSVC, Logistic Regression and Neural Network classifiers. This process is similar to using a classic text vectoriser. However, it uses a more intelligent vectoriser and would be expected to outperform the classical text classification. This approach is also used for binary classification for comparison against the text-based classifier.

6. EXPERIMENTAL SETUP

The setup of the research explains the process of how the data was obtained, pre-processed and structured for training. This section also details the metrics used to assess performance.

6.1 Dataset

The HateOffensive[3] dataset was used to conduct this research. The data was extracted using the Twitter API accompanied by a keyword-based search of Hatespeech.org lexicons, and over 25,000 tweets were initially collected. They predefined the definitions for each class and provided these to the annotators. Crowd-sourcing was used to assign the labels: hate, offensive but not hate or neither offensive nor hate, to each tweet. Each tweet was given at least three annotations, with the majority annotation assigned to the tweet. In the case that a majority annotation was not assigned, the tweet was discarded, therefore leaving 24,783 labelled data entries. The HateOffensive data contains 5.77% hate, 77.43% offensive and 16.80% neither hate nor offensive.

We conducted our research as a multi-class classification task, with all three classes from the HateOffensive. Furthermore, as part of the model evaluation we compute a binary-class classification comparison between the text-based approach and BERT embedding approach. The binary-class classification is achieved by conflating the hate and offensive into one 'hate' class and comparing against the neither class, 'not-hate'.

For training, the data is split randomly into two subsets. This split is 80% training set and 20% test set, meaning the training set contains 19,826 tweets and the test set contains 4,957. Both the training data and the test data are preprocessed.

6.2 Pre-processing

The first step of setting up the experiment is to preprocess the data. Twitter data has no set format except for the restriction of characters and Tweets contain user profile names, hashtags, emojis, mentions and URL links which add unnecessary noise, so we set out to re-structure our data to aid the classification.

To remove unnecessary information from the dataset, the TweetTokeniser function provided by the NLTK library [2] was used. This tokeniser recognises "!!!" as three separate exclamation marks and ":)" as a popular punctuation pattern which represents a smiley face as one entity, instead of ":" and ")".

Punctuation was removed from each text entry, except ".", "!", ",", "?" and "@", as we hypothesised that other punctuation was not important to the discrepancy between hate speech and offensive speech on Twitter, and "@" as it was required to remove the username. We then converted all text from its original form to lowercase and removed all instances of numerical figures.

[3]https://github.com/t-davidson/hate-speech-and-offensive-language/blob/master/data/labeled$_{d}ata.csv$

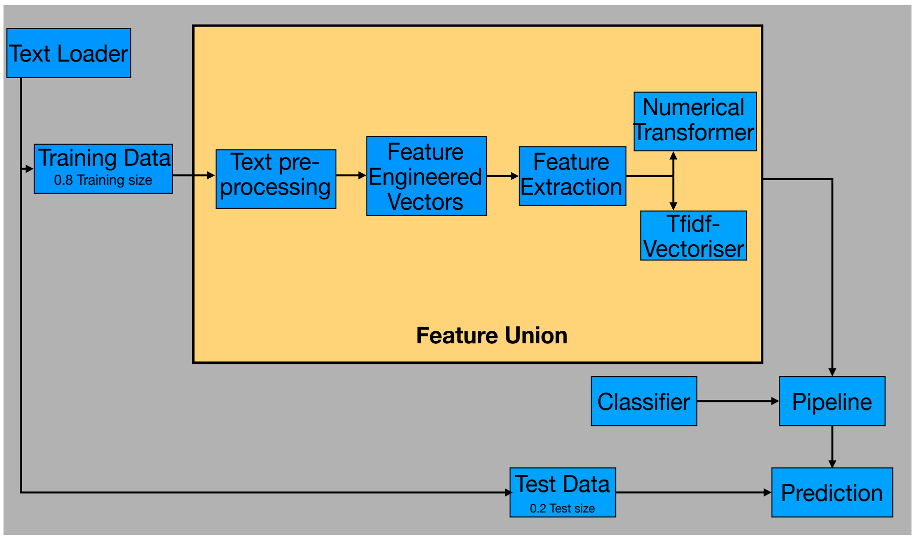

Figure 3: The model architecture of the handcrafted feature approach

We designed a pre-processing function which removes usernames from the tweet, this was signified by tweets that started with “@”. This was significant as 57% of the tweets contained a username, which generally has no meaning to the text. This process encountered one issue, where an offensive tweet was simply “@hoes”. We assumed this was not a username and therefore this tweet was kept as “hoes”. We also removed instances of characters and punctuation that occurred more than three times consecutively such as “yesssss!!!” and returned a stripped version of the same text such as “yes!”. Furthermore, 7131 tweets contain the string “RT” which signifies retweet metadata was removed. This removes any unnecessary noise and reduces confusion to classification models. When investigating the length of the inputs, there were multiple instances of a tweet containing HTML code which represented emojis, and long hashtags. The HTML code was removed, and the hashtags were split into words using the ekphrasis library[4], similar to the approach by Fohr et al. [1]. The last item we stripped from the data was any string that contained “Http” or “www”.

Listing 1: Part of the pre-processing steps involved in Section 6.2

```
!pip install ekphrasis
!pip install textstat

def dataStrip(sentence):
  #uses regex sub function to remove
      username, e.g. @username and retweet
       metadata exception made for one
      instance @hoes which simple removes
      @

  if '@hoes' in sentence:
    sentence = re.sub('@',"",sentence)
  else:
    sentence = re.sub('@[^\s]+','',
        sentence)
    sentence = re.sub('rt',"",sentence)
```

[4]https://github.com/cbaziotis/ekphrasis

```
  #uses regex findall function to search
      for occurrences of string or
      punctuation that occurs more than
      two times in a row

  p = re.findall(r'((\w)\2{2,})', sentence
      )
  if bool(p):
    for i in range(len(p)):
      sentence = sentence.replace(p[i][0],
          p[i][1])
  punc = re.findall(r'(([!?.,])\2+)',
      sentence)
  if bool(punc):
    for i in range(len(punc)):
      sentence = sentence.replace(punc[i
          ][0],punc[i][1])

  #using ekphrasis library all hashtags
      are split into word pieces. e.g.
      Donthurtme --> Dont hurt me

  sentence = text_processor.
      pre_process_doc(sentence)

  #Removes instances of HTML code
  sentence = re.sub('&#[^\s]+;',"",
      sentence)

  return(sentence)
```

To ensure that these procedures optimised the classification, we adopted a “leave-one-out” policy to test for the best combination in initial experiments. As a result of this, the removal of stopwords and porter-stemming, which strips common endings such as “ing”, ‘ly”, ‘ed”, of the raw text were dropped from the pre-processing.

For BERT, slight adjustments were made to the pre-processing steps.

BERT Pre-processing

The process of removing consecutive characters or punctuation was dropped in order to keep the bulk of the text raw, therefore keeping instances of multiple exclamation marks.

We believed that BERT would handle noise better with its word-piece tokenisation. Furthermore, the hashtag split function was also removed as BERT would handle this.

6.3 Feature Engineering

Feature engineering is the process of using the data already obtained, in this case, just the text, to formulate numerical or text-based vectors with the aim of improving the classification performance. The vectors are added into a dataframe alongside the text and label.

Firstly, Table 1 details information about the average values for a selected amount of numerical features. For example, the average word count for hate speech in the dataset is 12.64. Table 1 is split into three sections; word uniqueness and sentence length, Pronoun count and pronoun-verb pattern count, occurrence of recognised hate terms. The mean scores confirms that, on average, hate has a lower word count and smaller sentences, a higher ratio of pronoun to verbs and a higher occurrence of hate terms. It can be seen that the ratio of unique words is equal for all classes and the number of difficult words for hate is lower than the neither class, and that offensive tweets contain, on average, more pronouns per tweet. For pronoun-verb (the instance of a pronoun then a verb in a sentence), hate terms (uni-gram) and word count there is little disparity difference in the mean value between hate and offensive tweets, which potentially causes issues with classification.

Feature	Hate	Offensive	Neither
Average Sentence Length	8.97	9.61	9.18
No. Difficult Words	2.02	1.50	2.22
Ratio of Unique Words	0.96	0.96	0.96
Word Count	12.64	12.80	13.46
Pronoun - Verb	0.41	0.39	0.27
Pronoun Count	0.99	1.07	0.76
Hate terms (Uni-gram)	1.47	1.30	0.28
Hate Terms (Bi-gram)	0.21	0.01	0
Hate Terms (Tri-gram)	0.09	0	0

Table 1: Average value for each class on selected numerical features.

Following this analysis, we decided to remove the ratio of unique words and hate terms (tri-gram) feature columns as they indicated insignificant differences between the classes.

These features are then fed through a transformer which prepares and enables the features to be used in classification. If the feature is text-based, it is fed into a tfidfVectoriser, and if the feature is numerical, it is fed into a standard scaler.

The tfidfVectoriser combines multiple features used to compute text into a numerical array. The tfidfVectoriser determines the importance of a term to the document. To optimise the vectoriser, the gridsearch with 5-fold cross-validation was applied. The optimal parameters found included:

- **N-gram range** = (1,2) : Where the text is vectorised into uni-grams and bi-grams.
- **Stopwords** = "English": Rather than keeping stopwords in the term dictionary, these stopwords are removed.
- **Min_df** = 4: This parameter ignores terms that appear in 4 or less documents - this removes very rare terms
- **Max_df** = 0.85: This ignores terms that appear in over 85% of the documents - this removes common terms outwith the removed stopwords.

6.4 Feature Union & Pipeline

To set up the final part of the experiment, the features need to be combined into a feature union. This is a functionality provided by the sklearn python library. The feature union is used to feed multiple features into a single classification model. This meant that we could conduct a "leave-one-out" policy on the features to find the best combination of features for the classification task. The pipeline is the final step of the experimental setup and creates an environment where the features are fed into a transformer and then a classifier. The pipeline is where the parameters for the text classifiers are declared.

6.5 BERT Model

For both implementations of BERT we used a similar model setup, using the bert-based-uncased model. The bert-based-uncased is a model pre-trained on lower case English vocabulary. It has 12 layers, 768 hidden embedding layers, 12 head layers and 110 million parameters. This model is used as a base model for the BERT fine-tuning process and it forms the word piece embeddings to feed into the classifiers.

6.6 Performance metrics

The performance metrics used in this research were selected as they are suitable for the imbalance of the multi-class dataset; HateOffensive. The metrics used to analyse this paper were recall, precision, macro average F1 score and balanced accuracy. They are represented by values between 0 and 1.

- **Recall**: The ratio between correctly predicted examples from a class X and the total number of examples from a class X.
- **Precision**: The ratio between correctly predicted examples from a class X and the total number of examples that were predicted to be class X.
- **F1**: The F1 score is a weighted harmonic measure of both recall and precision. The formula for F1 is:

$$F1 = \frac{2 * (precision * recall)}{precision + recall} \tag{4}$$

  An F1 score is calculated for each class. Therefore the **macro average** F1 is an average of all classes in the data and the **micro average** F1 is a weighted average over the classes.
- **Balanced Accuracy (BAC)**: It is calculated as the macro average of the recall score for each class. This is a useful metric for an imbalanced dataset, as it ignores the weight of each class.

Limitation 2 is addressed by using metrics which do not favour the higher weighted class, such as the offensive class. To address **Limitation 3** we use four metrics which are commonly used in hate speech classification approaches. We use recall and F1 for the class-specific results, macro and micro F1 and BAC for overall performance results. Previous papers have used F1 without specifying which F1 metric and detail no class specific results.

7. RESULTS

In this section, we detail the results split into subsections based on the hypothesis we set.

7.1 Handcrafted feature impact on hate speech prediction

Table 2 presents the performance metrics for text-based and handcrafted feature approach on the HateOffensive dataset. The best performing model, based on hate F1 score in the HCF approach was the LinearSVC, which outperformed the baseline text-based approach by 21.9%. However, in predicting hate speech (hate recall) the HCF logistic regression model improved by 12% in comparison to the text-based classifier, therefore hypothesis 4 can be deemed valid.

However, whilst the hate recall and hate F1 score improved in all cases for the HCF approach, the overall macro average F1 and micro average F1 decreased for Logistic Regression and for LinearSVC there is only a 1.5% increase.

7.2 Classical model performance

From a test set containing 4957 tweets, the approach shown in Figure 4 correctly predicted 22.5% of the 280 hate tweets compared to the approach in Figure 5 which correctly predicted 35%. We have proven that a basic model can be improved by implementing handcrafted features. However, the figures display that while the performance for detecting hate increased with the features, the performance of predicting offensive class and the neither class decreased by 3.5% and 5.7% respectively. A reason for this could be that the features increased the confusion between the offensive and neither class while increasing the clarity between hate and offensive. Therefore, this may explain why neither of the classical approaches achieves the macro F1 score that we aimed to achieve (Aim 1 - 0.77).

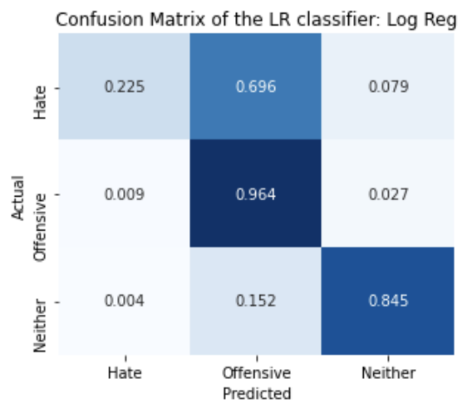

Figure 4: Confusion matrix for text only logistic regression model approach (in %)

The offensive class was the highest predicted class in Figure 5, which is mostly due to the large imbalance in the dataset. In comparing Figure 5 with Figure 2 we see that our model outperforms the classification of offensive tweets by 1.9%, but underachieves in correctly predicting the hate class by 26% and the neither class by 15.3%.

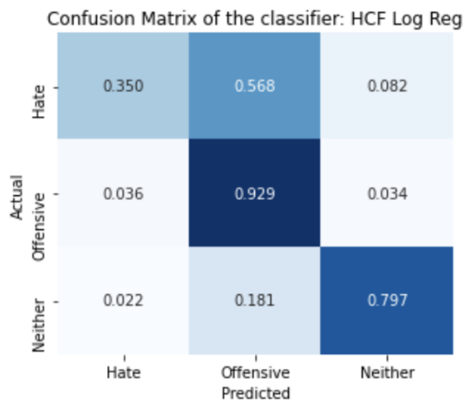

Figure 5: confusion matrix for Handcrafted Logisitic Regression model approach (in %)

7.3 Performance of hypothesised features

For the features engineered to prove the truth of **Hypothesis 1-3** we conducted a leave-one-out policy, as previously mentioned. The hypothesis were split into the three categories: **Hypothesis 1** - Part-of-speech, **Hypothesis 2** - Hate speech structure and **Hypothesis 3** - n-gram hate terms. We found that adding n-gram hate terms and hate speech structure improved the prediction of hate recall and hate F1 by 10% and 6.4% respectively. On the other hand, we found that part-of-speech features negatively impacted the prediction of the hate class by 4%. The results show that the pronoun count feature was the cause of the impact. We can therefore deduce that **hypothesis 2** and **3** aided hate speech classification and accept these hypothesis, however we then deny **hypothesis 1**.

7.4 BERT Fine-tuning

Table 3 shows the performance metrics for our optimised BERT fine-tune approach. It shows that for all metrics, the fine-tuning approach performs better than the other three approaches. Figure 6 shows that 55% of tweets labelled hate have been incorrectly classified as offensive; in comparison, this outperforms the HCF logistic regression model in predicting hate correctly by 5%.

Figure 6 highlights a similar issue found in Figure 4 and 5 such that most of the misclassification occurs between hate and offensive tweets. In Figure 6 the correct prediction of the offensive class is 4% better than [4]. The performance of BERT fine-tuning means we can accept **Hypothesis 5** as a valid hypothesis.

7.5 BERT embeddings

Table 4 provides three metrics for comparison between multi-class and binary-class classification; F1 score for hate class, balanced accuracy curve and macro average F1. The comparison marks the performance difference between using BERT embeddings as input features for LinearSVC, Logistic regression and neural networks, and a classical tfidfVectoriser input feature into the same three models.

We can observe that for all instances, the BERT embeddings approach was out-performed by the classical approach. Interestingly, the percentage difference between the performance of LinearSVC and the LinearSVC + BERT embedding was 93.9% for the multi-class prediction. This suggests

	Classification Model	Hate Recall	Hate F1	Macro Avg. F1	Micro Avg. F1	BAC
Text Based	LinearSVC	0.21	0.32	0.72	0.89	0.69
	Logistic Regression	0.23	0.35	0.72	**0.9**	0.69
	Multi-layered perceptron	0.25	0.27	0.66	0.87	0.65
HCF Approach	LinearSVC	0.31	**0.41**	**0.73**	**0.9**	**0.71**
	Logisitic Regression	**0.35**	0.37	0.7	0.89	0.69
	Multi-layered perceptron	0.27	0.36	0.7	0.88	0.67

Table 2: Results to show performance of text based approach (baseline) and hand-crafted feature approach

Parameters	Hate Recall	Hate F1	Macro Average F1	Micro Average F1	BAC
Learning Rate = $3e^{-5}$ Epochs = 3 Batch Size = 16 Max Length = 64	0.40	0.45	0.76	0.91	0.75

Table 3: BERT Fine-tuning approach performance metrics

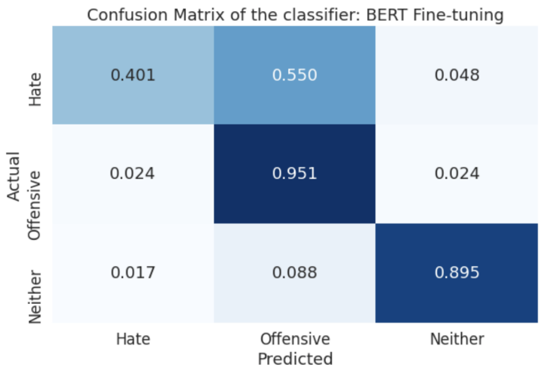

Figure 6: confusion matrix for BERT fine-tuning approach (in %)

an issue with the BERT embeddings being fed into our best performing classical model. Fohr et al. [1] propose that the embeddings approach performed worse as it is pre-trained on Wikipedia and BookCorpus data, therefore unlikely to have been trained on any hate speech vocabulary or any Twitter data.

On the other hand, the classical models are trained on the HateOffensive dataset and therefore better distinguish between hate and offensive text. Furthermore, for both approaches, the binary class classification achieves much higher scores, proving the difficulties of detecting the difference between hate and offensive.

We can, therefore, deny the hypothesis that BERT embeddings fed into a classical model will outperform the classic text approach.

7.6 Analysis

As shown in Figure 5, 56.8% of hate tweets in the test set were labelled as offensive, so further understanding of the misclassification was required. Firstly, all 159 tweets pre-labelled as hate that were predicted to be offensive were extracted. The tweets were analysed using the lime library [23], which uses the predicted probability of a sentence to explain the prediction choice on a word by word basis. A specific sentence which stood out for classification issues was "@Oskzilla fuck you niggah you gay. Lol".

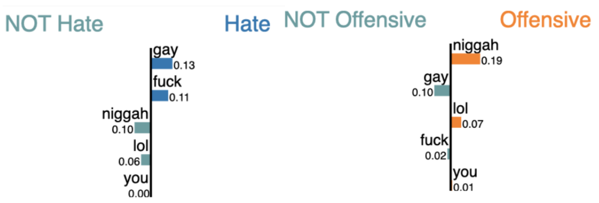

Figure 7: Word by word prediction of sentence.

Figure 7 details the predicted probability of each word, in this case "niggah" has been predicted to be an offensive word, therefore explaining the misclassification. We correct the spelling by removing "ah" and placing "er" to test the prediction.
Figure 8 details how the spell correction alters the predictability performance to correct the prediction. We replicated this analysis for the BERT fine-tuning, which predicted the sentence correctly, as expected due to BERT's word-piece tokenisation.

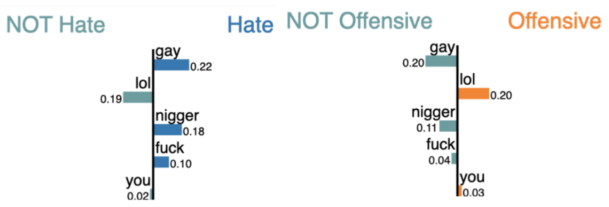

Figure 8: Word by word prediction of a spell corrected sentence.

Our BERT fine-tuning model achieves a macro average F1 score of 0.76, very similar to the performance of MacAveney et al.'s multi-view SVM which achieved 0.77, and a micro average F1 score of 0.91 that outperforms the score of 0.9 achieved by Davidson et al.. However, our model achieves a hate class recall score of 0.4 compared to the 0.61 achieved by Davidson et al.. MacAveney et al. use a hate class pre-

	Multi-Class			Binary		
Classification Approach	Hate F1	BAC	F1 Score	Hate F1	BAC	F1 Score
LinearSVC + BERT Embedding	0.02	0.54	0.54	0.94	0.8	0.81
LinearSVC	0.32	**0.69**	0.72	0.97	**0.92**	**0.92**
% Difference	93.94%	21.74%	25.00%	3.09%	13.04%	11.96%
Logistic Regression + BERT Embedding	0.21	0.58	0.61	0.94	0.8	0.82
Logistic Regression	**0.35**	**0.69**	**0.72**	**0.98**	0.88	0.90
% Difference	43.24%	15.94%	16.44%	4.08%	9.09%	8.89%
Neural Network + BERT Embedding	0.20	0.61	0.60	0.94	0.81	0.82
Neural Network	0.27	0.65	0.66	0.96	0.88	0.89
% Difference	25.93%	6.15%	9.09%	2.08%	7.95%	7.87%

Table 4: Comparison of performance of multi-class models vs binary class models.

cision score and Founta et al. and Zhang et al. reported micro average F1 scores of 0.89 and 0.91 respectively but no individual class results. This demonstrates the problem highlighted by **Limitation 3** which references the lack of standardised performance metrics. Moreover, this lends to **Limitation 5**, whereby the lack of standardisation means information is 'hidden' which causes problems with comparing against previous work.

Finally, even though our BERT fine-tuning approach outperforms Davidson et al. for the micro average F1 score, we do not improve on detecting hate. Therefore we argue that for the HateOffensive dataset, and other imbalanced hate speech datasets, the micro average F1 score has little meaning, and that instead individual class performance should be more transparent.

7.7 Discussion

To improve the overall macro average F1 score in the future, we analysed where misclassification has occurred in extreme cases, such as hate tweets labelled as neither and vice versa. First, we analysed tweets pre-labelled as neither using the lime algorithm, two examples are shown:

- "finally! warner bros. making superhero films starring a woman, person of color and actor who identifies as 'queer'" - "queer" and "color" are determined as hate by the lime algorithm. However, this tweet is celebrating that people of color, women and, non-heterosexual actors are being employed. The prediction probability for this tweet was 0.37 to hate, 0.3 to offensive and 0.33 to neither, indicating a difficult classification task.
- "i hate stereotypes. preconceived notions that black men are athletes or criminals.. black women are ghetto and confined to ..." - "hate" and "black" are the keywords in this sentence causing misclassification. The tweet refers to the distaste of racial stereotypes. This tweet had a predicted probability of 0.44 for hate, 0.33 for offensive and 0.23 for neither.

These tweets show how key words can have an important impact on text classification. Two examples of tweets prelabelled as hate, which our model classified as neither:

- "washing my coon hair" - with context this sentence would be easier to classify. The "my" insinuates the hair belongs to the person who is of African origin. Lime correctly labels "coon", a derogatory term, as hate however in the context we believe "coon" is being used in a non-abusive form.
- "told my dad to go buy cookies for the graduation reception ... this nigga bought oreos". This sentence has a 0.63 prediction probability of being classified as neither. The inference in the sentence suggests "nigga" is used as a racial slur; however, the reading of the sentence suggests otherwise.

These two examples highlight a potential incorrect labelling throughout the dataset and also that humans find it a difficult task to classify tweets.

The sentence "beans u beaner" was also incorrectly predicted as neither hate nor offensive. The word "beaner" is a derogatory term for a Hispanic. The word exists in 34 tweets in the dataset, with 17 of these tweets labelled to the neither class, therefore likely decreasing the hatefulness of the word to the classification model.

The complexity and dynamic style of Twitter language causes issues with classic approaches to text classification, i.e. new words may be used as abusive terms to describe groups of people which therefore are initially very difficult to detect.

8. CONCLUSION & FUTURE WORK

In this paper, we have addressed the challenges in automatic hate speech detection. We proposed four different approaches to detecting hate speech. Two classical methods which were the text-based approach and the handcrafted feature approach, and two BERT based approaches; Word embeddings and fine-tuning. The main work was designing the novel handcrafted feature approach and building our BERT fine-tuning model.

The features were explicitly designed based on the hypothesised syntactic structure of hate speech. The features included hate uni-gram and bi-gram occurrence counters, the occurrences of pronouns and verb pairs in the tweet and number of difficult words. The features improved the detection of hate by 12% for the hate recall; however, the overall performance of the approach was negatively impacted by the addition of the features. In future work, we would like to investigate features that would focus on detection offensive language as we would aim to improve the overall performance of the classification.

In the BERT word embedding approach, the embeddings are fed into a LinearSVC, Logistic Regression and neural network so that a comparison could be made against the text-based approach using the same models. These two approaches were also converted into a binary class classification task, where the hate and offensive classes were merged and compared against the neither class. This comparison concluded that classical text-based approach out-performed the BERT word embeddings.

We observed that the BERT fine-tuning approach was the best performing approach. However, work is still to be done in increasing the prediction success of the hate class for all approaches, and future work would involve implementing more features and improving BERT fine-tuning optimisation.

The field of natural language processing is ever-changing, and therefore 100% classification is almost impossible on real-world data. The pre-processing steps involved removing the HTML code for emojis, however, emojis can provide contextual information to a tweet. For example, "&#58380" represents a face wearing a mask, it is defined to mean "This emoji is sick and contagious; a very courteous emoji by not spreading disease". It can also be used to represent someone's dislike, and therefore the use of this emoji in combination with a group of people or an individuals name in a tweet could signify hate. Future work would seek to translate the HTML code into word meanings for the emojis. Additionally, as shown in Figure 7 and 8, the misspelling of common hate terms has an impact on the classification. Future work would investigate and compile a list of common misspellings in the dataset to be corrected. This would then be representative of real-world misspellings, meaning our approach could be applied to a variety of social media classification tasks.

Acknowledgments. I would like to give thanks to Craig Macdonald for his continued support and advice as my supervisor throughout the year.

9. REFERENCES

[1] D. F. Ashwin Geet d'Sa, Irina Illina. Bert and fasttext embeddings for automatic detection of toxic speech. 2020.

[2] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python.* O'Reilly Media, 2009.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

[5] O. de Gibert, N. Perez, A. García-Pablos, and M. Cuadros. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[7] A. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis. A unified deep learning architecture for abuse detection. *CoRR*, abs/1802.00385, 2018.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[9] D. Hicks and D. Gasca. A healthier twitter: Progress and more to do, 2019.

[10] S. Hinduja. Federal commission on school safety - cyberbullying research center, 2019.

[11] S. Hinduja and J. Patchin. Connecting adolescent suicide to the severity of bullying and cyberbullying. *Journal of School Violence*, pages 1–14, 08 2018.

[12] Q. Huang, V. K. Singh, and P. K. Atrey. Cyber bullying detection using social and textual analysis. In *Proceedings of the 3rd International Workshop on Socially-Aware Multimedia*, SAM '14, page 3–6, New York, NY, USA, 2014. Association for Computing Machinery.

[13] T. Inc. Hateful conduct policy, 2020.

[14] A. "John, A. C. Glendenning, A. Marchant, P. Montgomery, A. Stewart, S. Wood, K. Lloyd, and K. Hawton. "self-harm, suicidal behaviours, and cyberbullying in children and young people: Systematic review". *"J Med Internet Res"*, "2018".

[15] S. Kemp. *Digital around the world in 2019.* Hootsuite, WeAreSocial, Kepios, 2019.

[16] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[17] H. Lee, H. Lee, J. Park, and Y. Han. An abusive text detection system based on enhanced abusive and non-abusive word lists. *Decision Support Systems*, 113:22–31, 9 2018.

[18] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(2):4–22, Apr 1987.

[19] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder. Hate speech detection: Challenges and solutions. *PLOS ONE*, 14(8):e0221152, 2019.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] S. Petrov, D. Das, and R. T. McDonald. A universal part-of-speech tagset. *CoRR*, abs/1104.2086, 2011.

[22] S. Poria, E. Cambria, D. Hazarika, and P. Vij. A deeper look into sarcastic tweets using deep convolutional neural networks. *CoRR*, abs/1610.08815, 2016.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[24] T. safety team. Twitter: Enforcing our rules, 2019.

[25] L. A. Silva, M. Mondal, D. Correa, F. Benevenuto,

and I. Weber. Analyzing the targets of hate in online social media. *CoRR*, abs/1603.07709, 2016.

[26] F. C. S. Team, 2020.

[27] Z. 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*, pages 138–142, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[28] World interent statistics, 2020.

[29] W. Wu, X. Gao, and S. Gao. One-versus-the-rest(ovr) algorithm: An extension of common spatial patterns(csp) algorithm to multi-class case. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 3:2387–90, 02 2005.