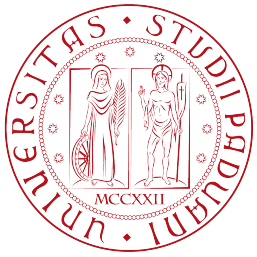


UNIVERSITÁ DEGLI STUDI DI PADOVA

DEPARTMENT OF MATHEMATICS "TULLIO LEVI-CIVITA"
Master of Science in COMPUTER SCIENCE

Hate Speech Detection: An Analysis of Existing Architectures

Candidate:
Luca PAJOLA

Supervisor:
Professor Mauro CONTI
Universitá di Padova

Co-Supervisor:
Professor N. Asokan
Aalto University

External-Reviewer:
Professor Roberto Navigli
Universitá La Sapienza

ACADEMIC YEAR 2017-2018

"Rather than love, than money, than fame, give me truth."

Henry David Thoreau

"So many people live within unhappy circumstances and yet will not take the initiative to change their situation because they are conditioned to a life of security, conformity, and conservatism, all of which may appear to give one peace of mind, but in reality nothing is more dangerous to the adventurous spirit within a man than a secure future. The very basic core of a man's living spirit is his passion for adventure. The joy of life comes from our encounters with new experiences, and hence there is no greater joy than to have an endlessly changing horizon, for each day to have a new and different sun."

Christopher McCandless

Abstract

With the spread of social networks and their unfortunate use for hate speech, automatic detection of the latter has become a pressing problem. In this paper, we reproduce seven state-of-the-art hate speech detection models from prior work, and show that they perform well only when tested on the same type of data they were trained on. Based on these results, we argue that for successful hate speech detection, model architecture is less important than the type of data and labelling criteria. We further show that all proposed detection techniques are brittle against adversaries who can (automatically) insert typos, change word boundaries or add innocuous words to the original hate speech. A combination of these methods is also effective against Google Perspective – a cutting edge solution from industry. Our experiments demonstrate that adversarial training does not completely mitigate the attacks, and using character-level features makes the models systematically more attack-resistant than using word-level features.
The contribution of this thesis is also resulted in a scientific paper:

- Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, N. Asokan. All you need is "love": Evading hate speech detection. In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (ACM CCS 2018 workshop: AISec 2018), in press, Toronto, Canada, October 19, 2018.

Acknowledgements

If I look backward in the last few years, and if I think about it, I see that I need to say thanks to a lot of people. First of all, a really and sincerely gratitude to my supervisor, Prof. Mauro Conti, that gave me these immense opportunity with amazing people for this thesis and he allows me to learn like never before. Thanks to prof. Asokan, which gave me the chance to work in its research group in Aalto University, where I worked with with Tommy Gröndahl and Mika Juuti and they taught and followed me carefully in these months.
Thanks to my family, Otello and Simonetta, my parents, and Nicola and Valentina, which supported and incited me in every decision that I made. Thanks to all of my friends that in these years stays with me in every thing, from the old one with whom I grow up (Chiara, Ilaria, Johny, Laura, Luca, Mattia B., Mattia C., Marco , Stefano, Veronica), to the new that I made during the University and my life in Helsinki (Ambra, Alexandra, Davide, Dario, Dennis, Federica, Filippo, Guglielmo, Michelle, Valentina, Veronica).
Thanks to Daniele, this thesis is dedicated to you...

Contents

Abstract iii

1 Introduction 1

2 Background 3
2.1 Text Classification Pipeline . . . 3
2.1.1 The Pipeline . . . 3
2.2 Models Description . . . 5
2.2.1 Logistic Regression . . . 5
2.2.2 MultiLayer Perceptron . . . 5
2.2.3 CNN . . . 6
2.2.4 Recurrent Neural Networks . . . 8
2.2.5 Language Models . . . 9
2.2.6 Loss and Optimizers . . . 9
2.3 Transfer Learning . . . 11
2.4 Evaluation Metrics . . . 12
2.4.1 Accuracy . . . 12
2.4.2 Precision, Recall and F-measure . . . 12
The binary case . . . 13
F1 score . . . 13
2.4.3 AUC . . . 14
2.5 Machine Learning Evasion . . . 14
2.5.1 Evasion at test time . . . 15
2.5.2 Attacks scenarios . . . 15
2.5.3 Adversarial Machine Learning . . . 16

3 Hate Speech and Related Works 17
3.1 The European Court of Human Rights . . . 17
3.2 Hate Speech in the NLP Community . . . 18
3.3 Offensive vs. Hate Speech . . . 19
3.4 The Effect of the Hate Speech . . . 19
3.4.1 Cyberbulling . . . 20
3.4.2 Sexism . . . 22
3.5 Hate Speech Representation . . . 22
3.6 Spam Detection and Evasion . . . 23

4 Dataset 25
4.1 Datasets Introduction . . . 25
4.1.1 Wikipedia (W) . . . 25
4.1.2 Twitter 1 (T1) . . . 26
4.1.3 Twitter 2 (T2) . . . 26
4.1.4 Twitter 3 (T3) . . . 27
4.2 Datasets Analysis . . . 27

4.2.1 Classes Distribution . . . 27
4.2.2 Sentences Examples . . . 27
4.2.3 Datasets Similarities . . . 30
4.2.4 Swear words . . . 33

5 Models & Performances **35**
5.1 Models Description . . . 35
5.1.1 Logistic Regression and Multilayer Perceptron over W dataset . 36
5.1.2 Logistic Regression over *T1* dataset . . . 37
5.1.3 LSTM over *T2* dataset . . . 38
5.1.4 CNN + GRU over *T1**, *T2* and *T3* datasets . . . 40
5.1.5 Transfer Learning . . . 40
5.2 Evaluations . . . 42
5.2.1 Evaluation Metric . . . 42
5.2.2 Replication and Re-Training . . . 43
5.2.3 Cross Application Between Datasets . . . 44
5.3 Swear Words Analysis . . . 44
5.4 Analysis of the Behaviour . . . 45
LR char over *W* . . . 46
5.4.1 MLP char over *W* . . . 47
5.4.2 LR word over *T1* . . . 48
5.4.3 LSTM RE *T2* . . . 49
5.4.4 CNN GRU *T1'* . . . 49
5.5 Augmentation . . . 50

6 Attacks **53**
6.1 Adversary models . . . 54
6.1.1 Word Changes . . . 54
Inserting Typos . . . 54
Leetspeak . . . 56
6.1.2 Word Boundary Changes . . . 56
6.1.3 Word Appending . . . 56
6.2 Mitigation . . . 57
6.3 Results . . . 58
6.4 Love Attack . . . 64
6.5 Trade-off: Performance vs. Security . . . 64

7 Considerations and Conclusions **67**

A Published Paper **69**

B Media Coverage **81**

Bibliography **83**

List of Figures

2.1 Document Classification Pipeline . . . 3
2.2 Feed-forward neural network . . . 6
2.3 Convolutional Neutal Network in a 2D space . . . 7
2.4 The dynamic of a RNN: each node represent a state at some time t , and the function f map the state t to the state $t+1$ . . . 8
2.5 Differences between LSTM and GRU . . . 9
2.6 Gradient descent. Illustration of how the gradient uses the derivatives of a function for finding the minimum (Ian Goodfellow and Courville, 2016) . . . 10
2.7 A comparison between the traditional machine learning and transfer learning . . . 11
2.8 AUC example . . . 14

4.1 Classes distribution for each dataset . . . 28
4.2 Similarity between the dictionaries; each dictionary in the rows represent *d1*, the dictionary that we want to cover, while the columns represent *d2*, the dictionaries used to cover . . . 31
4.3 Similarity between the words distributions; each dictionary in the rows represent *d1*, the dictionary that we want to cover, while the columns represent *d2*, the dictionaries used to cover . . . 32
4.4 Word ranking per dataset . . . 32

5.1 Confusion Matrices . . . 38
5.2 Architecture of CNN GRU . . . 41
5.3 ULMFiT structure . . . 42
5.4 Confusion Matrices of the models . . . 52

6.1 Typos Attack and Adversarial Training . . . 61
6.2 Leet Attack and Adversarial Training . . . 61
6.3 Insert Whitespaces Attack and Adversarial Training . . . 62
6.4 Remove White spaces Attack and Adversarial Training . . . 62
6.5 Insertion of Common Words Attack and Adversarial Training . . . 63
6.6 Insertion of Non Hateful Words Attack and Adversarial Training . . . 63
6.7 Trade off performance - security: a comparison of the performance over different depths of the Decison Trees in Adaboost classifier . . . 65

B.1 Wired - 26.9.18 . . . 81
B.2 Repubblica - 12.9.18 . . . 82

List of Tables

2.1 Confusion Matrix . . . 13

3.1 Comparison of the percentage of teenager which were bullied / cyberbullied in 2008. *Often*: from several times a week to few times per month, *occasionally*: happened just few times, *never*: never happened (Smith et al., 2008) . . . 20
3.2 *Usage*: percentage of the people which often or frequently use the technology; *Victims*: percentage of people which were victimized (Whittaker and Kowalski, 2014) . . . 20
3.3 Cyberbullying victimization over the last 30 days (Hinduja and Patchin, 2010) . . . 21

4.1 Datasets used in our replication. Union (∪) denotes the conflation of class elements. . . . 27
4.2 Top 10 words for each offensive / hateful class in each dataset . . . 30

5.1 Replicated machine learning models . . . 35
5.2 Original models and performances over W . . . 36
5.3 Original Classification Report . . . 38
5.4 Replicated Classification Report . . . 38
5.5 Deep Learning techniques applied. **RE** = Random Embedding . . . 39
5.6 Original evaluations with F1 score micro over the three dataset . . . 40
5.7 Median Classifier Evaluation . . . 43
5.8 F1-scores (macro-averaged across classes) of the two-class models trained and evaluated on each dataset (datasets used in original papers in bold) . . . 43
5.9 F1-scores (macro-averaged) of pre-trained two-class models applied to different test sets, with the original test set in parentheses . . . 44
5.10 Performance of two-class models on offensive but non-hateful speech . 45
5.11 Google Perspective "toxicity" scores on non-hateful sentences with and without a curse word . . . 45

6.1 **Word Changes**: F1-scores on the "hate" class from attacks and mitigation.**A**: Attacks, **AT**: Adversarial Training, **SC**: Spell Checker . . . 59
6.2 **Boundary Changes**: F1-scores on the ""hate class from attacks and mitigation.**A**: Attacks, **AT**: Adversarial Training, **RW**: Remove whitespace . . . 60
6.3 **Word Appending**: F1-scores on the "hate" class from attacks and mitigation.**A**: Attacks, **AT**: Adversarial Training . . . 61
6.4 **Love Attack**: F1-scores on the "hate" class from attack. . . . 64
6.5 **Perspective Attack**: level of toxicity (0.5 the threshold) . . . 64

Chapter 1

Introduction

Social networking has changed the way people communicate on-line. While the ability of ordinary people to reach thousands of others instantaneously undoubtedly has positive effects, downsides like polarization via echo chambers (Hosseini et al., 2017) have become apparent. This inter-connectedness of people allows malicious entities to influence opinions by posting hateful material, also known as hate speech.

Hate speech is not a universal concept. While laws targeting speech seen as harmful have existed throughout human civilization, the specific term was originally coined in the US in 1989 to address problems of "harmful racist speech" that was nonetheless protected in the US (Brennan, Afroz, and Greenstadt, 2011). In 1997, the European Union defined "hate speech" as texts that "spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance".Hate speech can be separated from merely offensive or shocking content "Burnap:Williams2015", although this distinction is non-trivial. In this paper, we denote non-hateful speech as "ordinary speech".

Typically, hate speech detection is cast as a classification problem. Standard machine learning algorithms are used to derive a discriminative function that can separate hate speech from ordinary speech.

Although several hate speech detection mechanisms have been reported in the research literature (Schmidt and Wiegand, 2017), to the best of our knowledge, there has so far been no systematic empirical evaluation comparing actual implementations of proposed models and datasets.

We study five recent model architectures presented in four papers. One architecture (Zhang Ziqi, 2018) is trained separately on three different datasets, giving us seven models in total. Six of these distinguish between two classes (Wulczyn, 2017), (Zeerak and Hovy, 2016) (Zhang Ziqi, 2018). One classifies among three, distinguishing between offensive and non-offensive ordinary speech (Davidson et al., 2017). Two models are character-based (using character n-grams as features) while the rest are word-based (using word n-grams or embeddings).

In the original papers, each of the two-class models was evaluated using a particular dataset. We show that none of the pre-trained models perform well when tested with any other dataset. This suggests that the features indicative of hate speech are not consistent across different datasets. However, we also show that all models perform equally well if they were retrained with the training set from another dataset and tested using the test set from the same dataset. This suggests that hate speech detection is largely independent of model architecture. We also tested each two-class model on offensive ordinary speech (Jay, 2009), and observe that they tend to classify it as hate speech. This indicates that the models fail to distinguish between hate speech and offensive ordinary speech, making them susceptible to false positives. We experimented with using a transfer learning approach where by using a pre-trained language model, we fine tuned it for the classification task with each

dataset, and conducted the same experiments. We show that the results are comparable but do not exceed the baselines.

Prior work has only considered what can be called naive adversaries , who do not attempt to circumvent detection. We show that all the models are vulnerable to adversarial inputs. There are many ways of attacking text-based detection models. A simple attack involves changing the input text so that a human reader will still get the intended meaning, while detection models mis-classify the text.

We suggest three such alteration techniques, all of which are easily automated:

- word changes;
- word-boundary changes;
- appending unrelated innocuous words.

Implementing two varieties of each attack, we show that all detection models are vulnerable to them, although to different extents. Combining two of our most effective attacks, we present a simple but powerful evasion method, which completely breaks all word-based models, and severely hinders the performance of character-based models. In addition, this attack significantly degrades the performance of Google Perspective API, which assigns a "toxicity" score to input text.

In conclusion, the idea of this research is to find the properties of this task (hate speech detection) with the aim to help future research; we summarize our contributions as follows:

- The first experimental comparative analysis of state-of-the-art hate speech detection models and datasets 5;
- Several attacks against effective against all models and possible mitigations 6;
- A simple but effective evasion method that completely breaks all word-based classifiers, and significantly impacts character-based classifiers as well as Google Perspective 6.4;
- A review of the limitations in current methods, and desiderata for future developments 7.

Chapter 2

Background

This chapter contains all the theoretical knowledge required for understanding the analysis and the concepts which will be describe in the following chapters. We will begin by describing how text classification tasks are handled in Section 2.1, followed by the description of the models involved in Section 2.2 and Section 2.3. Models are evaluated with different metrics, as explained in Section 2.4. In the end, we conclude an explanation of the basic concept in the machine learning evasion (Section 2.5).

2.1 Text Classification Pipeline

In this section I briefly explain how to the Text Classification pipeline is handle, based on the work of Korde (Korde, 2012). Text classifiers are important because we need a way for extracting information from the text, which contains more than the 80% of the information. Usually, the aim of a text classifier is to classify documents according to predefined categories.

2.1.1 The Pipeline

We can summarize the pipeline as in Figure 2.1.

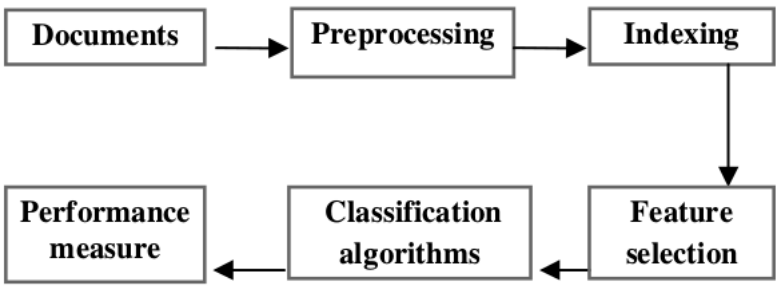


FIGURE 2.1: Document Classification Pipeline

Documents The first step is the samples collection. In this phase we need to collect documents, and they can have different origins, such as text file, pdf, html, ...

Pre-Processing In this stage we try to represent the documents in a good manner, trying to remove all the useless information. Here, common steps are:

- *Tokenization*: the role of a tokenizer is to transform a sentence from a string to a list of tokens (e.g. "I love going to the beach" become ['I', 'love', 'going', 'to', 'the', 'beach']);

- *Removing stop words*: stop words are terms which occur frequently (e.g. "the", "a", "and"); these words become insignificant and they can be removed;
- *Stemming words*: the stemming aim is to convert the words into the canonical form. (e.g. "books" to "book", "loving" to "love"); usually it is applied to the tokens.

Indexing Documents are represented by words; in order to reduce the complexity of the data representation, each document need to be transformed into a document vector. Usually documents are transformed into vectors of words. A basic representation is the following:

$$\begin{bmatrix} & T_1 & .. & T_t & c_i \\ D_1 & w_{11} & .. & w_{t1} & c_1 \\ .. & .. & .. & .. & .. \\ D_n & w_{1n} & .. & w_{tn} & c_n \end{bmatrix}.$$

Where T_i represents the *i-th* term (word) in our dictionary, which is made by the words that we decide to uses for representing a corpus. D_j represent the *j-th* document and c_j the label of D_j. About w_{mn}, it is just a weight that represent the *m-th* terms in the *n-th* document. This representation can be done in different ways, such as boolean (1 if the word is present in the document, 0 if not), or the frequency of the word, or the TF-IDF, or entropy etc... This representation clearly is a sparse matrix, and it raise the problem of the high dimensionality; moreover, with these approach we loose information about the correlation between words: we know if the word is presented or not (with a weight for the representation), but we do not know its position(s) in the document, or about its neighbors. Another way for representing the words and maintaining the relationship is the *n-grams* representation, where there is a representation as a sequence of words or characters. For example, in a word n-gram of size from 1 to 2, if we have the sentence "*I am Luca*" we will have the following n-grams: [*I*, *am*, *Luca*, *I am*, *am Luca*].

Feature Selection The main aim of the feature selection is to select a subset of features from the original representation vectors. The algorithm works by preserving the features which have a high score, according to a pre-defined metrics. In this way we reduce the feature space, which has an high dimensionality. This help the scalability, efficiency and accuracy of a text classifier. There are different techniques for choosing features, such as the text frequency (e.g. maintain all those word which frequency is greater than X), chi-square.

Classification Classification algorithm can be summarized in three main categories: supervise, unsupervised and semi-supervised learning. There are a lot of techniques that can help in the classification problem, such as Decision Trees, K-nearest Neighbor (KNN), Support Vector Machines (SVM), Neural Networks.

Performance Evaluation Metrics are useful techniques for evaluating algorithms or, like in this field, models, and they allow to understand the performances of a classifier and the comparison with others. Tons of metrics are available in the literature, and each one try to catch the performance of the model based on a specific aspect. This mean that the choice of the metric will influence how you weight the importance of different characteristics in the results.

2.2 Models Description

2.2.1 Logistic Regression

In this subsection I will explain the Logistic Regression, based on "Logistic Regression: A Self-Learning text" (Richardson, 2011).

Logistic regression models the probability that an outcome belongs to a particular category, where all the probabilities are in the range [0, 1]. The logistic regression is defined as following:

$$Y = \frac{e^{\beta_0+\beta_1 X_1+..\beta_n X_n}}{1+e^{\beta_0+\beta_1 X_1+..\beta_n X_n}}.$$

Where the component $\beta_0 + \beta_1 X_1 + ..\beta_n X_n$ is also known as ***logit***. Because of its nature, the logistic regression become popular due to the fact that it can perfectly describe the probability of an event. For example, given a vector X of values which describe a patient, Y will be the probability of the risk that that patient will getting a disease.

2.2.2 MultiLayer Perceptron

This subsection is inspired by "Deep Learning Book" (Ian Goodfellow and Courville, 2016).

The MLP (also known as feed-forward network) aim is to approximate some function f^*. Let's consider the following example:

$$y = f^*(\mathbf{x}),$$

where this is a mapping between $\boldsymbol{x}$ and y. The MLP define a mapping $y = f(\mathbf{x}, \theta)$ and learns the values of the parameter Θ that result in the best function approximation.

MLPs are also called ***networks*** because they are typically composed by different functions. For example we could have three function connected in a chain, f_1, f_2, f_3 and together they form

$$f(\mathbf{x}) = f_1(f_2(f_3(\mathbf{x}))).$$

In this case, f_1 is called *first layer* of the network, f_2 *second layer* of the network and so on. The length of this chain is called ***depth*** of the network. The final layer is called *output layer*. Each layer of the network is a vector of values; the dimensionality of these layers determinate the ***width*** of the network. We can also think of the layer as consisting of many ***units*** that act in parallel, each representing a vector-to-scalar function. Each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value. For understanding how feed-forward networks work, we can think about how linear model works. Linear models, such as linear regression and logistic regression, are widely use because of their efficiency in the fitting process. One limitation of the linear models is that their model capacity is limited to linear function and they cannot understand the interaction between any pair of input variables. We can extend the linear model to represent non-linear functions of $\boldsymbol{x}$ and then apply the linear model to the transformed input $\boldsymbol{x}$ given by $\phi(\mathbf{x})$, where ϕ is the non linear transformation. Now we need to think how to choose ϕ: in deep learning this parameter is learned; in this approach the model is defined as following:

$$y = f(\mathbf{x}; \theta, \mathbf{w}) = \Phi(\mathbf{x}; \theta)^T \mathbf{w}.$$

Here, we use an *optimization algorithm* to find θ that correspond to a good representation.

When we design a feed-forward neural network we need to:

- choose an optimizer;
- define a cost function;
- decide the shape of the units' output.

Each layer requires an ***activation function*** that it is used for computing the output of the layer.

The learning of these models are done in two steps:

1. forward: compute the output of the model with the current weights;
2. back-propagation: calculate the error given by the output and propagate the error in the network.

The ***back-propagation*** algorithm computes efficiently the gradient for minimizing the error that the model is doing.

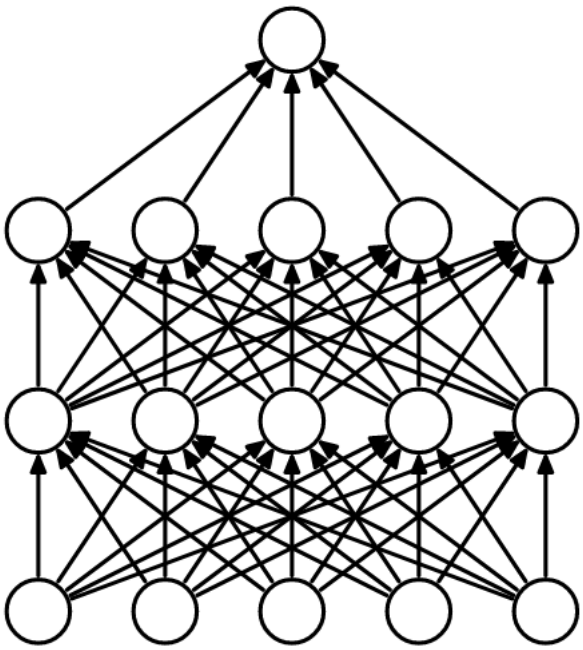


FIGURE 2.2: Feed-forward neural network

In Figure 2.2 (Srivastava et al., 2014) we see a full connected neural network, where each unit of the layer l_i is connected to all the units in the layer l_{i+1}. From the bottom, we can see the input layer, after two hidden layers and in the end the output layer. Usually, the output layer contains one units per class (in the classification tasks) or one units per outcome (regression task). The width of the layers are not constrained from the others.

2.2.3 CNN

Convolutional Neural Networks (CNNs) are networks known for processing data which structure is a grid. An example are the images, where we can think an image as a grid of pixel (2D grid in grey-scale images, 3D in RGB images). We can

define a CNN as a Neural Network that use ***convolution*** in place of general matrix multiplication in at least one of its layer.

Here, in a Convolutional layer, we have an *input*, a *kernel*, and the output of the operation between the input and the kernel, which is called *feature map*. Usually, the input is a multidimensional array of data, and the kernel is a multidimensional array of parameters that will be learn. So, if we continue to think about the image example (grey scale, 2D), the convolution layer will be defined as following:

$$S(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n).$$

This function is also called *convolution*.

Often, we apply a similar operation called ***cross-correlation***, which is defined as:

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n).$$

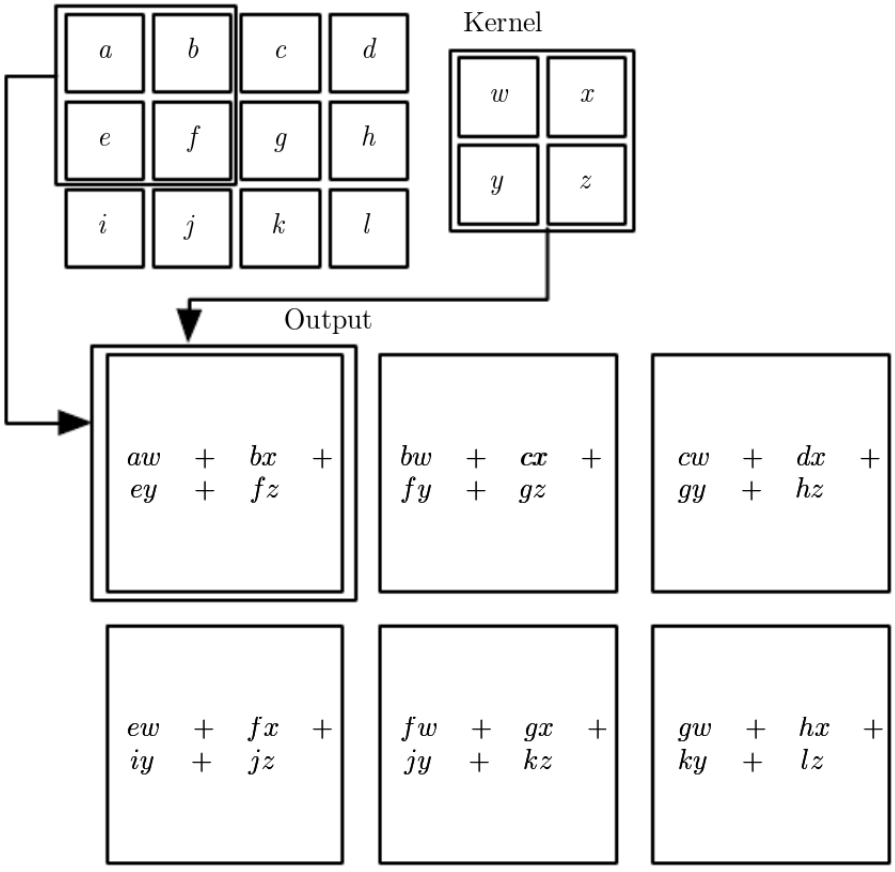


FIGURE 2.3: Convolutional Neutal Network in a 2D space

In Figure 2.3 we see an example of the CNN, where we have a kernel multiplication with the possibilities in the input.

Pooling The CNNs can be defined by three major steps:

1. the layer perform several convolutions and produce a set of linear activation;
2. each linear activation is process in a non linear activation function (*detector stage*);

3. *pooling stage*, which is a function that modify the output of the previous stage.

The idea behind the pooling is to replace the output of a convolution layer with its summary statistics. We can find a lot of pooling function, for example:

- *max-pooling* where it replace the output of the detector stage with the maximum of a rectangular neighborhood (e.g. given a vector $(1, 0, 2, 3)$ and a rectangular neighborhood of size 2, the output vector will be $(1, 2, 3)$);
- *average-pooling* where, instead of using the max function for the rectangular neighborhood, we use the average.

2.2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a family on Neural Networks for processing sequential data, such as text. Consider the sentences "I went to Nepal in 2009" and *In 2009, I went to Nepal*: the fact that 2009 is in the sixth or second position should not be relevant for the meaning of the information. In a traditional fully connected neural network, each feature is connected with some weight, so each word will be process based on the position on the sentence. So, in general, the input of a RNN is a vector $X^{(t)}$, where t is the time step which range is from 1 to T, as shown in Figure 2.4.

In the RNN have loops that allow the information to persist. This is done by the state called *hidden state*.

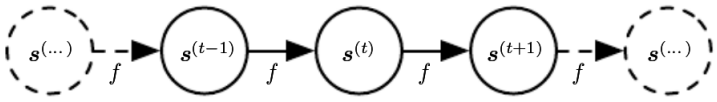


FIGURE 2.4: The dynamic of a RNN: each node represent a state at some time t , and the function f map the state t to the state $t + 1$

In our analysis we considered two type of RNN: *LSTM* and *GRU*.

Long Short Term Memory The LSTM networks are Neural Networks that can learn long-term dependencies. In the RNNs we have units that are repeated in the networks, and this can be seen as a chain. The LSTM units contains:

- *cell state*: only minimal interactions in the chain;
- *gates*: they are used for add or remove information carefully in the cell state; they are composed by a sigmoid neural network layer and pointwise multiplication; the role of the sigmoid is to describe how much the new information will be important for the cell state.

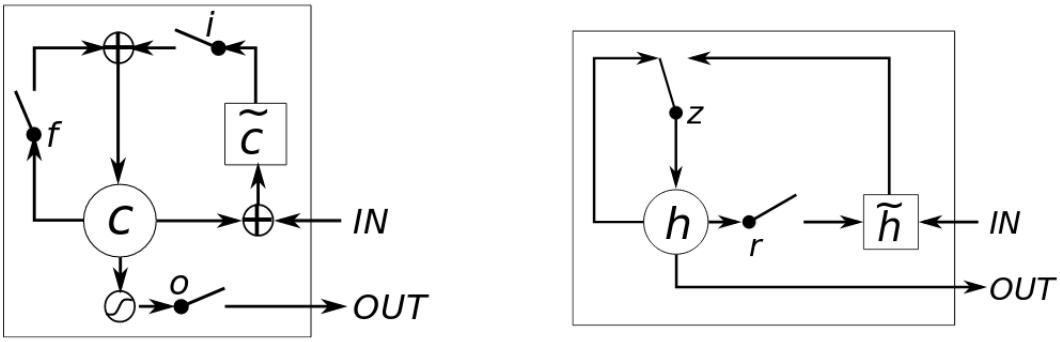
The input at time t is so used for modifying the memory of the network (the cell state) but it will be also combined to the cell state itself and the previous hidden state, for generating an output and a new hidden state. The LSTM unit works with these three components: hidden state, cell state and input.

The LSTM unit have the ability to remove or add information. As describe in (Hochreiter and Schmidhuber, 1997) and (Gers, Schmidhuber, and Cummins, 2000), the j^{th} units contains:

- *multiplicative input gate unit*: protect the memory context stored in j from the perturbation by irrelevant inputs;

- *output gate unit*: protects other units from the perturbations generated in the current unit.

Gated Recurrent Unit Chung explain the differences between the LSTM and the GRU (Chung et al., 2014), where GRU was defined in (Cho et al., 2014). The GRU, like the LSTM has gating units that handle the flow of information inside the unit but without having a separate memory cells. This make the structure of the GRU more lighter and the train it is faster, while the performances were comparable.



(A) LSTM (B) GRU

FIGURE 2.5: Differences between LSTM and GRU

2.2.5 Language Models

A Language Model (LM) is a probability distribution over sequences of words (Grave, Joulin, and Usunier, 2016). Given a dictionary D and V its size, each word will be represented as a one hot encoding vector of size V. Here, the word w_i will be represented by the vector x_i, where it is a vector with all zeros and in position i 1. Using the chain rule, the probability assigned to a chain of words is defined as the following:

$$p(x_1, ..., x_T) = \prod_1^T p(x_t | x_{t-1}, ..., x_1).$$

Recurrent Neural Networks, such as the LSTM, are perfect for this task, in the RNNs, the parameters are learned by minimizing the error in the prediction of the i^{th} word in a sentence, given the previous.

2.2.6 Loss and Optimizers

During the training phase, the learning is given by minimizing the *loss* that is made during foreword step, and this minimization is done by updating the weights (parameters) of the model, in that step called backward. There are several loss functions, and some of that are:

- *mean square error*, defined as the average of the square errors between the predictions and the real values (Gareth James, 2015):

$$MSE = \frac{1}{n} \sum_i^n (y_i^{true} - y_i^{pred})^2;$$

- *mean absolute error*, similar to the MSE, but in this case the error is in absolute value (Chollet and François, 2015):

$$MAE = \frac{1}{n}\sum_{i}^{n}|(y_i^{true} - y_i^{pred})|;$$

- *cross entropy*, which is a measure for estimating how much 2 distributions are close. If the distributions can assume only two values, Keras call this case *binary cross entropy*, otherwise it is called *categorical cross entropy*:

$$CE(o_d, t_d) = -\sum_{d \in D} t_d log o_d + (1 - t_d) log(1 - o_d),$$

  where o_d is the the distribution calculated, while t_d is the target distribution (Mitchell, 1997);

- *Kullback-Leibler divergence*: made by Kullback (Kullback and Leibler, 1951), is an information-based measure of disparity among probability distributions; given two distribution defined over X, P and Q, where Q is absolutely continuous respect to P, the measure of the divergence of Q from P is the $P - expectation$ of $-log_2(\frac{P}{Q})$ (Joyce, 2011):

$$D_{KL}(P, Q) = -\int x log_2(\frac{Q(x)}{P(x)}) dP.$$

The whole list of luss functions that are available and implemented for the Keras library are available in `https://keras.io/losses/`.

In general, the *optimizer* is used with the aim to minimize or maximize some functions, where in the NNs case is the loss function.

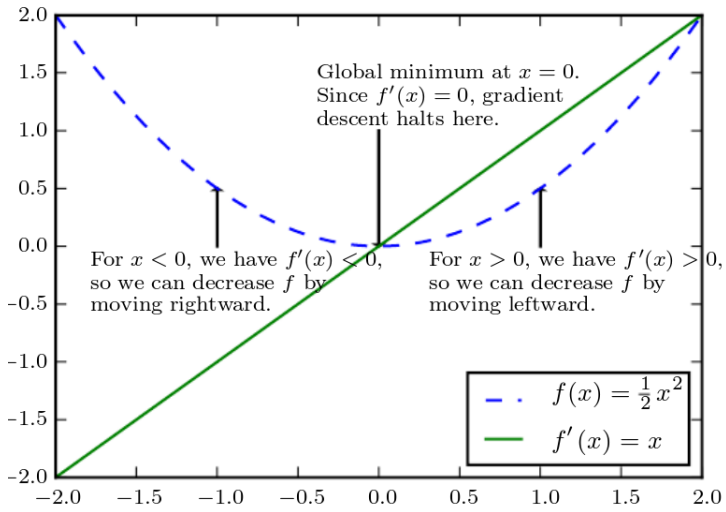


FIGURE 2.6: Gradient descent. Illustration of how the gradient uses the derivatives of a function for finding the minimum (Ian Goodfellow and Courville, 2016)

Given a function $y = f(x)$, the derivative $f'(x)$ give the slope of $f(x)$ in x. The derivatives gives us the information of the direction where the function grows, and the gradient uses this information for reaching the minimum / maximum of the

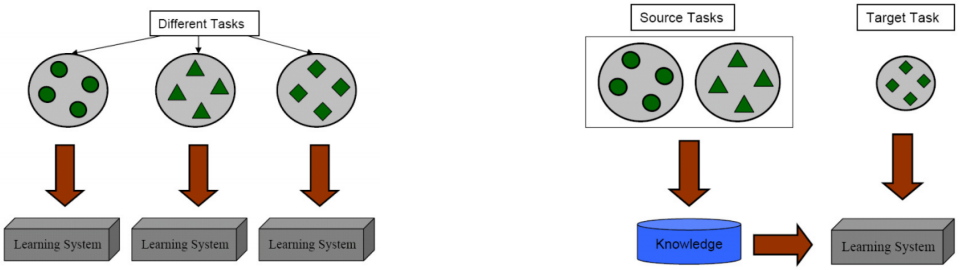
function. In Figure 2.6 it is explained how the gradient works. However, this technique has some performance limitation in the neural networks (Blum and Rivest, 1992) (Judd, 1990).

The process of minimizing the loss function can be accelerate by using other techniques, such as:

- *Stochastic Gradient Descendent* (SGD), where the idea is to obtain an estimation of the gradient by using minibatch (small subset of the training set) of m samples (Ian Goodfellow and Courville, 2016);
- *Adam*: this algorithm is efficient and it updates its parameters, such as the learning rate, on run time, during the training; this algorithm allows an efficient and a faster optimization compared to the SGD (Kingma and Ba, 2014).

2.3 Transfer Learning

. Machine learning techniques reached many successes in many tasks, from the regression to the classification or clustering (Wu et al., 2008) (YANG and WU, 2006). In general, the assumption is that both training and test set came from the same distribution. It means that if the distribution changes, the model must to be built from scratch. Pan et. al. analyse this aspect of the machine learning in "A Survey on Transfer Learning" (Pan and Yang, 2010). Building a model from scratch could be really expensive and we should be able to have a model built using some information of another existing model, in order to save time and resources. This is called *knowledge transfer* or *transfer learning*. The motivation behind the transfer learning is to use some existing knowledge for solving new tasks, for example the recognition of an a cat should be similar to the recognition of a dog.



(A) Traditional Machine Learning (B) Transfer Learning

FIGURE 2.7: A comparison between the traditional machine learning and transfer learning

Figure 2.7 (Pan and Yang, 2010) show the differences between the traditional approach and the transfer learning.

Definition 1. *Given a source domain D_S and learning task T_S, a target domain D_T and a learning task T_T, transfer learning aims to help improve the learning of the target function $f_T()$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.*

Definition 1 define in a formal manner what the transfer learning is.
Transfer learning task can be summarized in three sub-questions:

1. *What to transfer*: we need to understand what knowledge can be useful and what cannot; when we take the knowledge of a model we need to consider

that some knowledge are specific of that domain and other could be used for other domains;

2. *How to transfer*: when we now the useful knowledge, we need to develop an algorithm for transfer these knowledge;
3. *When to transfer*: when is useful to transfer and when we should not apply the transfer. In some situations we have that the source domain and the target domain are not related and a transfer of the knowledge could produce a *negative transfer* with bed performances for the model.

And three possible settings are available:

- *inductive transfer learning*, where target and source tasks are different (no matter if the domains are the same or not). Here, we have some the label data which is used for building the function $f_T()$; about the source domain, we can have 2 cases:
  - labeled data, and we use them with aim to achieve high performances in the target task;
  - unlabeled data, where the domain of the source data could be also different compared to the target source domain (it means that some information cannot be used directly) ;
- *transductive transfer learning*, where the source and target tasks are the same, while the domains are different. Here, the source data is labeled while in the target data not (unlabeled); here the feature spaces could be the same or different ;
- *unsupervised transfer learning*, where the target task is similar / related to the source task. Usually this choice is done for unlabelled data with purpose of dimensionality reduction or clustering.

2.4 Evaluation Metrics

Metrics are fundamental for evaluating and compare models. In this section I will introduce the metrics that I will use for my analysis.

2.4.1 Accuracy

The accuracy is one of the techniques used for evaluating the performances of a model. We can describe the accuracy as follow:

$$accuracy = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ Predictions}$$

.

Accuracy gives a global evaluation of the performances of the classifier.

2.4.2 Precision, Recall and F-measure

One common way for evaluating the models is based on the *precision*, *recall* and *F-measure*.

The binary case

The binary classification is a Machine Learning instance where the outcome is expressed by a discrete and binary values. Usually this is represented by "+" and "-" (Powers, 2011). In the binary classification the outcome cannot be both values together. Given the 2 classes, we can have four cases of predictions regions, as shown in Table 2.1.

		Predicted Class	
		-	+
True Class	-	TN	FP
	+	FN	TP

TABLE 2.1: Confusion Matrix

Let's now define the precision and recall:

$$Recall = \frac{TP}{TP + FN},$$

$$Precision = \frac{TP}{TP + FP}.$$

In the binary context the precision is the same as the accuracy metric.

This two metrics evaluate only the positive examples and predictions (class +), which mean that neither of them capture any information about the negative cases. This is applicable also to the F-measure, which is defined as the weighted harmonic mean of the precision and recall.

F1 score

The F1-score (also known as F-measure) is a metric that uses the precision and recall of the classes for give an evaluation of the model. Here, I consider the method provided by Scikit-Learn (Pedregosa et al., 2011) and it offers different and useful variations. This can be applied for both binary and multi-label classification, where in the latter case it is defined as the weighted average of the F1 score of each class.

The basic definition of the formula is described as following:

$$F1 = 2 * \frac{precision * recall}{precision + recall}.$$

As explained previously, the weight in the average has different effects, and Scikit-Learn offers the following:

- *binary*: it reports the result for the class which is defined as positive (only one class);
- *micro*: calculate metrics globally by counting the total true positives, false negatives and false positives;
- *macro*: calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account;
- *weighted*: calculate metrics for each label, and find their average, weighted by support (the number of true instances for each label). This alters "macro" to account for label imbalance; it can result in an F-score that is not between precision and recall;

- *samples*: calculate metrics for each instance, and find their average (only meaningful for multi-label classification where this differs from accuracy).

2.4.3 AUC

The Area Under the (ROC) curve is an estimation of how well the model performs. The ROC (Receiver Operating Characteristics) curve displays simultaneously the two types of errors (e.g. true positive and false positive) for all the possible thresholds. the AUC is the area under this curve, where the best result is 1, and we expect a model with an AUC of at least equal to 0.5 (Gareth James, 2015). An example of the plot of the AUC is given in Figure 2.8.

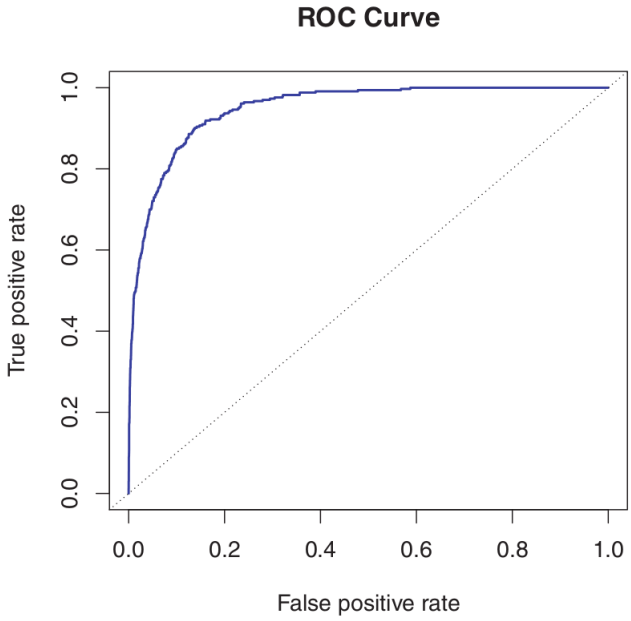


FIGURE 2.8: AUC example

2.5 Machine Learning Evasion

This section is a summary of the work of Biaggio et. al. (Biggio et al., 2013). Machine Learning is widely used in security-sensitive applications such es spam filtering, malware detection and network intrusion detection. Samples which are used for the prediction by the machine learning can be affected by other AI for example, in order to confound the learning. If we think to the spam detection problem, an adversary can avoid the detection by obfuscating those words that the classifier uses for understanding if the email is spam or not. When we design a model, we should use *preactive* protection mechanisms for preventing the impact of an adversarial attack; for doing that, we need to:

1. finding potential vulnerabilities of the model before that an adversary exploits the weakness;
2. investigate the impact of the vulnerabilities;
3. try to find some countermeasures to those attacks that can significantly impact to the performances of our model.

2.5.1 Evasion at test time

A classification algorithm is defined as:

$$f : X \to Y,$$

which is just a function which, given an input, it will returns some labels associated to that input. The labels are predefined.

Adversary Model First, we need to do some assumptions:

- What is the adversary knowledge?;
- What is the capability to modify and to manipulate the input data?.

The goal of the adversary is to defined in terms of utility function that the adversary want to maximize (or minimize). The aim is to have a miss-classified sample.

Adversary's knowledge The adversary's knowledge can include:

- the training set or a part of it;
- the feature representation of each sample (how samples are mapped into the feature space);
- the type of the learning algorithm and the form of the decision function;
- the trained classifier model;
- the feedback from the classifier.

Adversary's capabilities In this scenario, the adversary can only modify the test data and it cannot touch other components, such as the training set. In this restriction, the adversary's choices are:

- modification of the input data (limited or unlimited);
- modification to the feature vector (limited or unlimited);
- independent modifications to specific features.

In general we can think that the attacker can manipulate every feature while what is important is the level of modifications allowed.

2.5.2 Attacks scenarios

The attacker can has:

- *perfect knowledge*, where he / she knows perfectly every aspect of our classifier, from the training set, to the model (feature space, model type, hyper parameters). In this scenario the attacker try to miss-classify a sample with less modification as possible. Here the attacker know the function f that classify and try to find its weakness;

- *limited knowledge*, where, as in the previous scenario, the attacker wants to miss-classify the samples, but the knowledge is more limited in knowing for example the feature space (what are the features that the model uses) or the type of the model, but it does not know nothing about the final model or the training data. Under this scenario the attacker try to approximate the real function f (e.g. if we can have access to the classifications of some samples).

2.5.3 Adversarial Machine Learning

As described previously in this section, the goal of an attacker is to use some adversarial examples that fool the classifier (miss-classification). The aim of the attacker is to modify a sample s that is classified as c_i in such a way that s^* (the modified s) is perceptually indistinguishable from c_i but it is classified as c_j. The idea behind the ***adversarial machine training***, as explained by Kurakin et. al. (Kurakin, Goodfellow, and Bengio, 2016), is to inject adversarial examples in the training data. This means that, for each sample of the dataset, we will insert also a modify version which looks like an attacker attacks. In this way the model should learn and be more resistant over that specific class of attacks.

Chapter 3

Hate Speech and Related Works

Our study is focused on the hate speech detection. In this chapter I will discuss about hate speech, how is defined from the law and NLP community point of views (sections 3.1, 3.2), and the aspects / consequences that concerns this phenomena (Section 3.4). Moreover, in Section 3.3 we will show the importance of the swear words in this topic.

3.1 The European Court of Human Rights

What is "hate speech"? In June 2018, the European Court of Human Rights touched this topic (Human Rights, 2018) and reports the a part of the judgment of July 6 2006, Erbakan v. Turkey, as reported in Definition 2.

Definition 2. *[...] Tolerance and respect for the equal dignity of all human beings constitute the foundations of a democratic, pluralistic society. That being so, as a matter of principle it may be considered necessary in certain democratic societies to sanction or even prevent all forms of expression which spread, incite, promote or justify hatred based on intolerance . . . , provided that any "formalities", "conditions", "restrictions" or "penalties" imposed are proportionate to the legitimate aim pursued. (ECHR, 2006)*

One important point to consider is the *freedom of expression*, where we need to understand where is the limit of the latter. In Definition 3 we give a definition of it.

Definition 3. *Freedom of expression constitutes one of the essential foundations of [a democratic] society, one of the basic conditions for its progress and for the development of every man. Subject to paragraph 2 of Article 10 [of the European Convention on Human Rights], it is applicable not only to "information" or "ideas" that are favourably received or regarded as inoffensive or as a matter of indifference, but also to those that offend, shock or disturb the State or any sector of the population. Such are the demands of that pluralism, tolerance and broad-mindedness without which there is no "democratic society". This means, among other things, that every "formality", "condition", "restriction" or "penalty" imposed in this sphere must be proportionate to the legitimate aim pursued. (ECHR, 1976)*

The European Court of Human Rights uses two approaches for dealing with cases of incitement of hatred and freedom of expression (Human Rights, 2018):

- the approach of exclusion from the protection of the Convention, provided for by Article 17 (prohibition of abuse of rights)1, where the comments in question amount to hate speech and negate the fundamental values of the Convention;
- the approach of setting restrictions on protection, provided for by Article 10, paragraph 2, of the Convention2 (this approach is adopted where the speech in question, although it is hate speech, is not apt to destroy the fundamental values of the Convention).

Definition 4. *Article 10: Freedom of Expression (European Court of Human Rights)*

1. *Everyone has the right to freedom of expression. This right shall include freedom to hold opinions and to receive and impart information and ideas without interference by public authority and regardless of frontiers. This Article shall not prevent States from requiring the licensing of broadcasting, television or cinema enterprises;*

2. *The exercise of these freedoms, since it carries with it duties and responsibilities, may be subject to such formalities, conditions, restrictions or penalties as are prescribed by law and are necessary in a democratic society, in the interests of national security, territorial integrity or public safety, for the prevention of disorder or crime, for the protection of health or morals, for the protection of the reputation or rights of others, for preventing the disclosure of information received in confidence, or for maintaining the authority and impartiality of the judiciary.*

Definition 5. *Article 17: Prohibition of abuse of rights (European Court of Human Rights)*

Nothing in this Convention may be interpreted as implying for any State, group or person any right to engage in any activity or perform any act aimed at the destruction of any of the rights and freedoms set forth herein or at their limitation to a greater extent than is provided for in the Convention.

3.2 Hate Speech in the NLP Community

In the previous section we defined the hate speech from the European Court of Human Rights point of view. In this section I will discuss some aspects about the topic from the NLP community point of view.

One simple definition is given by Nockleby, where the hate speech is seen as any kind of communication that disparages a person or a group on the basis of some characteristics such as race, colour, ethnicity, sexual orientation, nationality religion or other characteristics (Levy, Karst, and Winkler, 2000).

For having some concrete examples of these sentences, we can consider some hateful sentences provided by Schmidt et. al. (Schmidt and Wiegand, 2017):

- Go fucking kill yourself and die already pile of shit scumbag;
- The Jew Faggot Behind The Financial Collapse;
- Hope one of those bitches falls over and breaks her leg.

As described by Schmidt et. al. in their survey of the hate speech, in the NLP community there are several tasks that try to study and analyze the phenomenon or its sub-classes (Schmidt and Wiegand, 2017), such as:

- *hate speech* (Schmidt and Wiegand, 2017) (Gröndahl et al., 2018) (Zeerak and Hovy, 2016) (Davidson et al., 2017) (Warner and Hirschberg, 2012a);
- *abusive messages* (Spertus, 1997);
- *hostile messages* (Spertus, 1997);
- *flames* (Spertus, 1997);
- *cyberbullying* (Hosseinmardi et al., 2015) (Zhong et al., 2016) (Van Hee et al., 2015);

- *offensive languages* (Razavi et al., 2010);
- *personal insults* ("Automatic identification of personal insults on social news sites");
- *profanity* (Sood, Antin, and Churchill, 2012);
- *toxicity* (Wulczyn, 2017).

3.3 Offensive vs. Hate Speech

An important distinction is done by Davidsio et. al. where it considers also the problem of the offensive messages (Davidson et al., 2017). In this work, Davidson defined the hate speech as *as 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*. In this definition the authors made an important distinction with the language that uses offensive terms and hate speech. In this section I try to analyze the importance of the distinction between these two categories.

The use of the offensive language is common in the spoken language; for example, Wang et. al. show that in a normal day of a student, the use of the swear words is around the 0.5% of the total used words (Mehl and Pennebaker, 2003) (1 swear word every 200 words).

Swear words are also called *taboo words*. But in which occasions are they used? Jey analyze the phenomenon (Jay, 2009); the use of swear words depends on the context and on the purpose of the speaker and they are used for expressing a wide set of emotions, from angry to joy, from frustration to surprise: this means that we can use these words for positive and negative emotions. In general we have two forms of use: *spontaneous form*, where we do not have a lot of control of what we are saying, and *reflective form*, where we can think of what to say. Clearly, we cannot consider hateful some sentences such as "*Holy shit! Fuck me!*", which denotes a surprise emotion, just because they contain some swear words. So, while taboo words are associated as feature of sexual harassment, blasphemy, hate speech or discrimination, they can also be used for positive outcomes, such as jokes, humor, sex talk, in-group slang, storytelling, ironic sarcasm in order to promote social harmony (e.g. "*this CD is fucking great*"). In these cases the swear words are used with out the intent of being offensive (not offensive for someone, but just offensive as language).

Davidson suggest a distinction between these 2 classes (hate speech and offensive language) because the hate speech has some legal and moral consequences while offensive language has not an intent to offend or damage someone. This distinction must to be considered in order to avoid mistakes in the recognition, where the detector can decide their judgment based just on swear words.

3.4 The Effect of the Hate Speech

Hate speech is something that we need to take care, and we need to understand why is important to block it. In this section I will analyze some effects of the hate speech in the people's life, based in some psychological studies.

3.4.1 Cyberbulling

One of the aspects of the hate speech is the cyberbullying (Schmidt and Wiegand, 2017). Smith et. al. studied the impact of this phenomenon in the life of teenagers (secondary school students) (Smith et al., 2008).

Definition 6. *Cyberbullying is bullying using messages and internet as method of propagation(Smith et al., 2008).*

In Definition 6 we can understand what the cyberbully is, where a "bullying" is defined as being an aggressive,intentional act or behaviour that is carried out by a group or an individual repeatedly and over time against a victim who can not easily defend him or herself (Olweus, 1993).

Smith et. al. show that this phenomenon start becoming important in 2008, thanks to the ease of use of the new technologies such as smart-phone (with instant messages). Table 3.1 shows a comparison between the bullying and cyberbullying in 2008.

Frequency	Bullyied	Cyberbullyied
Often	14.1%	6.6%
Occasionally	31.5%	15.6%
Never	54.4%	77.8%

TABLE 3.1: Comparison of the percentage of teenager which were bullied / cyberbullied in 2008. *Often*: from several times a week to few times per month, *occasionally*: happened just few times, *never*: never happened (Smith et al., 2008)

Another interesting study is done by Whittaker et. al., where they study the impact of the social media and the relations with the cyberbullying (Whittaker and Kowalski, 2014). In this work authors conducted three study cases over 196 female and 75 male (students, age from 18 to 25, average 18.8, standard deviation 1.2 8.6% Afro-American, 91.4% American). Here, I will analyze just the *Study 1*.

Technology	Usage	Victims
text	99.6%	56.8%
e-mail	98.4%	/
Facebook	86.5%	38.6%
YouTube	75.1%	11.4%
Instagram	70.9%	13.7%
Twitter	69.4%	45.5%
Instant messaging	14.5%	2.3%
Chat rooms	2.3%	2.3%

TABLE 3.2: *Usage*: percentage of the people which often or frequently use the technology; *Victims*: percentage of people which were victimized (Whittaker and Kowalski, 2014)

Study 1 The aim of this study is to understand the degree to which social media are related to the cyberbullying. The results shows that the 77% reported that they felt safe by using social media. Table 3.2 shows the collected statistics.

Definition 7. *Scale for Suicidal Ideation The Scale for Suicidal Ideation (SSI) it is a scale for the evaluate the idea of a suicide of an individual; the Suicidal Ideation is the previous act before the suicidal. It is important to identify the intensity of this phase for predicting the probability that a specific individual will commit a suicide (T. Beck, Kovacs, and Weissman, 1979).*

Unfortunately, the cyberbullying has also bad effects in the victims. Hinduja et. al. in 2010 conducted an analysis on the correlation between the cyberbullying and suicides (Hinduja and Patchin, 2010). In their study, they consider a sample of 1973 students and the measurement was the *suicidal ideation*, which include four questions with answer *yes / no*:

- have you ever felt so sad or hopeless almost every day for two weeks or more in a row that you stopped doing some usual activities;
- have you ever seriously thought about attempting suicide;
- have you ever made a specific plan about how you would attempt suicide;
- have you ever attempted suicide.

In the analysis they consider 4 aspects of the bullying: *traditional bullying victimization*, *traditional bullying offending*, *cyberbullying victimization* and *cyberbullying offending*. Let's focus only in the *cyberbullying victimization* point: the *cyberbullying victimization* represents the respondent's experience in the last month and the choices as answer of the task (Table 3.3) were 5, from "never" to "every day".

Type	**Percentage**
Received an upsetting email from someone you know	18.3%
Received an instant message that made you upset	16.0%
Had something posted on your MySpace that made you upset	14.2%
Been made fun of in a chat room	10.0%
Received an upsetting email from someone you didn't know (not spam)	9.7%
Had something posted about you on another web page that made you upset	9.5%
Something has been posted about you online that you didn't want others to see	9.2%
Been picked on or bullied online	9%
Been afraid to go on the computer	5.7%
One or more of the above, two or more times	29.4%

TABLE 3.3: Cyberbullying victimization over the last 30 days (Hinduja and Patchin, 2010)

The results about the suicide are: 20% reported seriously thinking about the suicide and 19% attempting suicide. Authors noticed an high correlation between samples which have been victimized by episode of bullying and cyberbullying and the *suicidal ideation*. This raise some important aspects of the hate speech and how dangerous it could be.

3.4.2 Sexism

Another form of hate speech on social networks is the *sexism* and Fox et. al. try to understand what are the effects on sexist comment on Twitter (Fox, Cruz, and Lee, 2015).

Study case: Twitter 172 samples were selected (86 males and 86 females) for the test. The age varies from 18 to 41 (mean = 20.7, standard deviation = 3.82) and there were different countries. The samples were randomly divided equally for the four tasks:

- *anonymous re-tweet*;
- *anonymous create*;
- *identified re-tweet*;
- *identified create*.

For the tasks, the participants were asked to use a public Twitter account, which was followed by other students. In the anonymous conditions, every account had a generic user-name and an avatar as profile photo, followed by 150 users. In this way the anonymity of the users behind these accounts were guaranteed. The second case is the identified one, where the accounts were associated with their real names and real information. Participants were asked to either re-tweet existing tweets using the hashtag or write their own tweets using the hashtag. After assigning these profiles, participants were involved in a survey with questions related to the hostile sexism items. The third step, participants were asked to evaluate the resume (Curriculum Vitae) of two females and two males candidates.

The results of this study were:

- more sexism hostility for anonymous participants rather than non anonymous participants after tweeting;
- participants who composed sexist tweets were more hostile in the evaluation of female candidate (less competent than male), while this phenomenon was more restricted for participants which just retweeted .

3.5 Hate Speech Representation

Schmidt and Wiegand wrote a survey on these topic which give an overview of what is going on in the research area and what are the approaches involved in the topic (Schmidt and Wiegand, 2017). First, they shows the features of the *hate speech*:

(i) *Simple Feature Space*: the use of surface-level features, such as bag of words; unigram and ngrams are highly involved, where sometimes they are use with in addition of other features; character ngrams can helps with the problem of the spelling (e.g. mistakes / typos in the words), for example sentences such as *kill yrself a$$hole* are highly hateful, but they can be a problem for models which uses words. Other surface information are the number of URL, lengths of sentences, words not available in the English dictionaries, the number of non-alpha numeric characters present in tokens;

(ii) *Word Generalization*: the limitation of the bag of word is that the words must to be present in both training and test set, in order to achieve good performances; *word clustering* is often used in these cases, where the words are replaces with the IDs of their cluster (every cluster represents a set of words). Another approach is the *word embeddings*, where each word is represented as a vector and similar words will have similar vectors (they will be close in the hyperspace). A similar approach is the *paragraph embedding*, where the sentence is represented with a unique vector which is, for example, calculated by the average of the word embedding that the sentence contains. An example of this category is shown in (Warner and Hirschberg, 2012b);

(iii) *Sentiment Analysis*: usually hate sentences are related to negative sentiments. In several analysis of the hate speech, this feature used for the classification;

(iv) *Lexical Resources*: some assumption can be done, such as thinking that hateful sentences contains some specific negative words; there are some sources which give the list of these words that have been collected over the years in the web (e.g. *hatebase.org*);

(v) *Linguistic Features*: linguistic features can be used for catching the relationship between words; for example, given the sentence *Jews are lower class pigs*, the linguistic feature can capture the relation between *Jews* and *pigs*;

(vi) *Knowledge based features*: the hate speech cannot be detected by just looking for some keywords; if we consider the sentence *Put on a wig and lipstick and be who you really are*, this is not hateful. If the sentence is isolated from the context, it is difficult to detect the hate speech; if the previous sentence was posted over a photo of a male guy, it could remark the sexuality orientation of the target person. The context play an important role in the hate speech;

(vii) *Meta Information*: we can consider other information as feature, such as the gender of the sender (who write the message), and the receiver (who receive the message), but also the age, sexual orientation, political orientation, country, nationality, ...;

(viii) *Multi-modal Information*: social are not based only on the text but also on images. We could use the information of the image for having other features and for knowing better the context of the sentence.

In our analysis we will exclude (vii)-(viii), because we focus on the detection of the hate speech based only on the mere text.

3.6 Spam Detection and Evasion

In our knowledge, in the hate speech classification task nobody considered the problem of the evasions. In a similar task, the spam detection, several approaches have been proposed. One example is the text obfuscation in Bayesian models (Stern, Mason, and Shepherd, 2004), where the model were attacked by injecting poisoned spam samples; from a group of 1000 common English words, two were taken and add to the object of the email and 100 were added to the corpus. The classifiers that were considered for the study had a degradation of the performance during the attack.

Chapter 4

Dataset

In this work we did not collect data for the experiments but we rely on datasets which are available on-line. The datasets focus on partially disjoint aspects of "hate", e.g. hate speech based on religion and ethnicity may not be highly similar to sexual connotated hate speech.

The datasets that we used are:

- Wikidetox (W): Toxic - Non Toxic (Wulczyn, 2017);
- Twitter (T1): Hate Speech - Offensive - Neither (Davidson et al., 2017);
- Twitter (T2): Racist - Sexist - Neither (Zeerak and Hovy, 2016);
- Twitter (T3): Hateful - Neither (Zhang Ziqi, 2018).

In this chapter I will describe (Section 4.1) and analyze the properties (Section 4.2) of the dataset that we used for this study.

4.1 Datasets Introduction

4.1.1 Wikipedia (W)

The dataset is a part of Wikipedia's "Detox" project targeting personal attacks in Wikipedia's edit comments. As described by Wulczyn et al. (Wulczyn, 2017), labels were gathered via crowd-sourcing, which is defined by the following steps:

- generation of a corpus based on Wikipedia comments ;
- choosing a question for the judge of humans;
- take a subsample of the corpus for the labeling purpose;
- designing a strategy for having reliable labels.

We denote this dataset as *W*. This dataset is a combination of two datasets:

- comments: the text which need to be judge;
- annotations: it contain the judgment of a comment of a specific crowd-worker.

The annotator can label the comment as:

- targeted at the recipient of the message (e.g. you suck);
- targeted as a third party (e.g. Bob suck);
- being reported or quoted (e.g. Bob said Henri sucks);

- another kind of attack;
- this is not an attack or harassment.

Here, each comment is labeled (judged) 10 times and a comment is referred as attack if the majority of annotators labeled it as such.

4.1.2 Twitter 1 (T1)

Davidson et al. (Davidson et al., 2017) presented a dataset with three kinds of comments from Twitter: hate speech, offensive but non-hateful speech, and neither. This is, to our knowledge, the only dataset with such a distinction. The hate speech data was collected by searching for tweets with known hate speech phrases, 3 and further labeling these tweets with a majority vote from three CrowdFlower workers each. We denote this dataset as T1. For the collection of the samples, Davidson et al. decide to use *Hatebase.org*, which is a database which contains examples of hate speech phrases collected over the web. Authors uses the *Twitter API* for collecting tweet which contains terms from the lexicon derived by Hatebase.org. The dataset contains 24783 samples, where 1430 are classified as hate speech (5%), 19190 offensive but not hateful (76%), and 4163 as neither (19%). The whole dataset contains 40817 words. For some experiments, we combined the offensive class with the neither one, and we denoted this dataset as T1*.

4.1.3 Twitter 2 (T2)

This dataset is collected by Waseem et al. (Zeerak and Hovy, 2016) and it collected data using the Twitter platform. This dataset contains three possible labels:“racism”, “sexism”, and “neither”. In our experiments, we combined the “racism” and “sexism” classes into one, comprising general hateful material. We denote this dataset as T2. For identifying hate speech, authors proposed the following:

1. uses a sexist or racial slur;
2. attacks a minority;
3. seeks to silence a minority;
4. criticizes a minority (without a well founded argument);
5. promotes, but does not directly use, hate speech or violent crime;
6. criticizes a minority and uses a straw man argument;
7. blatantly misrepresents truth or seeks to distort views on a minority with unfounded claims;
8. shows support of problematic hash tags. E.g. “#BanIslam”, “#whoriental”, “#whitegenocide”;
9. negatively stereotypes a minority;
10. defends xenophobia or sexism;
11. contains a screen name that is offensive, as per the previous criteria, the tweet is ambiguous (at best), and the tweet is on a topic that satisfies any of the above criteria.

4.1.4 Twitter 3 (T3)

Zhang et al. (Zhang Ziqi, 2018) decided to create their own dataset based on Twitter, which we call T3, and in this case the hate speech targets refugees and muslims. They collected data by using Twitter streaming API, by monitoring some words (muslim, islam, islamic, immigration, migrant, immigrant, refugee, asylum) and some hashtag (#refugeesnotwelcome, #DeportallMuslims, #banislam, #banmuslims, #destroyislam, #norefugees, #notmuslims). .

4.2 Datasets Analysis

In this section I will analyze the statistical properties of these datasets and try to analyze the correlations between them.

4.2.1 Classes Distribution

As described in the Chapter 3, there are different form of hate speech. The dataset that we considered vary on this aspect, which mean that they analyze different aspect of the hate speech, from toxicity to racism. In general, the hateful class is the minority one (compared to the other of the same dataset). Also, the datasets have different sizes: T3 is a small dataset (2.5K sentences), T1 and T2 are medium datasets with around 20 thousand of examples, meanwhile W is a huge dataset with more than 100 thousand of items. In general, the hateful class(es) are the minority, with a consequence of the unbalanced dataset. Table 4.1 and Figure 4.1 summarize these aspects.

Dataset	Domain	Classes (size)	Source
W	Wikipedia	personal attacks (13590) ordinary (102274)	(Wulczyn, 2017)
T1	Twitter	hate speech (1430) offensive (19190) ordinary (4163)	(Davidson et al., 2017)
T1*	Twitter	hateful (1430) offensive ∪ ordinary (23353)	(Davidson et al., 2017)
T2	Twitter	racist ∪ sexist (5013) ordinary (10796)	(Zeerak and Hovy, 2016)
T3	Twitter	hateful/racist (414) ordinary (2021)	(Zhang Ziqi, 2018)

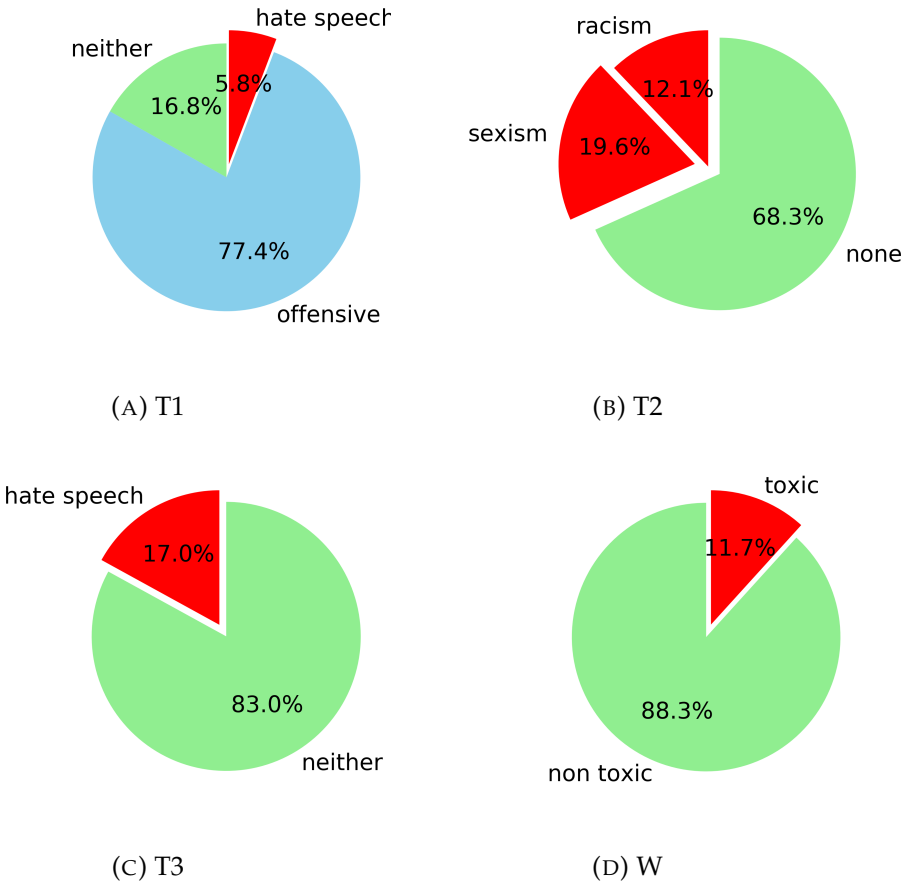
TABLE 4.1: Datasets used in our replication. Union (∪) denotes the conflation of class elements.

4.2.2 Sentences Examples

For understanding better the nature of the problem, I will show some examples of sentences for each class and for each dataset.

W

1. toxic:



(A) T1 (B) T2

(C) T3 (D) W

FIGURE 4.1: Classes distribution for each dataset

- Anon :What the heck are you talking about? This is an encyclopedia, not a book store;
- Thank goodness your interests arent purely rote, Larry, or the WP no doubt would be more less, and less more;
- On Google Vereenigde Oost - Indische Compagnie gets more hits than Vereenigde Oostindische Compagnie which is correct?;

2. non toxic:

- i have a dick, its bigger than yours! hahaha;
- Iraq is not good === === USA is bad;
- fuck off you little asshole. If you want to talk to me as a human [...].

In *W* it seems that the toxicity is generated from the use of an inappropriate language, generated for example with the use of swear words. From the three toxic examples, the first one seems (from my point of view), that the sentence is labelled as such just because of the word "dick"; the second one in my opinion does not have any sense of being there, while the third one is a clear example of an attack to a person (hate speech).

T1

1. hate speech

- We hate niggers, we hate faggots and we hate spics-kkk rally;
- Let's kill cracker babies!. WTF did I just hear???????? WOW;
- @MarkRoundtreeJr: LMFAOOOO I HATE BLACK PEOPLE. This is why there's black people and niggers;

2. offensive

- bad bitches is the only thing that i like;
- lames crying over hoes thats tears of a clown ;
- pussy is a powerful drug;

3. neither

- "Bae" sounds like such a ghetto word. Use something else;
- "Brownies for my brownie" I love this movie;
- "@riananewman: @22EdHam brownies tonight" what kind?!?!.

T1 seems to capture well the aspects of the three classes; the hate class contains examples of racism (first and third one), incite to hate (second one). The offensive, as also described by Davidson (Davidson et al., 2017), seems to contain swear words in its examples (e.g. bitches, hoes, pussy) while the *neither* class shows some examples of the common speech / dialogues.

T2

1. racism

- Drasko they didn't cook half a bird you idiot # mkr;
- @ummayman90 When people leave Islam, they don't announce it. They just quietly walk away;
- @anjemchoudary Idiots like you making such declarations have no contact with reality. Islam is inhuman and must be outlawed;

2. sexism

- I've had better looking shits than these two! # MKR2015 # MKR # killerblondes;
- Sorry # killerblondes but you are a long LONG way from refined. # mkr;
- Not dumb blondes...mmm...perhaps just delusions of adequacy? # MKR;

3. none

- # mkr at least there are judging honestly;
- My little lemon tarts #mkr;
- I score them a 3 and that's generous # MKR.

T2 contains samples of racism and sexism, where the racism's sentences attack the Islam; the sexism is contain some offense words against the female gender (e.g. "dumb blondes" in the third sentence).

T3

1. hate speech
    - Spain was islamic for 800 years. But we got rid of the vermin # BanIslam;
    - Tell Patagonia to go back to Patagonia! Syrian Iran ;
    - @FoxNewsResearch @toni_s_r_ # NoRefugees Keep USA safe!;
2. neither
    - @CNN Always a lone wolf when its Muslim. Every day # ramadan 2017 has given us a murder terror attach by stone age throwbacks. # NoRefugees;
    - @AMike4761 Starbucks committing suicide # NoRefugees ;
    - A staunch Tory Trump.

T3 follow the consideration of *T2*, where the hateful class seems with a political / religious tone versus other countries / races.

W	T1 HS	T1 Off	T2	T3
fuck	rt	bitch	rt	banislam
nigger	bitch	rt	sexist	banmuslims
shit	faggot	bitches	mkr	rt
suck	like	like	im	islam
ass	nigga	hoes	islam	muslim
faggot	ass	pussy	women	refugenotwelcome
hate	white	hoe	muslims	muslims
u	fuck	im	like	norefugees
go	trash	dont	dont	hate
like	nigger	ass	girls	bansharia

TABLE 4.2: Top 10 words for each offensive / hateful class in each dataset

The top hateful words, for each dataset seems to be similar, as shown in table 4.2, suggesting that there are some terms which are highly used in general in hateful sentences. Also, there are some words like "women", "like" or "go" which are used in the common language and alone are not hateful. Another particularity is given by the class of offensive and hateful of T1, where we have some common words which appear in both top 10 categories (e.g. bitch, ass, like).

4.2.3 Datasets Similarities

One of the aims of this research is to study how well the models scale. There are a lot of factors here to consider (e.g. architecture, feature representation, dataset, etc ...). The data try to capture different form of hate speech, and this could be a problem from the scalability point of view, because some terms could be not recognize as "problematic" for the model. The first question is to understand the similarity between vocabularies; for conducting this operation I used the parser *word_tokenize* provided by *NLTK* (Bird and Klein, 2009), where, given a sentence, it returns the tokens for which the sentence is composed by; for example, given the sentence:

Hello, how are you? I'm starving,

the algorithm will produce:

$$'Hello', ',', 'how', 'are', 'you', '?', 'I', "m', 'starving'.$$

This means that the vocabularies will contain words, but also punctuation and other symbols. As result of this operation, where all the sentences were treated with the lower case, W contains 215818 tokens, $T1$ 38038, $T2$ 25121, $T3$ 9687. The obtained sized shows that the huger the dataset, the bigger the vocabulary. What we can expected is that a huge vocabulary most likely will contain the words of the smaller dictionaries. In the first moment, let's see how these dictionaries are similar, where the evaluation is given by the following formula:

$$f(d1, d2) = size(d1 \cap d2)/size(d1).$$

Figure 4.2 the previous idea, where the small dictionaries of $T1$, $T2$ and $T3$ cannot cover well the dictionary of W which is huger; the opposite, W can cover better the other vocabularies. The vocabularies still very different, where in the best scenario one word is lost every two.

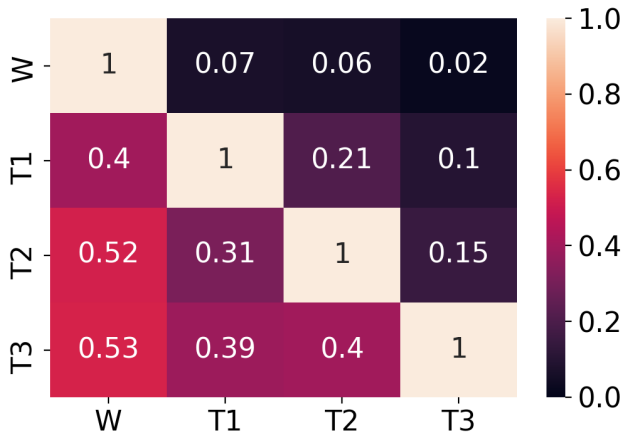


FIGURE 4.2: Similarity between the dictionaries; each dictionary in the rows represent $d1$, the dictionary that we want to cover, while the columns represent $d2$, the dictionaries used to cover

Given this assumption, we should think if this is a problem: the datasets seem widely different. The point here, is that maybe we are loosing just some specific words which appeared just once in the dataset, such as typos, misspellings, errors, or just names. For a better comparison we could, instead of looking for the similarity of the datasets, look for the similarity of the words distribution. In this case, the idea is that the common words are similar. The metric for the evaluation here is:

$$f(d1, d2) = \sum((d1 \cap d2).count) / \sum(d1.count),$$

where $d_i.count$ give the frequency of the words that appear in the dataset D_i.

Figure 4.3 shows that the dictionaries can cover most of the sentences of the other datasets. The best case is W which seems to loose one words every 10. This confirm the hypotheses that a lot of words appears just few times, as shown in the Figure 4.4; the word ranking is made by ordering the word by their frequency in a descendent way, and plot the ordered distribution, where in the x axis we have the rank of words

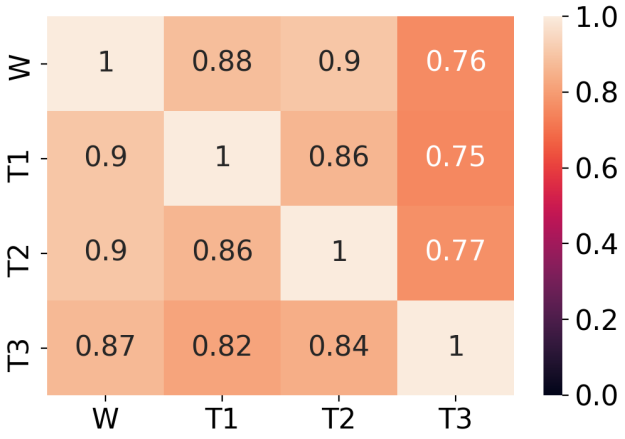
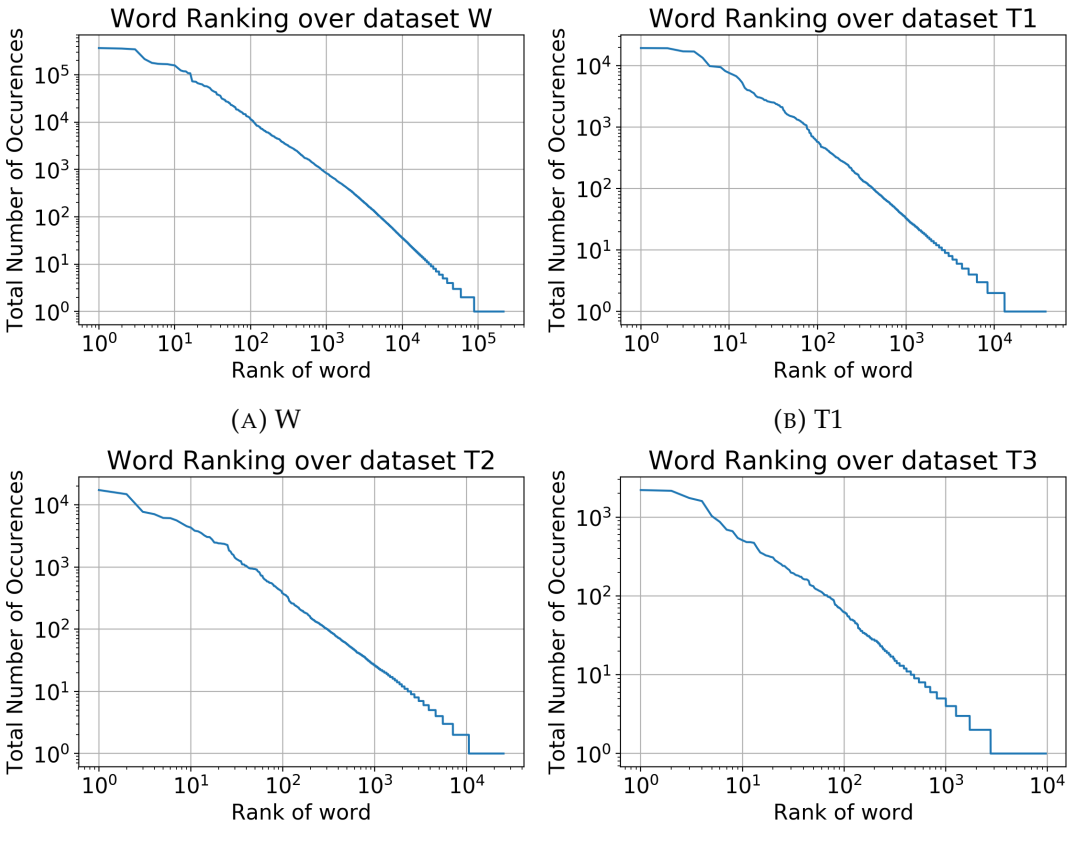


FIGURE 4.3: Similarity between the words distributions; each dictionary in the rows represent *d1*, the dictionary that we want to cover, while the columns represent *d2*, the dictionaries used to cover

and in the y axis the frequency. Now the question is: are these words relevant for the classification?



(A) W

(B) T1

(C) T2

(D) T3

FIGURE 4.4: Word ranking per dataset

4.2.4 Swear words

The swear words plays an important role in the classification, as mentioned in Section 3.3; for this reason, we should understand the presence of these words in the classes of the datasets. Since I do not have a list of all possible swear words, I decided to use some common words (Jay, 2009); the list of words are "fuck", "fucking", "shit", "hell", "damn", "goddamn", "ass", "bitch", "sucks". Here, I just count the number of occurrences inside each class and see how many swear words per sentence we have (the results will show the percentage of sentences affected by swear words).

- W: 1.27% (Non Toxic) / 29.7% (Toxic);
- T1: 28.1% (Hate Speech) / 50.4% (Offensive) / 1.03% (Neither);
- T2: 7.00% (Sexism) / 1.51% (Racism) / 3.52% (None);
- T3: 0.72% (Hate Speech) / 0.72% (Neither).

Clearly, these results can just have an idea of what is going on, because we are using just few swear words and not a detailed list. However, it seems that the toxic class of *W* contains more swear words compared to the non toxic one . *T1* seems to have the correct distribution of swear words (more in the offensive class, not a lot in the none class), while in *T2* and *T3* it seems that they contain the same amount for all the classes. This situation will be studied more in detail in Section 5.3, where we will see if the use of swear words will affect the decision of the classifier.

Chapter 5

Models & Performances

It is the time for understanding the models which are used for this research. We begin by describing four recent papers (Wulczyn, 2017), (Davidson et al., 2017) (Badjatiya et al., 2017), (Zhang Ziqi, 2018) on hate speech detection. We reproduce and systematically analyze the performance of seven models presented in these papers. Each paper studies different machine learning architecture for the hate speech detection. Here, the representation of a sentence is bases on two main approaches: TF-IDF vector based on word or character ngrams, or word embedding. The machine learning models vary from simple architectures such as the Logistic Regression or a Multilayer Perceptron to more complex and deep architecture such as the LSTM or the GRU. We can summarize the models that we analyzed in Table 5.1 and discuss them in the remainder of this chapter.

Model	Dataset(s)	Source
LR char	W	(Wulczyn, 2017)
MLP char	W	(Wulczyn, 2017)
LR word	T1	(Davidson et al., 2017)
CNN+GRU	T1*, T2, T3	(Zhang Ziqi, 2018)
LSTM	T2	(Badjatiya et al., 2017)

TABLE 5.1: Replicated machine learning models

After, we will analyze the performances of these models, by understanding how well they perform in their dataset, how well they scales, the errors types and how the handle the taboo words.

In this Chapter we will see a description of these models (Section 5.1), a comparison in the performances (Section 5.2). As suggested in the previous chapters, swear words are quite important in this distinction, so we will test the offensive class of T1 and we will see which label the sentences will be assigned (Section 5.3). In Section 5.4 we will try to focus on some prediction for each class and for each dataset, trying to understand why we have some predictions, and trying to understand how the models are "thinking".

5.1 Models Description

In this first part of the chapter I will describe the architectures of the models which are involved.

5.1.1 Logistic Regression and Multilayer Perceptron over W dataset

Let's start with the models which are studied by Wulczyn et al. in (Wulczyn, 2017). The models are applied for the dataset Wikidetox ("*W*".

This model propose two main model architecture:

- Logistic Regression;
- MLP.

In all the architectures, as last layer, there is a *softmax* function and use the *cross-entropy* as loss function. Authors studied the effect over 3 architecture variants:

- Model Architecture: Logistic Regression, Multi-Layer Perceptron;
- N-gram type: word, char;
- label type: One Hot, Empirical Distribution.

In order to provide reproducibility, authors provided a division of the samples (training, test and evaluation) with ratio 3:1:1. As evaluation metric, authors choose two metrics: the area under the receiver operating curve (AUC). The results of their work, Table 2 of (Wulczyn, 2017), is described in Table 5.2.

Model Type	N-gram type	Label Type	AUC
LR	Word	OH	94.62
		ED	95.55
	Char	OH	96.18
		ED	96.24
MLP	Word	OH	95.25
		ED	96.15
	Char	OH	95.90
		ED	96.59

TABLE 5.2: Original models and performances over W

The results are obtained using *train* as training set and the *dev* as evaluation set (train and dev are label of samples for allowing the replication). Authors provide also the best hyper-parameters for each model, so replicating the results was not difficult. As we can see, character models seems to perform better than word based model. Furthermore, for allowing the possibility of a comparison between the other dataset, which don't have the Empirical Distribution, we decided to continue to use only the models which use the One Hot encoding as labels (MLP char OH and LR char OH).

Both techniques uses n-gram as feature representation and they are weighted using their TF-IDF. A function of Scikit-Learn (Pedregosa et al., 2011) called *TfidfVectorizer* compute this transformation and convert a sentence into a vector of real numbers (vector representation).

Logistic Regression For the Logistic Regression, we take the top 10000 n-grams in the range [1, 5], which mean that we only consider the most common n-gram for the representation. As said previously, here we are using n-grams over characters. In this case the TF is also scale using the following formula:

$$TF = 1 + log(TF).$$

In the end, vector are normalized by using the norm "*L2*". About the model, authors provided two versions: one which uses the MLP for simulating the Logistic Regression and the second one the Logistic Regression provided by Scikit-Learn (Pedregosa et al., 2011). In the analysis we used only the letter one.

MLP The feature representation in this case is the same as the Logistic Regression. Here, what it changes, is the model architecture. As MLP, it is created by using three Dense Layer, each one with 50 units, and all of them uses the norm *L2* a kernel regularizer. Here, the activation function of these three layers is the *ReLu*. The last layer, which is the one for the classification, is a dense layer which contains two units (one per target class) and it is followed by the *Softmax* function. The optimizer used is *Adam* with the default settings, and as loss function *kullback laibler divergence*. The learning rate is 0.001, epochs 16, and batch size 200. The architecture of the model is designed using Keras (Chollet and François, 2015).

5.1.2 Logistic Regression over *T1* dataset

Devidson et al. (Davidson et al., 2017) present a Logistic Regression character based as model, over the Twitter dataset T1. Davidson provide the code for the experiments and also in this case the replication was an easy job.

The model represent data in a similar way of the MLP and Logistic Regression over the Wikidetox dataset. Also here, authors use the TfidfVectorizer, while in this case it is based on unigram, bigram and trigram over words and not characters. Here, the words are lowercased and also stemmed, by using the Porter Stemmer provided by NLTK (Bird and Klein, 2009). The filter of the n-grams is applied on the top 10000, which have a minimum frequency of 5 elements, and maximum distribution frequency lower than 0.75 (in this case, which is a float, the value represents a proportion of documents).

The furst step of the pipeline is a feature selection phase, which is made by the function *SelectFromModel* from the Scikit-Learn's library, which is defined as a meta-transformer for selecting features based on importance weights. The model for the feature selection is the Logistic Regression (Scikit function), where the penality used is the norm *l1* and the class weight is balanced, which mean that we try to balance the error based on the classes distribution (an error in a smaller class will have a greater error instead of the error of a class with higher frequency). The new feature space is after used by a Logistic Regression (Scikit function) with norm "l2".

The original model also include some features such as the sentiment of the tweets, by using sentiment analyzer designed for social media. This kind of extra features are not used in our replication because it allow us to be more free with scaling in another datasets and because we are interested in just the text as feature for the decisions that classifier will do. These kind of features can maybe help the classifier but for an accurate discussion we need to analyze in details the effect on the classifier.

In the paper, they evaluate the performances with a classification report (function of scikit learn) and a confusion matrix. As we can see from the comparison between tables 5.3 and 5.4 and the Figure 5.1 , the choice of using only the n-gram with TFIDF and not external features affected a bit the performances of the classifier, but in general they still the same.

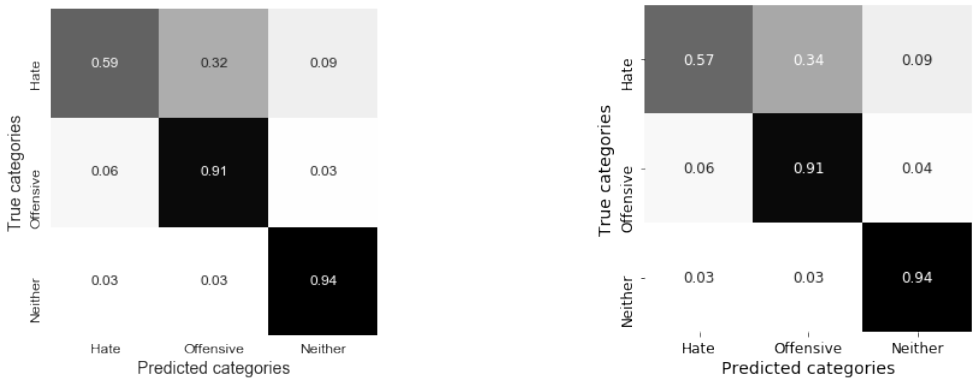
Table 5.4 shows that the majority of the errors came from the zone of hateful and offensive. Here, the possible explanation is derived from the fact that those classes

	precision	recall	f1-score	support
0	0.44	0.59	0.51	164
1	0.96	0.91	0.93	1905
2	0.83	0.94	0.88	410
avg / total	0.91	0.89	0.90	2479

TABLE 5.3: Original Classification Report

	precision	recall	f1-score	support
0	0.45	0.57	0.50	164
1	0.96	0.91	0.93	1905
2	0.82	0.94	0.88	410
avg / total	0.91	0.89	0.90	2479

TABLE 5.4: Replicated Classification Report



(A) Original (B) Replicated

FIGURE 5.1: Confusion Matrices

have a consistent number of relevant words in common (e.g. "bitch" was in both hateful and offensive top 10 words 4.2).

5.1.3 LSTM over *T2* dataset

Badjatija et al. (Badjatiya et al., 2017) try to solve the problem of the hate speech with the use of the deep learning . In their work, they experiment with different approaches, as summarized in Table 5.5. The replications in this case are not easy because the architectures are not described in detail in the paper and the *github* repository does not contain the whole code for obtaining the same results.

In the replication I decided to focus only in the LSTM techniques with the 2 variants of Random Embedding and Glove Embedding (with the addition of the GBDT), as a total of 4 models.

The difference between the random embedding and the Glove embeddings (Pennington, Socher, and Manning, 2014) is that, while in the first case the vectors which represent the words are initialized with random values, in the second approach the vectors are initialized with vectors obtained by a train of these vector in a huge corpus.

	Method	Prec	Recall	F1
Baselines	Char n-grams + LR	0.729	0.778	0.753
	TFIDF + Balanced SVM	0.816	0.816	0.816
	TFIDF + GBDT	0.819	0.807	0.813
	BoWV + Balanced SVM	0.791	0.7888	0.789
DNNs Only	CNN + RE	0.813	0.816	0.814
	CNN + GloVe	0.839	0.840	0.839
	Fast Text + RE	0.824	0.827	0.825
	Fast Text + Glove	0.828	0.831	0.829
	LSTM + RE	0.805	0.804	0.804
	LSTM + GloVe	0.807	0.809	0.808
DNN + GBDT	CNN + Glove + GBDT	0.864	0.864	0.864
	CNN + RR + GBDT	0.864	0.864	0.864
	Fast Text + Glove + RE + GBDT	0.853	0.854	0.853
	Fast Text + RE + GBDT	0.886	0.887	0.886
	LSTM + Glove + GBDT	0.849	0.848	0.848
	LSTM + RE + GBDT	0.930	0.930	0.930

TABLE 5.5: Deep Learning techniques applied. **RE** = Random Embedding

The CNN architectures were avoided because our last model is a CNN with GRU, and it allow us to have different architectures to compare between each other. About *Fast Text*, we decided to avoid it because it is just a library which handle text classification.

In the replications, while the LSTM + RE and LSTM + Glove gave us similar results, the addition of the GBDT didn't gave the same results as the authors. While in the second part with DNNs only, there is a use of a neural network with an embedding representation, the idea of the third part (DNN + GBDT) is to use the DNN for training the embedding and then use them as new feature space for the GBDT.

In the end, both LSTM with random embedding and glove are better of the version of the GBDT and, for the next replications, we decided to use as best model of the paper the LSTM with Random Embedding (f1 score obtained 0.806). The Glove result is the same as authors, but it is slower in the pre-processing for loading the embedding from a pre-build dictionary and the results are approximately equal to the random embedding one.

While in the previous model a sentence was represented by an array, here we have a matrix, where we have one row for each word in the sentence and the column correspond to the embedding which represent the word. One issue of this approach is given from the fact that we need to have all the input with the same size: for this reason we need to use the padding, which mean adding some noise, or remove some part of the sentence, in order to achieve the requirement. In this model, the padding is the length of the biggest sentence. This problem can be avoided by using the library *pytorch* (Paszke et al., 2017), which allow to use the sentences with different sizes; however authors uses Keras and we need to replicate these model faithfully. In the case of random embedding, we have that the dimension of the embedding is 200, and the values at the beginning are initialized with random numbers. This process is handle by the *Embedding* function provided by Keras (Chollet and François, 2015), which create a dictionary where each word has its own embedding. Here, the embedding are trainable, which means that during the backward phase, the embeddings are updated in order to minimize the error.

Dataset	Evaluation
T1	0.82
T2	0.94
T3	0.92

TABLE 5.6: Original evaluations with F1 score micro over the three dataset

The whole model is built using the Keras library, and it is build as following:

- Embedding(size = 200, length = maxpadding);
- Dropout(rate = 0.25);
- LSTM(units = 50);
- Dropout(rate = 0.50);
- Dense(units = 2);
- Activation(softmax).

The optimizer used is *adam* with the default settings, and the loss function the "*categorical cross entropy*". The model was trained with 4 epochs.

5.1.4 CNN + GRU over *T1**, *T2* and *T3* datasets

The second deep learning approach is given by Zhang et al. (Zhang Ziqi, 2018), where the idea is that the Convolution Neural Network combined with a Recurrent Neural Network is able to capture in a more effective way co-occurring word n-grams as useful patterns for the classification. The architecture in this case is well described so it is easy to replicate the results shown. This model is tested with the dataset T1*, T2 and T3.

The CNN + GRU is built entirely with Keras library (Chollet and François, 2015). The input is an embedding matrix where the padding is fixed to 100 words. The choice of this pre-setted number is given by the fact that the model should analyze tweets, and the size should be enough. In this case the model uses word embedding with 300 dimension pre-trained on the 3-billion-word Google News corpus. After the embedding later there is a Convolutional Neural Network with 100 filters and window size 100, and the layer is followed by a max pooling layer with a pool size of 4. The features generated are given as input to a GRU layer, which threats the feature dimension as timestep and outputs 100 hidden units per timestep. The output is flattened by a global max pooling layer, which take the maximum value for each timestep dimension. In the end, there is a *softmax* layer, which uses the elastic net regularization that linearly combine the norms L1 and L2. As loss function is used the categorical cross entropy and as optimizer Adam.

The model is shown in Figure 5.2.

5.1.5 Transfer Learning

Recently, *transfer learning* has been argued to improve the performances in the text classifications tasks "ULMFT". The approach used by the authors uses a pretrained Linguistic Model (LM), which is trained on a huge corpus; this LM is after optimized for the current task, by using the dataset, and the new LM is used as backbone for a

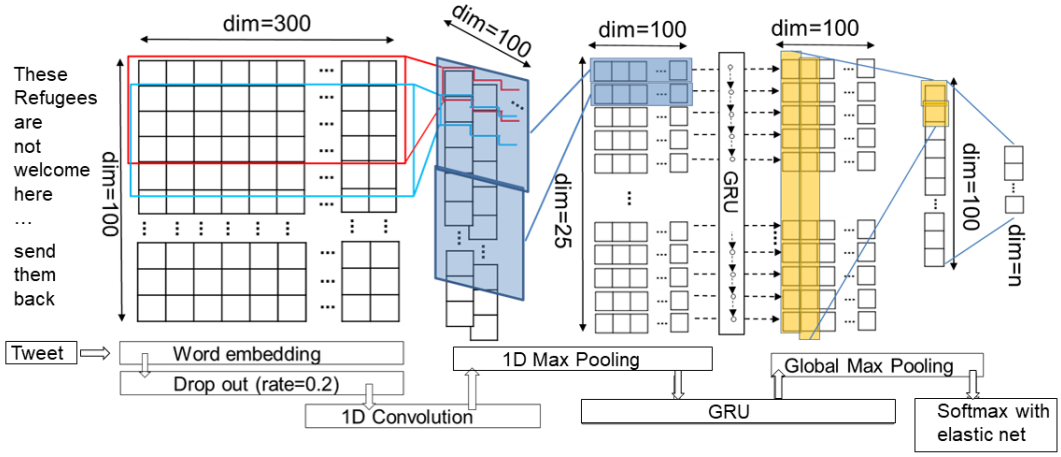


FIGURE 5.2: Architecture of CNN GRU

classifier, which is built on the top of that; Figure 5.3 summarize the structure of this model. The model consists in a regular LSTM. This LM is trained over WIkitex-103 (Merity et al., 2016) and it consists on 28595 preprocessed Wikipedia articles, for a total of 103 millions of words. At this point, the LM is trained, and the second phase is the *fine-tuning* for our target domain. This fine tuning uses two techniques:

1. discriminative fine-tuning: each layer uses different learning rates;
2. slanted triangular learning rates: for let that the model converge quickly, the model at the beginning increases its learning rate, for reaching an optimal region quickly, and after the learning rates will decrease after every epoch.

Finally, for the classification, the layer which produce the distribution of the words is truncated and substituted with the target layer (one output for each class). The weights that connect this layer to the network must to be learned; the *gradual unfreezing* technique is used, where for few epoch we train the entire classifier, where only the last layer's weights are trained, and after we allow the learning for few epochs also to the previous layer, and so on.

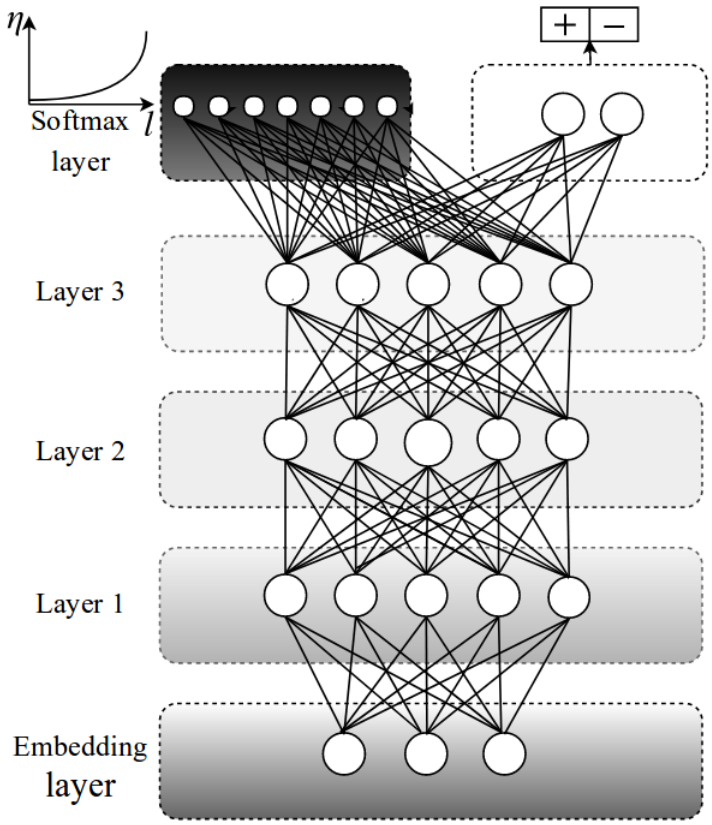


FIGURE 5.3: ULMFiT structure

5.2 Evaluations

It is the time for answering two questions:

- how well these models perform?
- how well these models scale?

We can answer to these question by analyzing the models in two scenario:

- replication and re-training;
- cross application between datasets.

In this section we present the analysis only for the two classes models, where for the dataset *T1* we combined the *offensive* class with the *neither*, and in dataset *T2* we combined the *racism* class with the *sexist* one. This choice allow an alignment between the datasets, where all of them contain two classes: hate speech and not hate speech.

5.2.1 Evaluation Metric

The chosen metric is the F1 score with the "macro" variant, provided by Scikit Learn (Pedregosa et al., 2011). This metrics allow us to understand how good is a model, by also considering of the problem of the unbalanced dataset. This is a problem, because we care about a good classification of hateful contents and we need to give importance to the error that the classifiers will make in this class.

	Accuracy	F1 micro	F1 weighted	F1 macro
W	0.883	0.883	0.828	0.469
T1	0.774	0.774	0.676	0.291
T1*	0.942	0.942	0.914	0.485
T2	0.682	0.682	0.554	0.406
T3	0.83	0.83	0.753	0.454

TABLE 5.7: Median Classifier Evaluation

Let's now think to a simple classifier which labels all the samples as part of the majority class. Its performances are summarized in Table 5.7. It is easy to see how metrics like accuracy doesn't take into account too much the problem of the unbalanced dataset. The F1 macro seems to be the only one to observe this problem and this is the reason why we choose this metrics for the following evaluations.

5.2.2 Replication and Re-Training

Every model that we have been presented was trained with a specific dataset, and the choice of some hyper-parameters were taken due to the nature of the dataset. Here, we try to retrain the models with different datasets and evaluate them, by using the same original hyper-parameters; this choice can affect the performances of the models over the new datasets. For allowing a comparison of the performances between the models, each datasets were splitted in training and test set in the same way for every training. This allow us to avoid penalizations due to the randomness.

Model	Dataset			
	W	T1*	T2	T3
LR char	**0.86**	0.63	0.82	0.85
MLP char	**0.86**	0.63	0.81	0.85
CNN+GRU	0.87	**0.70**	**0.83**	**0.81**
LSTM	0.85	0.64	**0.78**	0.79

TABLE 5.8: F1-scores (macro-averaged across classes) of the two-class models trained and evaluated on each dataset (datasets used in original papers in bold)

As we can see in Table 5.8, the results indicate that the models are roughly equally effective when applied to different texts, provided that they were trained using the same kind of text. Inferior performance on T1* by all models can be explained by two factors. First, the dataset is highly imbalanced, with the "hate" class taking up only 5% of the training set. Second, this dataset is derived from Davidson et al.'s original three-class corpus that separates *offensive speech* from hate speech. To constrain classification into only two categories, T1* assimilates the "offensive" and "non-hate" classes into one. Of these, the offensive class dominates, covering roughly 80% of the combined class in the training set. It may have more overlap with hate speech than non-offensive speech, making classification more challenging.

About the transfer learning, this architecture doesn't reach the baselines of other papers, where in the Twitter dataset results (F1 score macro) are 0.62 (T1), 0.75 (T2), 0.80 (T3), where the baselines are 0.66 (T1), 0.84 (T2), 0.86 (T3). We did not consider these models for the further analysis due to the complexity of the model and the huge demand of resources.

5.2.3 Cross Application Between Datasets

To estimate the adaptivity of models pre-trained with one dataset, we applied them to all test sets. We first trained each model with the training data used in the original paper presenting it, and then applied the resulting classifier to the test sets of all four datasets.

Model, training dataset	Dataset			
	W	**T1***	**T2**	**T3**
LR char, W	(0.86)	0.37	0.50	0.24
MLP char, W	(0.86)	0.38	0.50	0.25
CNN+GRU, T1*	0.11	(0.70)	0.48	0.51
CNN+GRU, T2	0.14	0.28	(0.83)	0.44
CNN+GRU, T3	0.13	0.48	0.50	(0.81)
LSTM, T2	0.23	0.33	(0.78)	0.47

TABLE 5.9: F1-scores (macro-averaged) of pre-trained two-class models applied to different test sets, with the original test set in parentheses

As we can observe from the results reported in Table 5.9, none of the pre-trained models transfer well to any other dataset. These results indicate that linguistic hate speech indicators are not well retained across different datasets. This may be due to the lack of words shared between datasets, or differences between the relevant features of particular subcategories of hate speech.

In Section 4.2.3 we discussed the similarity between the dataset, and we saw that the similarity between the datasets were high, where the datasets were sharing the common words. However, we showed that the vocabulary were quite different: this could be the problem, were the words that are important for the classification are not present in the dictionary of the classifier.

5.3 Swear Words Analysis

An important distinction can be drawn between hate speech and speech that uses offensive vocabulary without being hateful. The latter is separated from hate speech in Davidson et al.'s dataset T1 (Davidson et al., 2017). To our knowledge, this is the only dataset (and paper) that distinguishes these categories. Especially given the lack of any precise definition of hate speech in either legal or academic contexts, it is important to investigate the extent to which existing approaches are sensitive to the hateful-offensive distinction.

A particular concern is the extent to which two-class models assign offensive but non-hateful text to the "hate" class. This can be treated as a false positive, assuming the hateful-offensive distinction is appropriate. Of course, the assumption is not always clear: for instance, Google Perspective offers to detect "toxic" comments; but whether "toxicity" should include all offensive text remains a subjective assessment. Nevertheless, we maintain that there is a significant distinction between non-hateful offensive speech and hate speech, and categorizing the former as "hateful" is problematic. To estimate the performance of the models on offensive but non-hateful text, we applied the "offensive" data-points from T1's test set to all two-class models, except CNN-GRU trained on T1*, which is built from T1 and hence not independent of the task. As seen in Table 5.10, CNN+GRU trained on T3 was the only model

to succeed in this task. All other models performed on a random or below-random level.

Model, training dataset	Assignment to non-hate
LR char, W (Wulczyn, 2017)	0.40
MLP char, W (Wulczyn, 2017)	0.36
CNN+GRU, T2 (Zhang Ziqi, 2018)	0.23
CNN+GRU, T3 (Zhang Ziqi, 2018)	0.89
LSTM, T2 (Badjatiya et al., 2017)	0.36

TABLE 5.10: Performance of two-class models on offensive but non-hateful speech

Additional experimentation revealed that the seeming success of CNN+GRU trained on T3 was due to the prevalence of unknown words in the "offensive" test set, mapped to the unknown token (<*unk*>). This token was associated with the non-hateful class in the model, and on average over 40% of words per sentence in this test set were mapped to it. Hence, the performance simply reflected the model's small vocabulary.

The results are suggestive of the problematic and subjective nature of what should be considered "hateful" in particular contexts. As the labels have been manually gathered via crowd-sourcing, there is no guarantee that labels from different people were consistent. Within datasets, this problem has can be addressed to some extent by majority voting, but there is no method of guaranteeing agreement between datasets. It seems that the labeling in the two-class models represents offensiveness more than hatefulness as such.

Manual experimentation suggests similar conclusions concerning Google Perspective. Some examples of Perspective's scores on non-hateful sentences appended with a common English curse-word (marked with "F" here, but in original form in the actual experiment) are presented in Table 5.11.

Sentence → Modified sentence	Old → New score
You are great → You are F great	0.03 → 0.82
I love you → I F love you	0.02 → 0.77
I am so tired → I am F tired	0.06 → 0.85
Oh damn! → Oh F!	0.64 → 0.96
Food is amazing → Food is F amazing	0.02 → 0.68

TABLE 5.11: Google Perspective "toxicity" scores on non-hateful sentences with and without a curse word

None of the examples in Table 5.11 are hateful, but the curse-laden variants are all considered "toxic" with a high likelihood. Clearly "toxicity", as Perspective currently classifies it, is not assimilable to hate speech in any substantive (or legal) sense.

5.4 Analysis of the Behaviour

In this section I will try to analyze in details how the models work and we will try to understand why samples are classified in a specific way.

The analysis are done by looking the effect in the classification for every component of the sentence (ngrams for the TF-IDF algorithms, words for the embeddings). Clearly, here it is not consider the effect of the combinations of the elements, but we can just have an overview and an idea of what the problems are.

During this process I will do some considerations about the sentences that for someone else can be wrong (due to the difficult nature of the hate speech identification).

Also, we will see some examples of the *offensive* classification, by trying to understand the results of the models against the offensive class of *T1*.

At first we analyze the confusion matrices for the models and let's try to understand where the errors occur, which are summarized in Figure 5.4. In *W*, the models seem to work in the same way; here, both models perform very well in the *non toxic* classification, while the problem is in the huge amount of error on the *toxic* class, which is also the minority class in the dataset. As we can see, the models in general mis-classify toxic comments, by preferring the labelling to the non toxic class. Now, let's see the LR Word over *T1*: few mistakes are done between the *neither* class and the *hate* and *offensive* classes, which mean that it is not a difficult task to separate these two groups. The major error, as in the previous case is given by the hate sentences which are classified as offensive, which is the majority class (this behaviour is true for all the models, where the models assign more the bigger class). In all the models the non hateful classes are easy to recognize, while the hateful have a lot of mistakes. The CNN GRU over *T1**, shows that only the 31% of hate sentences are correctly classified (1 out of 3), showing and highlighting that the problem between the hate class and the swear word must to be take carefully and considered for the analysis. As discuss in Chapter 3, we cannot let that the models' judgments are biased by the swear words.

LR char over *W*

Sentence 1

You are a child

This sentence should be classified as non toxic, while is classified as toxic with a strong probability: 0.92. Here it seems that there are not elements that make the sentence toxic. By looking through the ngrams of the sentence, it seems that the ngram"*you a*" is toxic and it could affect the prediction.

Sentence 2

== I HAVE A HARD ON FOR WAR == SUP, NIGGS????????????

Classified as non toxic, it is toxic (0.87). The toxicity, in my opinion, is given by the term *niggs*. Unfortunatly, non of the ngrams that involves this word are recognized as toxic.

Sentence 3

Dylan's a pedophile

Classified as non toxic, it is toxic (0.89). The toxic ngrams are "s a p", "pe", not enough considering that there are 61 ngramns involved. None of these words alone are toxic, but the combination of those are toxic. In my opinion here the problem

is due to the representation, where the TF-IDF represent a sentence by loosing the relationships between words.

Sentence 4

== Cunnywafts == Nothing but a group of mediocre cunnywafts

True label: 0; predicted : 1 (0.51). This is a case of predictions on the border. Here, the models considers 147 ngrams. The toxic ngrams are "dio", "cun", where the first one is super toxic (probability equal to one of being toxic). Probably, the high ratio of non toxic decrease the final probability.

Offensive Sentence 1

is that ya bitch

Toxicity at 0.93 as probability. The reason is that there are the ngrams of "you" and "bitch" that are considered as toxic, where the latter one has a huge toxicity value. Here, the use of a swearword conditioned a lot the classifier.

Offensive Sentence 2

cant you see these hoes wont change

This sentence is correctly predicted as non toxic (0.95). The question is: is the classifier that is working properly or is just that he doesn't recognize the swearword "hoes". In the ngrams lists, we do not have any "hoe" or "hoes" ngram. Moreover, in the training set these words appears in 8 sentences, a number which is too small for having appearing in the top 10000 ngrams. For this reason, this classification is correct just because of wrong reasons: if we use this swearwords for offending someone, this should be toxic. If, for example, given the following:

You are a hoe

which clearly is hateful, the classifier classify it as non toxic with the probability of the 83%. The occurrences with the words "hoe" or "hoes" is more than 3000, where in 2884 cases it was classified as non toxic; this raise the problem that a lot of examples in the offensive list of *T1* are well classified due to a lack of information in the training set. This behaviour could occur also with other swearwords.

5.4.1 MLP char over *W*

This model shares the dataset with the previous one, which means that the common words / ngrams are the same, Let's see if the errors are comparable with the previous case.

Sentence 1

You are a child

This sentence is mis-classified as in the previous model (toxic instead of non toxic). This denote a similarity in the behaviour between the logistic regression and the Multilayer Perceptron with 3 layers. The sentence *you are a* is classified as toxic with a probability of 0.92.

Sentence 2

Dylan's a pedophile

Classified as non toxic with a probability of 0.75, it is toxic. Also for this sentence, as for the previous model, the problem is given by the ngrams which cannot capture the toxicity of this sentence. This also show another proof of the similarity of these two models.

If we compare the predictions of LR char and MLP char over the same test set of *W* we can see that the 98.85% of the predictions are the same, which means that the models works practically in the same way. This model also shows that sentences such as *you are a hoe* are mis-classified, and it has the same problem with the offensive words: the sentence *this CD is fucking amazing* is toxic (0.97) and the decision is biased by the swear word "fuck".

5.4.2 LR word over *T1*

This is the special model with the 3 classes. It is curious now to see if and how the offensive classes are handled.

Sentence 1

fucking queer

Hate: 0.7 - Offensive: 0.3 - Neither: 0

The correct answer is 1 (offensive). Honestly, in my opinion, this is an example of hate speech because of the meaning of "queer". It is easier to analyze this model because it uses the words as representation in the ngrams; while the word "fucking" alone is consider offensive (correct, it is a taboo word), "queer" is considered hateful (0.58), while the combination of both generates a probability of being hateful equal to 0.7.

Sentence 2

God damn birds

Hate: 0.1 - Offensive: 0.22 - Neither: 0.78

The true label is offensive. Here, the word "damn" (alone) is classified as offensive, but the words "God" and "birds" are not. Probably, here it is a problem of the training set, where the ngram "God damn" should be more frequent for being classified as offensive.

Sentence 3

But rich is a light skin coon. So it's expected

Hate: 0.5 - Offensive: 0.15 - Neither: 0.35

The real label is neither. Here, the classifier sees the word "coon", which is a very offensive word for black people.

Offensive Sentence 1

this CD is fucking amazing

This sentence is correctly classify as offensive.

5.4.3 LSTM RE *T2*

Sentence 1

yes. yes please. take my money

It is classified as non hateful, while it is. Here, a lot of mis-classification are difficult to understand because they refer to facts such as TV-shows and with out the context are difficult (e.g. a lot of #mkr hashtag).

Offensive Sentence 1

this CD is fucking amazing

Non Hateful: 1 - Hateful: 0
This sentence is perfectly classified as non hate speech.

Offensive Sentence 2

God damn

Non Hateful: 0.7 - Hateful: 0.3
Also this sentence is correctly classified.

Offensive Sentence 3

You ever fuck a bitch and she start to cry? You be confused as shit

Non Hateful: 0.14 - Hateful: 0.86
This sentence which contains a lot of slang is classified as hateful "bitch".

5.4.4 CNN GRU *T1'*

Here I just consider the case of the model trained over the dataset *T1'* because it is more interesting to see how the model handle the differences between offensive and hateful sentences.

Sentence 1

fucking queer

Hateful:0.68 - Non Hateful: 0.32 Here we have the same problem as in the LR word model. The same considerations here are done.

Sentence 2

it's your man make me white trash cousin fucker

Hateful:0.65 - Non Hateful: 0.35 This should be non hateful but just offensive. Here, the slang and the use of swear words make the sentence hateful for the classifier.

Offensive Sentence 1

this CD is fucking amazing

Hateful:0.15 - Non Hateful: 0.85 It seems that the use of swear word is well classified thanks to the non hate class which contain also the offensive class.

5.5 Augmentation

Due to the limit amount of examples available for each dataset, we try to apply a couple of augmentation technique:

- Google Translate;
- Synonimous Replacement.

In this case, the models are retrained by using the default hyper-parameters. However, none of these techniques gave us successful results. The idea behind these approaches is to generate sentences with the same meaning but different structure.

Google Translate This technique uses the Google API in order to translate sentences. The idea is, given a sentence s_i, which is written in English, we translate it in another language, such as Spanish, and we re-translate it in English. If the sentence is different from the original one, we insert it in the augmented dataset, otherwise we just discard it. For example, given the following sentence:

Hi buddy,

the Spanish translation is:

Hola Amigo,

and, if we come back to English, the result will be:

Hello friend.

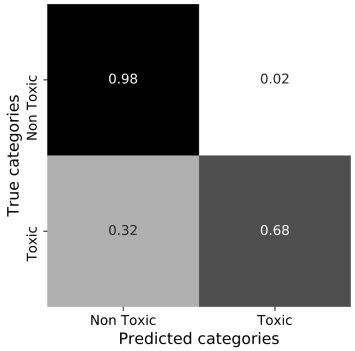
The two sentences have the same meaning, but they uses different words.

Synonymous Replacement The algorithm is to replace some words with theirs synonymous in the sentences. Given a sentence s, first the algorithm will split it by using the *word_tokenize* funcion, provided by NLTK, which create the tokens of the sentence. This list of words are processed, and, by using the *wordnet* funcion (NLTK), we select randomly some words and we replace those with the synonymous contained in the wordnet. For example, given:

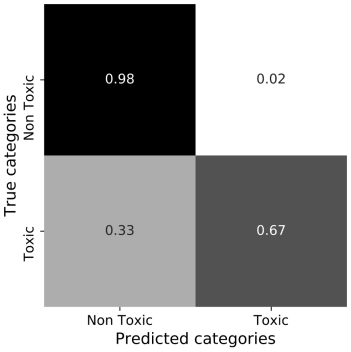
The cat is on the table,

the algorithm will produce:

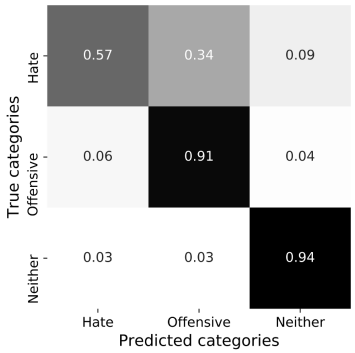
the kitten is on the table.



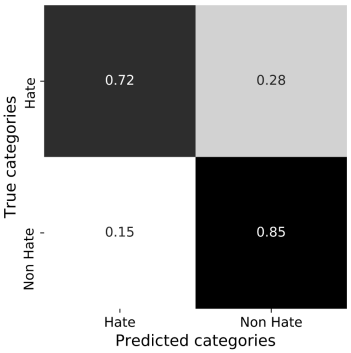
(A) LR char W



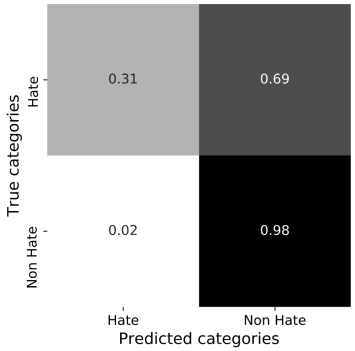
(B) MLP char W



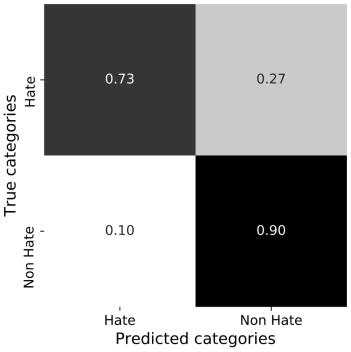
(C) LR word T1



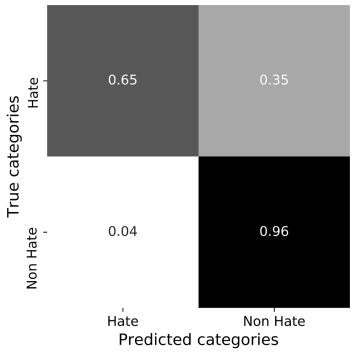
(D) LSTM RE T2



(E) CNN GRU T1*



(F) CNN GRU T2



(G) CNN GRU T3

FIGURE 5.4: Confusion Matrices of the models

Chapter 6

Attacks

The goal of the adversary which is considered in this work is to fool a detection model into classifying hate speech input as ordinary speech. The assumption that we made is that the adversary has complete control of the input and he/she can modify it in order to evade detection, while retaining the semantic content of the original sentence hate speech. Another assumption is that the adversary has not white-box access to the model parameters. The adversary model is relaxed in one attack, where some knowledge of the training set is required. Classifiers that rely on surface-level features can be attacked with malformed input, constituting *evasion attacks* (Biggio et al., 2013). In this Chapter we describe six evasion attacks against the seven classifiers that we replicated (Section 6.1), all attacks are based around altering input text with easily implementable and automatic methods. We categorize the attacks into three types, and experiment with two alternatives from each:

- ***Word changes***
  - Insertion typos;
  - Leetspeak;
- ***Word boundary changes***
  - Inserting whitespace;
  - Removing whitespace;
- ***Word appending***
  - Appending common words.
  - Appending non hateful words.

We differentiate the attack types, because they target distinct (albeit sometimes overlapping) aspects of the classifier. Word changes are done to change the identities of words. For word-level models, the new identities are likely to become "unknown" (i.e. non-identified tokens denoted by "*unk*"), while retaining the readability and semantic content of the original text to a maximal extent from a human reader's perspective. Word boundary changes, on the other hand, not only alter word identities but also the sentence structure from a superficial perspective. Finally, word appending makes no alterations to the original text, but only adds unrelated material to confuse classification. There are theoretical reasons to believe certain attacks to be more effective against some classifiers than others. In particular, as word tokenization plays no role in character-models, it is clear that these are less susceptible to word-boundary changes than word-models. Furthermore, character-models are expected to be more resistant to word changes, as many character n-grams are still retained after the transformations are applied. In contrast, all word-based models

are expected to be vulnerable to attacks that alter word identities (i.e. both word changes and word-boundary changes). Word appending attacks, in contrast, have no a priori reason to work better against character- than word-based models, or vice versa. We applied the attacks to all seven classifiers, by poisoning the hate class samples from the respective test sets. We analyze also some counter measures for the attacks 6.2. The results are shown in Section 6.3. Based on the obtained results, we combined the most powerful attacks, and we present the "Love Attack" (Section 6.4). We will conclude the chapter with some considerations about the trade-off between the performance and security aspect (Section 6.5).

6.1 Adversary models

In this section we will explain in more details the algorithms involved in the attacks.

6.1.1 Word Changes

Word change attacks introduce misspellings or alternative spellings into words. They may make words entirely unrecognizable to word-based models, and change the distribution of characters in character-based classification. The aim of these two attacks is to make unrecognizable the words which are modified in such a way that a human being can still recognize the meaning of those, while the classifier not.

Inserting Typos

In 2017, Hosseini et al. (Hosseini et al., 2017) showed that Google's "toxicity" indicator Perspective could be deceived by typos. However, Perspective has since been updated, and the examples the authors provided no longer succeed to the extent they did then. We review these results in ***Section 6.3***. Still, we consider the proposal to be worth further investigation with larger datasets and automatic typo-generation.

Algorithm As an attack, typo generation has three desired properties: (i) reducing the detection likelihood of hate speech, (ii) avoiding correction by automatic spell checkers, and (iii) retaining the readability and meaning of the original text. If the second goal is not met, the defender can include a spell-checker as a pre-processing stage in the classification. Satisfying the third goal requires the word to remain recognizable, and not to be mapped to some other word in the reader's mental lexicon.

To make the attack successful, word-level changes must not only fool the classifier, but be human-readable and retain the original interpretation to a maximal extent. This means that we cannot simply introduce typos via random changes to words. Instead, the change must have minimal impact on readability.

We used the empirical finding from cognitive psychology that characters in the middle of words have a smaller effect on readability than the first and last characters (Rayner et al., 2006). To maximize understandability, we restricted the alterations to a single switch between two characters. The algorithm switches the order of two characters in the word, excluding the first and final character. The probability of choosing particular characters is calculated by two factors: characters closer to the middle of the word are preferred, and characters that are close to each other are preferred. So, first, a character is chosen between the second and second-to-last character of the word, based on a Gaussian distribution centering in the middle character (hence, only words with four of more characters are applicable).After having a set o

values, one for each character, the set is normalized, in a way that the sum is equal to one (each element is divided by the sum of the set): in this way we have a set of probabilities. Next, a second character is chosen based on the combined effect of the first distribution and a second Gaussian distribution centered in the first selected character. These characters are then switched to create the typo. The Gaussian distribution is defined as $N(0, 0.5)$, where the value of sigma is obtained by experimental tests. Every character of the word (not the extreme) is mapped into a real values in the range [-1; 1] and the probability is obtained by the value of the Gaussian in that point.

Example of execution Let's say that we want to apply the algorithm to the word "*swords*". First, the algorithm excludes the border characters, so the word will be *word*. The first thing is mapping the characters into the x axis:

$$[-1, -0.33, 0.33, 1].$$

Now, let's take the value of the Gaussian $N(0, 0.5)$ in each point:

$$[0.11, 0.64, 0.64, 0.11];$$

normalized:

$$[0.07, 0.43, 0.43, 0.07],$$

and after the cumulative sum:

$$[0.07, 0.50, 0.93, 1.0].$$

For picking the character, the algorithm generates a sample from the *uniform distribution* $U(0,1)$. The character choose will be the one which range contain the extracted sample in the cumulative sum. Suppose that the extracted sample from the uniform distribution is 0.65, which correspond to the letter "*r*". It's time to choose the second character, and the Gaussian it will be defined as $N(0.33, 0.5)$, where 0.33 correspond to the value of "*r*" in the x axis. The second distribution is extracted in the same way as in the first part, where each point will have the correspondent value of the Gaussian curve:

$$[0.02, 0.33, 0.80, 0.32].$$

As described in the previous paragraph, we want to now to pick a new character which is close to the selected one, and having a greater probability for those characters which are close to the center. So, let's multiply the new distribution with the previous one:

$$[0.11, 0.64, 0.64, 0.11] * [0.02, 0.33, 0.80, 0.32] = [0.001, 0.14, 0.3412, 0.02].$$

We want to avoid to extract the same character, so it's values will be set to 0:

$$[0.001, 0.14, 0, 0.02],$$

and the vector normalized:

$$[0.01, 0.84, 0, 0.14].$$

At this point, the algorithm calculate the cumulative sum of the vector, extract a sample from the Uniform distribution $U(0,1)$ and the correspondent character will be switched with the previous one.

Leetspeak

In addition to typos, we also consider a simplified variety of leetspeak , which is a type of Internet slang replacing characters with numbers. (Different variants may involve more alterations, but we only consider character-to-number alterations here). Leetspeak has been shown to be easily readable to humans (Perea, Duñabeitia, and Carreiras, 2008), but will be unrecognizable to word-models unless it is also present in the training set.

Algorithm The leetspeak transformation algorithm is deterministic and it mapped some characters into numbers or other symbols. In this algorithm version, the mapping is defined as following:

$$(a:4),(e:3),(l:1),(o:0),(s:5).$$

The changes retain readability well, given the visual similarity between the original and leet characters (Perea, Duñabeitia, and Carreiras, 2008). Clearly, this attack is deterministic, which means that it could be esily mitigate in the preprocessing phase by substituting these kind of transformations in the words.

6.1.2 Word Boundary Changes

The tokenization attack differs from word transformation by retaining word-internal characters, but introducing or removing characters that result in a word-based model separating between different tokens. We use space as the most evident choice.

Algorithms We implemented two simple algorithms for introducing or removing whitespace. We predict removal to be more effective against word-based models on theoretical grounds. Character-based models are likely not highly susceptible to either variant.

- ***Inserting Whitespace***: Word-based models rely on tokenizing the text into a sequence of words, based on characters treated as word-separators. Therefore, a simple way to make words unrecognizable is to change the tokenization by introducing additional separators between words. The effect of this attack on readability is small, but it results in most words becoming unrecognized by the classifier. We used a simple approach of splitting each (content-)word into two by randomly selecting a character from the word and adding a space before it. In a word-based model a previously recognized word will turn into <unk> <unk>;
- ***Removing Whitespace***: Conversely, removing all spaces leaves a single <unk> datapoint. Word-based models' performance will then depend entirely on how this single token is classified. Character-models, in contrast, will only lose the information related to the space token, which might deteriorate their performance, but likely not much.

6.1.3 Word Appending

All text classification is based on the prevalence of class-indicating properties in the data. Therefore, adding material that is indicative of one class oven another makes it more likely for the classifier to assign the sample to that class. In many tasks this

is appropriate, but hate speech detection constitutes an important exception. Hate speech and non-hate speech are not reversible: adding hateful material to ordinary speech will turn the text into hate speech, whereas adding non-hateful material to hate speech may not change its status. This vulnerability invites a possible attack, where non-hateful material is inserted to hate speech to change the classification. For our attack model, we assume that the adversary is aware of the use of additional unrelated words to distract automatic classifiers. Assuming also that the beginning of the additional material is clear enough from the discourse, readability and semantic retainment are secured.

Algorithms We generated a random number (between 10 and 50) or words at the end of each text in the "hate" class of the test set. The bounds of 10 - 50 are given by experimental tests. The words were randomly chosen from two possible sources, yielding two varieties of the attack.

- ***Appending common English words***: Google has provided a list containing the most common words appearing in a large book corpus (Michel et al., 2011). We used random words from the top 1000 of these words, excluding "*stopwords*". The rationale behind this attack is that it requires no knowledge of the training data on the part of the attacker. Further, the common English words are likely to be contained in the training corpus of many different datasets;
- ***Appending common "non hate" words***: Here, we assume the attacker knows, or correctly guesses, a list of words in the training data's non-hateful class. He then appends the text with randomly chosen common (content) words from this class. We are still not assuming white-box access to the model itself.

6.2 Mitigation

The aim of this section is to study if there are techniques which can easily mitigate the effect of a specific attack.

Word Changes We tried two methods of improving classifier performance on test data modified via word changes: *adversarial training*, and adding a *spell checker* to test set pre-processing. In the adversarial training we augmented the training set with stochastically transformed versions of the original training examples, for all classes (doubling the training set size). The purpose of adversarial training was to add transformed varieties of each word into the model's vocabulary, making it more likely for the model to associate them with the correct class. The random nature of the typo-algorithm limits the scalability of the approach with long words, as the range of possible typos becomes larger. In contrast, given the deterministic nature of our leetspeak algorithm, we can expect adversarial training to work well against it. About the spell checker, we added a spell-checker to the pre-processing stage of the test set to find out how resistant our typo-introduction algorithm was to automatic correction. We used Python's *Autocorrect* library for this.

Word boundary changes For this attack we tried only with the adversarial training. For the whitespace insertion attack, we appended the training set with randomly split versions of the original data to include word-parts into the model's vocabulary (thus doubling the training set). Given that an n character word can be split

in two in $n-1$ possible ways, going through all possible combinations of splits in a sentence quickly results in a combinatory explosion. Hence, the effect of adversarial training is expected to be limited on analytic grounds. For completeness, we also conducted adversarial training via space removal, although this is close to useless on the grounds that it only adds whole comments as single words to the model vocabulary, and associates them with one class. As this method can only have an effect if it encounters the exact comment again, it cannot scale and hence its relevance is close to none in word-based models. Character-models, in contrast, can make use of it, but its inclusion is mostly redundant, as the data points are not changed much in comparison to the originals. ***Removing spaces***: Another mitigation method is available only for character-based models, where spaces are removed from both training and test data at the pre-processing stage. This method will by necessity remove any effect of adding spaces, but will also remove potentially relevant information from the data, as one common character is removed entirely. The effectiveness of this mitigation method thus depends on the relevance of the space character for the classification.

Word Appending The adversarial training in this case is similar for both attack, where we just add noise in all the sample of the dataset: for each sample, we add a copy of it and add from 10 to 50 random words, which are picked from the dictionary (Google common words for the first attack, non hateful words from the second one). The dataset also in this case will be doubled.

6.3 Results

We performed our six attacks on each of the seven model-dataset combinations replicated from prior work. This yields 42 attacks in total. We used adversarial training to mitigate all attacks, and tried additional defense methods for the typo attack (spell-checking) and tokenization attacks (space removal in training; this method is only available for character-models). Attack effectiveness varied between models and datasets, but the performance of all seven hate speech classifiers was significantly decreased by most attacks. Word-based models were most affected by tokenization changes, and character-based models by word appending. A significant difference between word- and character-based models was that the former were all completely broken by at least one attack, whereas the latter were never completely broken. The two character-models performed comparably across all attacks. The training dataset had a major influence on attack resilience, as demonstrated by the differences between the CNN+GRU model trained with T2 and T3, respectively. However, CNN+GRU trained on T2 was more resilient to all attacks than LSTM trained on T2, indicating that model choice also had an effect. The results from all attacks and their mitigation are presented in tables 6.1,6.2 and 6.3. Since our attacks affect data points in the "hate" class, we only report the F1-scores from this class.

We additionally experimented with Google Perspective, using example sentences from Hosseini et al. (Hosseini et al., 2017), reproduced below.

- **S1**: Climate change is happening and it's not changing in our favor. If you think differently you're an idiot;
- **S2**: They're stupid, it's getting warmer, we should enjoy it while it lasts;
- **S3**: They are liberal idiots who are uneducated;

- **S4**: idiots. backward thinking people. nationalists. not accepting facts. susceptible to lies;
- **S5**: They are stupid and ignorant with no class;
- **S6**: It's stupid and wrong;
- **S7**: If they voted for Hilary they are idiots;
- **S8**: Anyone who voted for Trump is a moron;
- **S9**: Screw you trump supporters.

Hosseini et al. provide examples of introducing manual typos, punctuation and spaces in each sentence, and demonstrate its effects on Google Perspective. We applied the same modifications to the same example sentences, but received much higher toxicity scores. This indicates that Perspective has been updated, and now performs better on adversarial data.

Word changes Word-models were more susceptible to leetspeak than typos, whereas no clear difference can be found in character-models. In addition, word-models were much more vulnerable to both attacks than character-models. Adversarial training had a major positive effect on performance against both attacks, but its effect was larger on the leetspeak attack. This is unsurprising given the deterministic nature of the leetspeak algorithm. The determinacy also indicates that the leetspeak attack could easily be mitigated by a counter-algorithm transforming numbers into corresponding characters.

Models	**Dataset**	**Baseline**	**Typos**			**Leet**	
			A	AT	SC	A	AT
LR char	W	0.75	0.60	0.71	0.68	0.61	0.74
MLP char	W	0.75	0.55	0.71	0.68	0.59	0.73
CNN + GRU	T1*	0.43	0.31	0.35	0.36	0.00	0.33
CNN + GRU	T2	0.76	0.27	0.67	0.68	0.09	0.77
CNN + GRU	T3	0.69	0.23	0.61	0.43	0.03	0.76
LSTM	T2	0.70	0.40	0.66	0.67	0.19	0.71
LR WORD	T1	0.50	0.30	0.42	0.37	0.04	0.48

TABLE 6.1: **Word Changes**: F1-scores on the "hate" class from attacks and mitigation.**A**: Attacks, **AT**: Adversarial Training, **SC**: Spell Checker

Word boundary changes Neither character-model was affected by the whitespace insertion attack, but the performance of both was markedly decreased by whitespace removal. We suggest this may be due to the fact that whitespace is involved in the beginnings and ends of words. Unlike adding whitespace, removing it abolishes all n-grams concerning word boundaries, which may be especially relevant for classification. All word-models were completely broken by white space removal, and severely hindered by whitespace addition. As expected, adversarial training had no impact on whitespace removal. Resistance to whitespace addition, in contrast, was improved, and reached levels close to the baseline, differing from it only 6 - 10%. Overall, removing whitespace was much more effective than its addition

for all models, both as an attack and in avoiding adversarial training as mitigation. Character-models performed slightly worse when trained with- out spaces, but not much, the largest drop in F1-score being 3%. We conclude that removing spaces in pre-processing during both training and testing makes character-based models resistant to tokenization attacks with only a minor reduction in predictive power. No comparable mitigation exists for word-models, where word boundary removal will force the text to be tokenized as a single *unk*.

Models	**Dataset**	**Baseline**	**Insert**			**Remove**		
			A	AT	RW	A	AT	RW
LR char	W	0.75	0.75	0.71	0.74	0.54	0.75	0.74
MLP char	W	0.75	0.75	0.71	0.72	0.56	0.76	0.72
CNN + GRU	T1*	0.43	0.04	0.34	-	0.00	0.00	-
CNN + GRU	T2	0.76	0.43	0.66	-	0.00	0.00	-
CNN + GRU	T3	0.69	0.08	0.63	-	0.00	0.00	-
LSTM	T2	0.70	0.42	0.64	-	0.00	0.02	-
LR WORD	T1	0.50	0.18	0.44	-	0.01	0.02	-

TABLE 6.2: **Boundary Changes**: F1-scores on the ""hate class from attacks and mitigation.**A**: Attacks, **AT**: Adversarial Training, **RW**: Remove whitespace

Word appending Unlike other attacks, word appending affected character- and word- models comparably. Words from the non-hate class of the training set had a systematically larger impact than common English words, but the difference was very minor (1 - 2%) on character-models. The largest difference was observed on the LSTM model, where non-hate words had almost twice the effect of common words (0.27 vs. 0.15). The only model not affected by either appending attack was the three-class word-based LR model from Davidson et al. (Davidson et al., 2017), trained on T1. We attribute this result to the fact that the major non-hate class of this dataset was the "offensive" class. Common English words or words from the "neither" class rarely indicate offensive speech, making it unlikely for the hate speech to be classified as such. This data imbalance also likely explains the negative effect of adversarial training, which was not observed on any other model. Adversarial training worked very well on all two-class models, reaching predictive power close to the baseline. The effect was the smallest with CNN+GRU trained on T1*, leaving 16% behind the baseline with adversarial training. The dataset T1* is drawn from T1 by combining offensive and non-offensive ordinary speech into a single class. As offensive speech takes the overwhelming majority of T1, T1* is highly imbalanced. We therefore expect adversarial training to result in the appended words to associate with the "non- hate" class more readily than the "hate" class, which would account for its limited success in mitigation.

Models	Dataset	Baseline	Common		Non Hate	
			A	AT	A	AT
LR char	W	0.75	0.48	0.68	0.47	0.67
MLP char	W	0.75	0.50	0.72	0.48	0.67
CNN + GRU	T1*	0.43	0.04	0.38	0.01	0.27
CNN + GRU	T2	0.76	0.64	0.75	0.50	0.74
CNN + GRU	T3	0.69	0.18	0.70	0.14	0.64
LSTM	T2	0.70	0.27	0.68	0.15	0.69
LR WORD	T1	0.50	0.48	0.44	0.45	0.30

TABLE 6.3: **Word Appending**: F1-scores on the "hate" class from attacks and mitigation.**A**: Attacks, **AT**: Adversarial Training

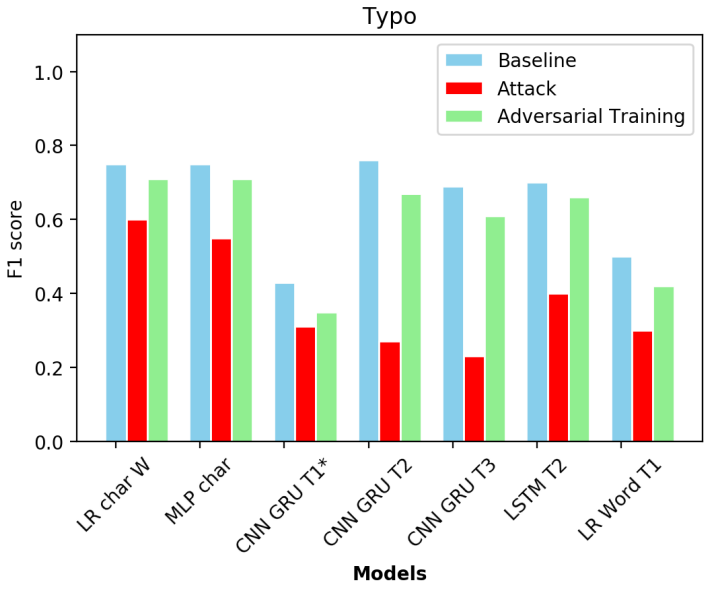


FIGURE 6.1: Typos Attack and Adversarial Training

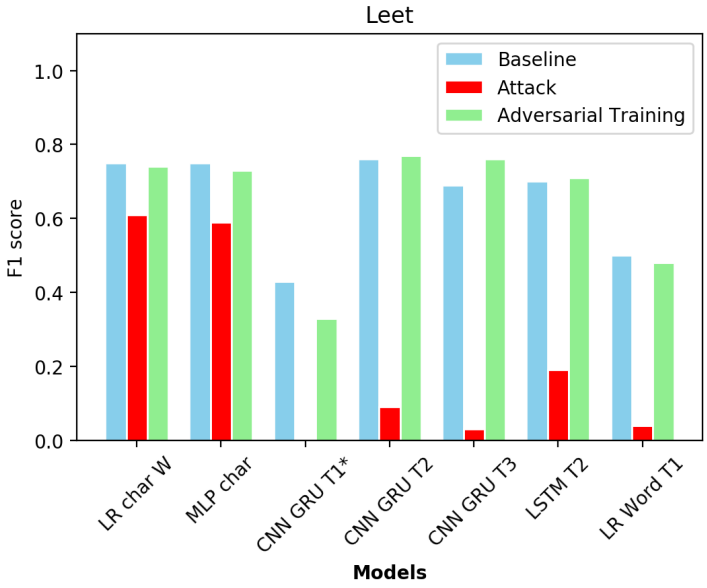


FIGURE 6.2: Leet Attack and Adversarial Training

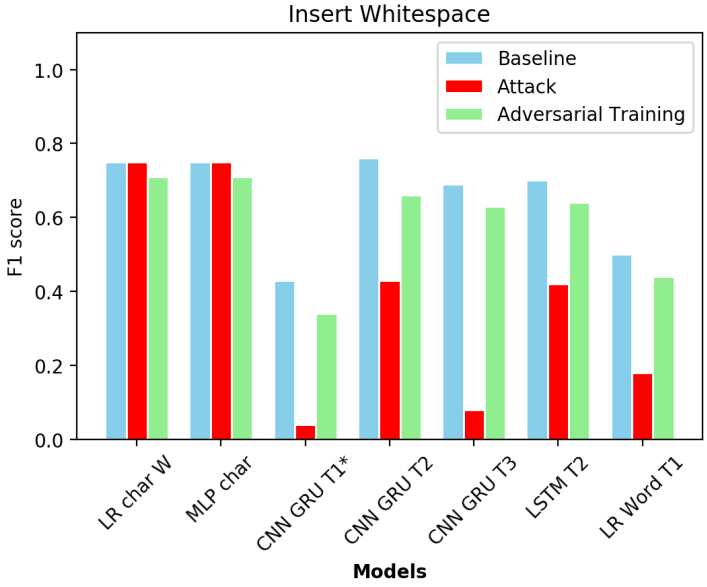


FIGURE 6.3: Insert Whitespaces Attack and Adversarial Training

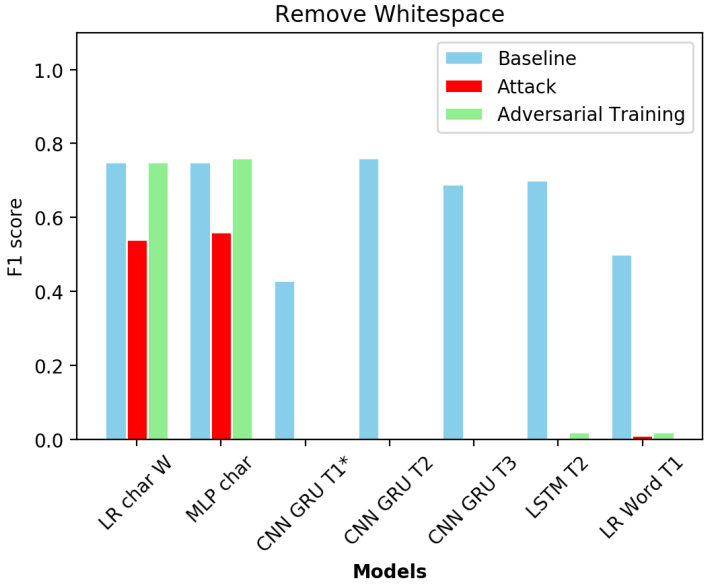


FIGURE 6.4: Remove White spaces Attack and Adversarial Training

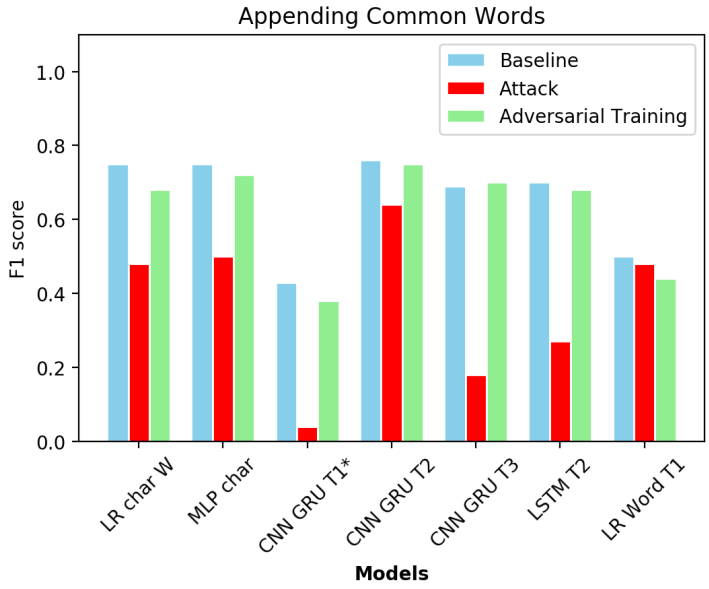


FIGURE 6.5: Insertion of Common Words Attack and Adversarial Training

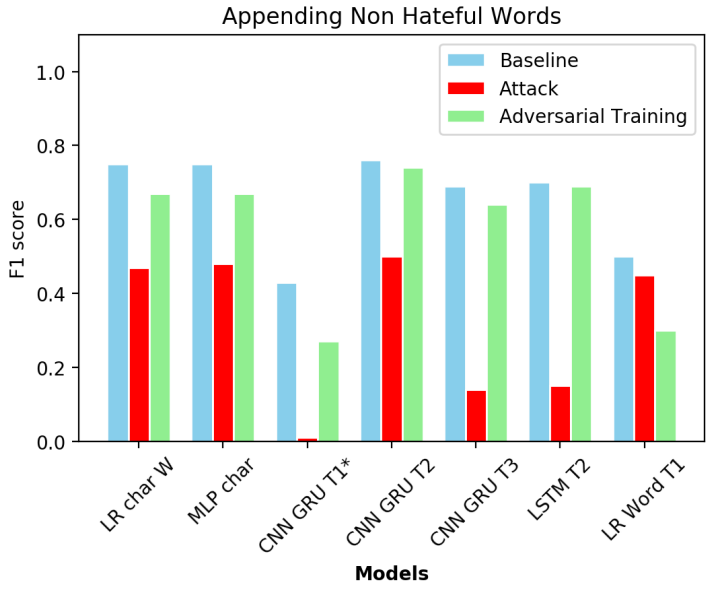


FIGURE 6.6: Insertion of Non Hateful Words Attack and Adversarial Training

6.4 Love Attack

Finally, we present the results from our attack combining the two most powerful approaches we experimented with: space removal and word appending. Furthermore, instead of using a long list of common or non-hateful words, we minimize the hindrance on read- ability by appending the text with only one word: "love". We choose this word because, intuitively, it is likely to negatively correlate with hate speech. Our results support this hypothesis. As expected, the attack completely broke all word-models, and significantly hindered character-models. Based on the word appending results, the performance on character-models would decrease with more appended non-hateful words. Since the original message can relatively easily be recovered, and the single unrelated word has a minimal effect on readability (assuming it to be separable from the rest by the reader), we consider this attack highly successful against state-of-the-art hate speech classification.

Models	Dataset	Baseline	Love
LR char	W	0.75	0.52
MLP char	W	0.75	0.55
CNN + GRU	T1*	0.43	0.01
CNN + GRU	T2	0.76	0.00
CNN + GRU	T3	0.69	0.00
LSTM	T2	0.70	0.00
LR WORD	T1	0.50	0.00

TABLE 6.4: **Love Attack**: F1-scores on the "hate" class from attack.

Sentence	Original	Common	Space	No Space	Typo	Leet	Love
s1	0.95	0.83	0.77	0.46	0.9	/	0.35
s2	0.92	0.77	0.76	0.51	0.77	0.77	0.37
s3	0.98	0.78	0.64	0.51	0.77	0.86	0.37
s4	0.95	0.79	0.89	0.51	0.8	0.77	0.37
s5	0.97	0.91	0.82	0.51	0.51	0.73	0.37
s6	0.88	0.74	0.56	0.51	0.29	0.78	0.35
s7	0.99	0.82	0.51	0.51	0.88	0.83	0.15
s8	0.96	0.7	0.39	0.39	0.7	0.39	0.35
s9	0.9	0.42	0.56	0.56	0.9	0.94	0.35

TABLE 6.5: **Perspective Attack**: level of toxicity (0.5 the threshold)

6.5 Trade-off: Performance vs. Security

In this chapter we showed several attacks that degradate the performance of the models. We can now think why this is happening and if there are some countermeasures to avoid these phenomena. We can think the problem as a trade-off given by the model, between the performance and the security, as shown by Grosse et. al. (Grosse, T. Smith, and Backes, 2018).

For showing this concept, I built a simple classifier, an Adaboost (Zhu et al., 2006), trained with 100 Decision Trees with different depths, from 1 to 5 (L. Breiman

and Stone, 1984), and I train and evaluate the model over W. The sentences are represented as one hot encoding vectors of the top 10000 common words in the training set. The models, that differs just for the depth of the trees, are evaluated twice, in the test set and in the poisoned test set, which is the test set with the common words appending attack. The results are shown in Figure 6.7.

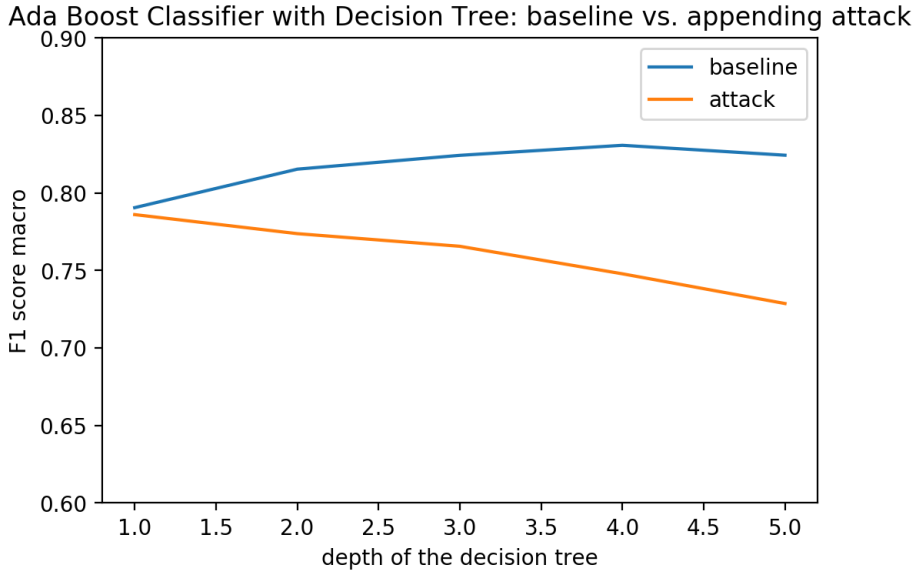


FIGURE 6.7: Trade off performance - security: a comparison of the performance over different depths of the Decison Trees in Adaboost classifier

By increasing the depth, the model perform better, but also it is more vulnerable for the attacks. It seems that the reason is that we are trying to let the model being more elastic, to generalize more, but for this reason, it allows more vulnerabilities. In the first model, the more restrictive, the attack does not work at all, but also the classification performances is the worst.

Chapter 7

Considerations and Conclusions

In this work we evaluated the performance of seven state-of-the art hate speech classifiers, which were presented in prior works. We showed that in general, the discussed techniques work equally when trained with the task's dataset, while their classification performances go down when the models are tested in other domains. It seems that the model architecture is not truly important, and complex models such as Deep Neural Networks perform as well as simpler models, such as the Logistic Regression.

Many problems rise during the cross application phase: the challenge seems to be that the dataset are quite small (the largest one is around 100 thousand samples), and from the similarity analysis of the dataset, we saw that the words used are quite similar, which mean that in different times, some few terms made the difference in the classification. The use of swearwords must be considered, for avoiding misclassifications sentences (offensive sentences perceived as hateful).

Another challenge with the data is given by the context, where, if for the offensive classification we can look for some specific terms, in the hateful we cannot do the same. For understanding this point, we can consider the following sentence:

He wants to kill all the Jews.

This, without any doubt, *MUST* to be considered as hateful sentence. Now, let's consider the following:

Hitler wants to kill all the Jews.

Are there any differences? How should we classify this sentence? As hateful or not? The fact is that this is something that happened for real, and if we consider this sentence as hate speech, we are not allowing free speak in our social network / blog about history, or science, or other topics that are not hateful. What are the differences between these two sentences? Well, the point is just one, the second is an historical fact. But we, as human being, we can see this difference, because we are linking the meaning of the sentence with other knowledge: this means that the text alone is not enough. In my opinion, in this early stage of text classification, we cannot think that these sentences can be correctly classified. Just for showing, both sentences are classified as Toxic (0.95 vs 0.92) in Google Perspective; I think that the aspect of the context must be considered to during the data retrieval, where, if future authors want to do this process, they need to be sure that those who vote will classify the sentence based on just the mere text and not external knowledge, and in this way, examples such as the previous two sentences will be considered as hateful. But because of this limitation, hate speech detector cannot be used right now.

The current architectures also do not consider the problem of the adversaries. The current architectures seems to suffers a lot the attacks, and all the word based

models became unusable when we apply *word changes*, *word boundary changes* and *word appending*. The simple *love attack* shows and highlight the problem, showing that also in online algorithms such as in Google Perspective, the decision of the model can be easily manipulated. Only the character architectures seems to be more resistance to these attacks. In my opinion here, future authors can also avoid these cases, but they should be aware to these problems, and, at least, restrict the domain of their applications. If the attack scenario is considered, we should remember that there is the trade-off between the performance and the security.

We can summarize these concepts in few points: (i) the architecture does not matter significantly in terms of classification performance, (ii) focus on the data collection and sanification, (iii) think about the swearwords, and the relevance of having those in the non hateful class(es), (iv) remember the problem of the meaning of the sentences (e.g. Hitler speech), (v) be aware of sentences' manipulations and prepare for adversaries, (vi) the trade off security-performance.

In conclusion, the hate speech is a problem that require a lot of effort for a solution, and it will be discussed in the following years. The current proposed solutions seem to be far-away from a real development. The topic is serious, and more effort must to be done.

Appendix A

Published Paper

In this appendix we report the paper published at the 11th ACM Workshop on Artificial Intelligence and Security (AISec2018) (Gröndahl et al., 2018).

All You Need is "Love": Evading Hate Speech Detection

Tommi Gröndahl
tommi.grondahl@aalto.fi
Aalto University

Luca Pajola
luca.pajola@aalto.fi
Aalto University

Mika Juuti
mika.juuti@aalto.fi
Aalto University

Mauro Conti
conti@math.unipd.it
University of Padua

N. Asokan
asokan@acm.org
Aalto University

ABSTRACT

With the spread of social networks and their unfortunate use for hate speech, automatic detection of the latter has become a pressing problem. In this paper, we reproduce seven state-of-the-art hate speech detection models from prior work, and show that they perform well only when tested on the same type of data they were trained on. Based on these results, we argue that for successful hate speech detection, model architecture is less important than the type of data and labeling criteria. We further show that all proposed detection techniques are brittle against adversaries who can (automatically) insert typos, change word boundaries or add innocuous words to the original hate speech. A combination of these methods is also effective against Google Perspective – a cutting-edge solution from industry. Our experiments demonstrate that *adversarial training* does not completely mitigate the attacks, and using character-level features makes the models systematically more attack-resistant than using word-level features.

1 INTRODUCTION

Social networking has changed the way people communicate online. While the ability of ordinary people to reach thousands of others instantenously undoubtedly has positive effects, downsides like polarization via echo chambers [13] have become apparent. This inter-connectedness of people allows malicious entities to influence opinions by posting hateful material, also known as *hate speech*.

Hate speech is not a universal concept. While laws targeting speech seen as harmful have existed throughout human civilization, the specific term was originally coined in the US in 1989 to address problems of "harmful racist speech" that was nonetheless protected in the US [4]. In 1997, the European Union defined "hate speech" as texts that "spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance."[1] Hate speech can be separated from merely offensive or shocking content [7], although this distinction is non-trivial. In this paper, we denote non-hateful speech as "ordinary speech."

Typically, hate speech detection is cast as a *classification problem*. Standard machine learning algorithms are used to derive a discriminative function that can separate hate speech from ordinary speech. Although several hate speech detection mechanisms have been reported in the research literature [24], to the best of our knowledge, there has so far been no systematic *empirical* evaluation comparing actual implementations of proposed models and datasets.

We study five recent model architectures presented in four papers. One architecture [29] is trained separately on three different datasets, giving us seven models in total. Six of these distinguish between two classes [1, 28, 29]. One classifies among three, distinguishing between offensive and non-offensive ordinary speech [7]. Two models are *character-based* [28] (using character n-grams as features) while the rest are *word-based* (using word n-grams or embeddings). In the original papers, each of the two-class models was evaluated using a particular dataset. We show that none of the pre-trained models perform well when tested with any other dataset. This suggests that the features indicative of hate speech are not consistent across different datasets. However, we also show that all models perform equally well if they are retrained with the training set from another dataset and tested using the test set from the same dataset. This suggests that hate speech detection is largely independent of model architecture. We also tested each two-class model on offensive ordinary speech [7], and observe that they tend to classify it as hate speech. This indicates that the models fail to distinguish between hate speech and offensive ordinary speech, making them susceptible to false positives. We experimented with using a *transfer learning* approach [12]. Using a pre-trained language model, we fine-tuned it for the classification tasks, and conducted the same experiments. We show that the results are comparable but do not exceed the baselines.

Prior work has only considered what can be called *naive adversaries*, who do not attempt to circumvent detection. We show that all the models are vulnerable to adversarial inputs. There are many ways of attacking text-based detection models. A simple attack involves changing the input text so that a human reader will still get the intended meaning, while detection models misclassify the text. We suggest three such alteration techniques, all of which are easily automated: (i) word changes, (ii) word-boundary changes, and (iii) appending unrelated innocuous words. Implementing two varieties of each attack, we show that all detection models are vulnerable to them, although to different extents.

Combining two of our most effective attacks, we present a simple but powerful evasion method, which completely breaks all word-based models, and severely hinders the performance of character-based models. In addition, this attack significantly degreades the performance of Google Perspective API,[2] which assigns a "toxicity" score to input text.

[1] https://www.echr.coe.int/Documents/FS_Hate_speech_ENG.pdf

[2] https://www.perspectiveapi.com/

We summarize our contributions as follows:

- The first experimental comparative analysis of state-of-the-art hate speech detection models and datasets (Section 2).
- Several attacks against effective against all models and possible mitigations (Sections 3).
- A simple but effective evasion method that completely breaks all word-based classifiers, and significantly impacts character-based classifiers as well as Google Perspective (Section 4.4).
- A review of the limitations in current methds, and desiderata for future developments (Section 7, Section 5).

2 REPLICATION AND MODEL COMPARISON

We begin by describing four recent papers [1, 7, 28, 29] on hate speech detection (Section 2.1). We reproduce and systematically analyze the performance of seven models presented in these papers (Section 2.2). The datasets focus on partially disjoint aspects of "hate": e.g., hate speech based on religion or ethnicity [29] may not be highly similar to sexually connotated hate speech [1]. As all datasets except one are drawn from Twitter, they do not properly represent the range of online text. We believe future research should focus on collecting and properly evaluating datasets, especially outside Twitter.

We tested the performance of all pre-trained models against four datasets. We further re-trained the proposed models on each of the other three datasets, and compared the results. All model architectures performed comparably well, when the training set and test set were taken from the same dataset. However, our results indicate that hate speech detection is highly context-dependent and transfers poorly across datasets.

2.1 Models and datasets

In this section, we describe all replicated models and their respective datasets. The datasets, labelled W, T1, T1*, T2, and T3, are summarized in Table 1, and Table 2 provides additional statistics on sentence lengths. All datasets were divided between a training set used to train the model, and a test set used for measuring model performace.

Each paper proposes a different machine learning model for hate speech detection: two papers use feature extraction -based models [7, 28] and two use recurrent neural networks [1, 29]. All models lowercase content and remove punctuation. We summarize the models and datasets in Table 3, and discuss them in the remainder of this section.

W [28]: As a part of Wikipedia's "Detox" project targeting personal attacks in Wikipedia's edit comments, Wulczyn et al. [28] experimented with logistic regression (LR) and multilayer perceptron (MLP) models, using n-gram features on both the character- and word-level. Word n-gram sizes ranged from 1 to 3, and character n-gram sizes from 1 to 5. Labels were gathered via crowdsourcing, each comment being labeled by 10 evaluators. We denote this dataset as W. Wulczyn et al. made their tests on both the one-hot encoded majority vote between the two classes (attack or non-attack), and empirical distributions based on different votes. The former gives a classification, whereas the latter may be interpreted as class probabilities. On one-hot encoded labels, character-models performed better both in the original paper and our replication. We therefore use these models in our tests.

Dataset	Domain	Classes (size)	Source
W	Wikipedia	personal attacks (13590) ordinary (102274)	[28]
T1	Twitter	hate speech (1430) offensive (19190) ordinary (4163)	[7]
T1*	Twitter	hateful (1430) offensive ∪ ordinary (23353)	[7]
T2	Twitter	racist ∪ sexist (5013) ordinary (10796)	[27]
T3	Twitter	hateful/racist (414) ordinary (2021)	[29]

Table 1: Datasets used in our replication. Union (∪) denotes the conflation of class elements.

Dataset	Mean	Std.	Min.	Max.	25%	50%	75%
W	84	161	1	9949	20	42	89
T1, T1*	20	12	1	321	11	18	27
T2	20	8	1	45	14	21	26
T3	20	7	1	48	16	21	25

Table 2: Sentence lengths in the datasets.

Model	Dataset(s)	Source
LR char	W	[28]
MLP char	W	[28]
LR word	T1	[7]
CNN+GRU	T1*, T2, T3	[29]
LSTM	T2	[1]

Table 3: Replicated machine learning models.

T1 [7]: Davidson et al. [7] presented a dataset with three kinds of comments from Twitter: hate speech, offensive but non-hateful speech, and neither. This is, to our knowledge, the only dataset with such a distinction. The hate speech data was collected by searching for tweets with known hate speech phrases,[3] and further labeling these tweets with a majority vote from three CrowdFlower workers each . We denote this dataset as T1. A vast majority of the dataset contains offensive speech (76%), and only a small portion actual hate speech (5%). Davidson et al. use a word-based logistic regression (LR) classifier (1 − 3-grams), which we replicated.

T2 [1]: In their paper, Wulczyn et al. [28] mention that future work should involve the use of deep neural networks (DNNs), which have proven useful in a number of text classification tasks [10]. Currently, the dominant approaches to text classification use recurrent neural networks (RNNs), and in particular long short-term memory (LSTM) networks. These are applied for hate speech classification by Badjatiya et al. [1], who use them both alone and with

[3]The phrases were collected from https://www.hatebase.org/

gradient-boosting decision trees (GBDTs). For the latter, they first train the LSTM on the training data, and then use the average of the word embeddings learned by the LSTM as the input for the GBDT. They report a major increase in performance when compared with only using LSTMs. However, we were unable to replicate these results with the code they provide,[4] and received a better score using only the LSTM. Therefore, we use the LSTM alone instead of LSTM+GBDT. Like Badjatiya et al., we initialize the word embedding layer randomly.

As data, Badjatiya et al. use a corpus of 16000 tweets originally collected by Waseem et al. [27]. These contain three possible labels: "racism", "sexism", and "neither". In our experiments, we combined the "racism" and "sexism" classes into one, comprising general hateful material. We denote this dataset as T2.

T1*, T3 [29]: In addition to LSTMs, convolutional neural networks (CNNs) have been popular in text classification research. Badjatiya et al. [1] experiment with them, as do Zhang et al. [29]. The latter add CNNs and RNNs together, by giving the output of a CNN to a gated recurrend unit (GRU) network. Word embeddings were initialized with Google word vectors trained on news data.[5]

Zhang et al. [29] use dataset T1 from Davidson et al. [7], but combine offensive and ordinary speech into one category, as opposed to genuine hate speech. We denote this dataset as T1*. Zhang et al. further created their own small dataset of hate speech targeted toward refugees and muslims, which we denote as T3. Finally, in addition to T1* and T3, they experimented with T2. We replicate all three experiments, resulting in three distinct CNN+GRU models.

2.2 Model performance

In this section, we present replications of the original models (Section 2.2.1), and cross-apply all two-class models to all two-class datasets (Section 2.2.2).

2.2.1 Replication and re-training. In the original papers, the hyperparameters of each model have been optimized to the particular training sets used. However, we additionally trained each model with *every* dataset, and compared the results. As reported in Table 4, all models perform comparably on all four datasets.

Model	Dataset			
	W	T1*	T2	T3
LR char	**0.86**	0.63	0.82	0.85
MLP char	**0.86**	0.63	0.81	0.85
CNN+GRU	0.87	**0.70**	**0.83**	**0.81**
LSTM	0.85	0.64	**0.78**	0.79

Table 4: F1-scores (macro-averaged across classes) of the two-class models trained and evaluated on each dataset (datasets used in original papers in bold).

The results indicate that the models are roughly equally effective when applied to different texts, provided that they were trained using the same kind of text.

[4] https://github.com/pinkeshbadjatiya/twitter-hatespeech
[5] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

Inferior performance on T1* by all models can be explained by two factors. First, the dataset is highly imbalanced, with the "hate" class taking up only 5% of the training set. Second, this dataset is derived from Davidson et al.'s original three-class corpus that separates *offensive speech* from hate speech. To constrain classification into only two categories, T1* assimilates the "offensive" and "non-hate" classes into one. Of these, the offensive class dominates, covering roughly 80% of the combined class in the training set. It may have more overlap with hate speech than non-offensive speech, making classification more challenging.

2.2.2 Cross-application between datasets. To estimate the adaptivity of models pre-trained with one dataset, we applied them to all test sets. We first trained each model with the training data used in the original paper presenting it, and then applied the resulting classifier to the test sets of all four datasets.

Model, training dataset	Dataset			
	W	T1*	T2	T3
LR char, W	(0.86)	0.37	0.50	0.24
MLP char, W	(0.86)	0.38	0.50	0.25
CNN+GRU, T1*	0.11	(0.70)	0.48	0.51
CNN+GRU, T2	0.14	0.28	(0.83)	0.44
CNN+GRU, T3	0.13	0.48	0.50	(0.81)
LSTM, T2	0.23	0.33	(0.78)	0.47

Table 5: F1-scores (macro-averaged) of pre-trained two-class models applied to different test sets, with the original test set in parentheses.

As we can observe from the results reported in Table 5, none of the pre-trained models transfer well to any other dataset. These results indicate that linguistic hate speech indicators are not well retained across different datasets. This may be due to the lack of words shared between datasets, or differences between the relevant features of particular subcategories of hate speech.

2.3 Offensive speech vs. hate speech

An important distinction can be drawn between hate speech and speech that uses offensive vocabulary without being hateful. The latter is separated from hate speech in Davidson et al.'s dataset T1 [7]. Especially given the lack of any precise definition of hate speech in either legal or academic contexts, it is important to investigate the extent to which existing approaches are sensitive to the hateful-offensive distinction.

A particular concern is the extent to which two-class models assign offensive but non-hateful text to the "hate" class. This can be treated as a false positive, assuming the hateful-offensive distinction is appropriate. Of course, the assumption is not always clear: for instance, Google Perspective[6] offers to detect "toxic" comments; but whether "toxicity" should include all offensive text remains a subjective assesment.

To estimate the performance of the models on offensive but non-hateful text, we applied the "offensive" datapoints from T1's test set to all two-class models, except CNN+GRU trained on T1*, which

[6] https://www.perspectiveapi.com/

is built from T1 and hence not independent of the task. As seen in Table 6, CNN+GRU trained on T3 was the only model to succeed in this task. All other models performed on a random or below-random level.

Model, training dataset	Assignment to non-hate
LR char, W [28]	0.40
MLP char, W [28]	0.36
CNN+GRU, T2 [29]	0.23
CNN+GRU, T3 [29]	0.89
LSTM, T2 [1]	0.36

Table 6: Performance of two-class models on offensive but non-hateful speech.

Additional experimentation revealed that the seeming success of CNN+GRU trained on T3 was due to the prevalence of unknown words in the "offensive" test set, mapped to the unknown token (<*unk*>). This token was associated with the non-hateful class in the model, and on average over 40% of words per sentence in this test set were mapped to it. Hence, the performance simply reflected the model's small vocabulary.

The results are suggestive of the problematic and subjective nature of what should be considered "hateful" in particular contexts. As the labels have been manually gathered via crowd-sourcing, there is no guarantee that labels from different people were consistent. Within datasets, this problem has can be addressed to some extent by majority voting, but there is no method of guaranteeing agreement between datasets. It seems that the labeling in the two-class models represents offensiveness more than hatefulness as such.

Manual experimentation suggests similar conclusions concerning Google Perspective. Some examples of Perspective's scores on non-hateful sentences appended with a common English curse-word (marked with "F" here, but in original form in the actual experiment) are presented in Table 7.

Sentence → Modified sentence	Old → New score
You are great → You are F great	0.03 → 0.82
I love you → I F love you	0.02 → 0.77
I am so tired → I am F tired	0.06 → 0.85
Oh damn! → Oh F!	0.64 → 0.96
Food is amazing → Food is F amazing	0.02 → 0.68

Table 7: Google Perspective "toxicity" scores on non-hateful sentences with and without a curse word.

None of the examples in Table 7 are hateful, but the curse-laden variants are all considered "toxic" with a high likelihood. Clearly "toxicity", as Perspective currently classifies it, is not assimilable to hate speech in any substantive (or legal) sense.

2.4 Transfer learning

The replicated models use both classical machine learning methods, and more recent deep neural networks (DNNs). Recently, *transfer learning* has been argued to improve performance text classification [12]. This approach to deep learning is based on fine-tuning a pre-trained model for a particular task, instead of training the entire model from scratch. A natural question to ask is whether performance on our datasets could be improved by using transfer learning.

We implemented a particular transfer learning method called *ULMFiT* [12], based on the code the authors provide.[7] The model uses a language-model pre-trained on a large dataset from Wikipedia [17], fine-tunes it for a particular dataset, and then fine-tunes a classifier on top of the language-model. We refer the reader to the original paper for technical details [12].

Our preliminary results indicate that ULMFiT is unable to reach the baselines from the replicated models. On the Twitter datasets T1–T3, ULMFiT results (0.62, 0.75, 0.80) remained below all respective baselines (0.66, 0.84, 0.86). However, more systematic evaluation of transfer learning is needed in future research, and we are currently working on this question.

2.5 Summary of replication and cross-application

All four two-class models perform highly similarily when trained on each of the four datasets. In particular, neither the features used (characters vs. words) or model complexity influenced the test score in any significant way. This suggests that the features learned by all models (including LSTM and GRU models) are similar to relatively simple n-grams, as opposed to involving more complex relations between features.

3 ATTACKS

Adversary model The goal of the adversary we consider in our work is to fool a detection model into classifying hate speech input as ordinary speech. We assume that the adversary has complete control of the input and can modify it to evade detection while retaining the semantic content of the original hate speech. We do not assume that the adversary has whitebox access to the model parameters. The adversary model is relaxed in one attack, where some knowledge of the training set is required.

Classifiers that rely on surface-level features can be attacked with malformed input, constituting *evasion attacks* [15]. In this section we describe six evasion attacks against the seven classifiers we replicated. All attacks are based around altering input text with easily implementable and automatic methods. We categorize the attacks into three types, and experiment with two alternatives from each:

- **Word changes**
  1. Inserting typos
  2. Leetspeak [22]
- **Word-boundary changes**
  1. Inserting whitespace
  2. Removing whitespace

[7] https://github.com/fastai/fastai

- **Word appending**
    1. Appending common words
    2. Appending non-hateful words

We differentiate the attack types, because they target distinct (albeit sometimes overlapping) aspects of the classifier. Word changes are done to change the identities of words. For word-level models, the new identities are likely to become "unknown" (i.e. non-identified tokens denoted by <*unk*>), while retaining the readability and semantic content of the original text to a maximal extent from a human reader's perspective. Word boundary changes, on the other hand, not only alter word identities but also the sentence structure from a superficial perspective. Finally, word appending makes no alterations to the original text, but only adds unrelated material to confuse classification.

There are theoretical reasons to believe certain attacks to be more effective against some classifiers than others. In particular, as word tokenization plays no role in character-models, it is clear that these are less susceptible to word-boundary changes than word-models. Furthermore, character-models are expected to be more resistant to word changes, as many character n-grams are still retained after the transformations are applied. In contrast, all word-based models are expected to be vulnerable to attacks that alter word identities (i.e. both word changes and word-boundary changes). Word appending attacks, in contrast, have no *a priori* reason to work better against character- than word-based models, or vice versa.

We applied the attacks to all seven classifiers, by transforming the hate class samples from the respective test sets.

3.1 Word changes

Word change attacks introduce misspellings or alternative spellings into words. They may make words entirely unrecognizeable to word-based models, and change the distribution of characters in character-based classification.

In 2017, Hosseini et al. [11] showed that Google's "toxicity" indicator Perspective could be deceived by typos. However, Perspective has since been updated, and the examples the authors provided no longer succeed to the extent they did then. We review these results in Section 4.4. Still, we consider the proposal to be worth further investigation with larger datasets and automatic typo-generation.

In addition to typos, we also consider a simplified variety of *leetspeak*, which is a type of Internet slang replacing characters with numbers. Different variants may involve more alterations, but we only consider character-to-number alterations here. Leetspeak has been shown to be easily readable to humans [22], but will be unrecognizable to word-models unless it is also present in the training set.

3.1.1 Algorithms. **Inserting typos:** As an attack, typo generation has three desiderata: (i) reducing the detection likelihood of hate speech, (ii) avoiding correction by automatic spell checkers, and (iii) retaining the readability and meaning of the original text. If the second goal is not met, the defender can include a spell-checker as a pre-processing stage in the classification. Satisfying the third goal requires the word to remain recognizable, and not to be mapped to some other word in the reader's mental lexicon.

To make the attack succesful, word-level changes must not only fool the classifier, but be human-readable and retain the original interpretation to a maximal extent. This means that we cannot simply introduce typos via random changes to words. Instead, the change must have minimal impact on readability. We utilize the empirical finding from cognitive psychology that characters in the middle of words have a smaller effect on readability than the first and last characters [23]. To maximize understandability, we restricted the alterations to a single switch between two characters.

The algorithm switches the order of two characters in the word, excluding the first and final character. The probability of choosing particular characters is calculated by two factors: characters closer to the middle of the word are preferred, and characters that are close to each other are preferred. First, a character is chosen between the second and second-to-last character of the word, based on a Gaussian distribution centering in the middle. Hence, only words with four of more characters are applicable. Next, a second character is chosen based on the combined effect of the first distribution and a second Gaussian distribution centered around the previously chosen word. Two random non-edge characters are consequently chosen, favoring characters in the middle of the word. These characters are then switched to create the typo.

Leetspeak: To transform original text into simple leetspeak, we introduce the following character replacements:

(a: 4) (e: 3) (l: 1) (o: 0) (s: 5).

The changes retain readability well, given the visual similarity between the original and leet characters [22].

3.1.2 Mitigation. We tried two methods of improving classifier performance on test data modified via word changes: *adversarial training*, and adding a *spell checker* to test set pre-processing.

Adversarial training: We augmented the training set with stochastically transformed versions of the original training examples, for all classes (doubling the training set size). The purpose of adversarial training was to add transformed varieties of each word into the model's vocabulary, making it more likely for the model to associate them with the correct class.

The random nature of the typo-algorithm limits the scalability of the approach with long words, as the range of possible typos becomes larger. In contrast, given the deterministic nature of our leetspeak algorithm, we can expect adversarial training to work well against it.

Spell checking: We added a spell-checker to the pre-processing stage of the test set to find out how resistant our typo-introduction algorithm was to automatic correction. We used Python's *Autocorrect* library for this.

3.2 Word boundary changes

The tokenization attack differs from word transformation by retaining word-internal characters, but introducing or removing characters that result in a word-based model separating between different tokens. We use space as the most evident choice.

3.2.1 Algorithms. We implemented two simple algorithms for introducing or removing whitespace. We predict removal to be more effective against word-based models on theoretical grounds.

Character-based models are likely not highly susceptible to either variant.

Inserting whitespace: Word-based models rely on tokenizing the text into a sequence of words, based on characters treated as word-separators. Therefore, a simple way to make words unrecognizable is to change the tokenization by introducing additional separators between words. The effect of this attack on readability is small, but it results in most words becoming unrecognized by the classifier.

We used a simple approach of splitting each (content-)word into two by randomly selecting a character from the word and adding a space before it. In a word-based model a previously recognized word will turn into <*unk*> <*unk*>.

Removing whitespace: Conversely, removing all spaces leaves a single <*unk*> datapoint. Word-based models' performance will then depend entirely on how this single token is classified. Character-models, in contrast, will only lose the information related to the space token, which might deteriorate their performance, but likely not much.

This attack has a marked negative impact on surface-level readability, but still allows the reader to recover the original content. We take the adversary's main goal to be getting his/her message across to their target, even if this required some effort on other end. As English sentences are rarely reformulable into other grammatical sentences only by changing whitespace distribution, ambiguity does not arise and information loss is marginal.

3.2.2 Mitigation. **Adversarial training:** For the whitespace insertion attack, we appended the training set with randomly split versions of the original data to include word-parts into the model's vocabulary (thus doubling the training set). Given that an n character word can be split in two in $n-1$ possible ways, going through all possible combinations of splits in a sentence quickly results in a combinatory explosion. Hence, the effect of adversarial training is expected to be limited on analytic grounds.

For completeness, we also conducted adversarial training via space removal, although this is close to useless on the grounds that it only adds whole comments as single words to the model vocabulary, and associates them with one class. As this method can only have an effect if it encounters the exact comment again, it cannot scale and hence its relevance is close to none in word-based models. Character-models, in contrast, can make use of it, but its inclusion is mostly redundant, as the datapoints are not changed much in comparison to the originals.

Removing spaces Another mitigation method is available only for character-based models, where spaces are removed from both training and test data at the pre-processing stage. This method will by necessity remove any effect of adding spaces, but will also remove potentially relevant information from the data, as one common character is removed entirely. The effectiveness of this mitigation method thus depends on the relevance of the space character for the classification.

3.3 Word appending

All text classification is based on the prevalence of class-indicating properties in the data. Therefore, adding material that is indicative of one class oven another makes it more likely for the classifier to assign the sample to that class. In many tasks this is appropriate, but hate speech detection constitutes an important exception. Hate speech and non-hate speech are not reversible: adding hateful material to ordinary speech *will* turn the text into hate speech, whereas adding non-hateful material to hate speech *may* not change its status. This vulnerability invites a possible attack, where non-hateful material is inserted to hate speech to change the classification.

For our attack model, we assume that the adversary is aware of the use of additional unrelated words to distract automatic classifiers. Assuming also that the beginning of the additional material is clear enough from the discourse, readability and semantic retainment are secured.

3.3.1 Algorithms. We generated a random number (between 10 and 50) or words at the end of each text in the "hate" class of the test set. The words were randomly chosen from two possible sources, yielding two varieties of the attack.

Appending common English words: Google has provided a list containing the most common words appearing in a large book corpus [18]. We used random words from the top 1000 of these words, excluding stopwords. The rationale behind this attack is that it requires no knowledge of the training data on the part of the attacker. Further, the common English words are likely to be contained in the training corpus of many different datasets.

Appending common "non-hate" words: Here, we assume the attacker knows, or correctly guesses, a list of words in the training data's non-hateful class. He then appends the text with randomly chosen common (content) words from this class. We are still not assuming white-box access to the model itself.

4 RESULTS

We performed our six attacks on each of the seven model-dataset combinations replicated from prior work. This yields 42 attacks in total. We used adversarial training to mitigate all attacks, and tried additional defence methods for the typo attack (spell-checking) and tokenization attacks (space removal in training; this method is only available for character-models).

Attack effectiveness varied betweeen models and datasets, but the performance of all seven hate speech classifiers was significantly decreased by most attacks. Word-based models were most affected by tokenization changes, and character-based models by word appending. A significant difference between word- and character-based models was that the former were all completely broken by at least one attack, whereas the latter were never completely broken. The two character-models performed comparably across all attacks.

The training dataset had a major influence on attack resilience, as demonstrated by the differences between the CNN+GRU model trained with T2 and T3, respectively. However, CNN+GRU trained on T2 was more resilient to all attacks than LSTM trained on T2, indicating that model choice also had an effect.

The results from all attacks and their mitigation are presented in Table 8. Since our attacks affect datapoints in the "hate" class, we only report the F1-scores from this class.

Model, DS	Orig.	Word changes					Boundary changes						Word appending			
		Typos			Leet		Insert			Remove			Common		Non-hate	
		A	AT	SC	A	AT	A	AT	RW	A	AT	RW	A	AT	A	AT
LR char, W	0.75	0.60	0.71	0.68	0.61	0.74	0.75	0.71	0.74	0.54	0.75	0.74	0.48	0.68	0.47	0.67
MLP char, W	0.75	0.55	0.71	0.68	0.59	0.73	0.75	0.71	0.72	0.56	0.76	0.72	0.50	0.72	0.48	0.67
CNN+GRU, T1*	0.43	0.31	0.35	0.36	0.00	0.33	0.04	0.34	–	0.00	0.00	–	0.04	0.38	0.01	0.27
CNN+GRU, T2	0.76	0.27	0.67	0.68	0.09	0.77	0.43	0.66	–	0.00	0.00	–	0.64	0.75	0.50	0.74
CNN+GRU, T3	0.69	0.23	0.61	0.43	0.03	0.76	0.08	0.63	–	0.00	0.00	–	0.18	0.70	0.14	0.64
LSTM, T2	0.70	0.40	0.66	0.67	0.19	0.71	0.42	0.64	–	0.00	0.02	–	0.27	0.68	0.15	0.69
LR word, T1	0.50	0.30	0.42	0.37	0.04	0.48	0.18	0.44	–	0.01	0.02	–	0.48	0.44	0.45	0.30

Table 8: F1-scores on the "hate" class from attacks and mitigation.
Attack: A; Mitigations: AT = adversarial training, SC = spell-checker, RW = removing whitespace (character-models only)
Expected pattern is attack reducing score and mitigation restoring it; deviations highlighted and discussed in sections 4.1–4.3.

4.1 Word changes

Word-models were more susceptible to leetspeak than typos, whereas no clear difference can be found in character-models. In addition, word-models were much more vulnerable to both attacks than character-models. Adversarial training had a major positive effect on performance against both attacks, but its effect was larger on the leetspeak attack. This is unsurprising given the deterministic nature of the leetspeak algorithm. The determinacy also indicates that the leetspeak attack could easily be mitigated by a counter-algorithm transforming numbers into corresponding characters.

4.2 Word boundary changes

Neither character-model was affected by the whitespace insertion attack, but the performance of both was markedly decreased by whitespace removal. We suggest this may be due to the fact that whitespace is involved in the beginnings and ends of words. Unlike adding whitespace, removing it abolishes all n-grams concerning word boundaries, which may be especially relevant for classification.

All word-models were completely broken by white space removal, and severely hindered by whitespace addition. As expected, adversarial training had no impact on whitespace removal. Resistance to whitespace addition, in contrast, was improved, and reached levels close to the baseline, differing from it only $6-10\%$. Overall, removing whitespace was much more effective than its addition for all models, both as an attack and in avoiding adversarial training as mitigation.

Character-models peformed slightly worse when trained without spaces, but not much, the largest drop in F1-score being 3%. We conclude that removing spaces in pre-processing during both training and testing makes character-based models resistant to tokenization attacks with only a minor reduction in predictive power. No comparable mitigation exists for word-models, where word boundary removal will force the text to be tokenized as a single <*unk*>.

4.3 Word appending

Unlike other attacks, word appending affected character- and word-models comparably. Words from the non-hate class of the training set had a systematically larger impact than common English words, but the difference was very minor $(1-2\%)$ on character-models. The largest difference was observed on the LSTM model, where non-hate words had almost twice the effect of common words (0.27 vs. 0.15).

The only model not affected by either appending attack was the three-class word-based LR model from Davidson et al. [7], trained on T1. We attribute this result to the fact that the major non-hate class of this dataset was the "offensive" class. Common English words or words from the "neither" class rarely indicate offensive speech, making it unlikely for the hate speech to be classified as such. This data imbalance also likely explains the negative effect of adversarial training, which was not observed on any other model.

Adversarial training worked very well on all two-class models, reaching predictive power close to the baseline. The effect was the smallest with CNN+GRU trained on T1*, leaving 16% behind the baseline with adversarial training. The dataset T1* is drawn from T1 by combining offensive and non-offensive ordinary speech into a single class. As offensive speech takes the overwhelming majority of T1, T1* is highly imbalanced. We therefore expect adversarial training to result in the appended words to associate with the "non-hate" class more readily than the "hate" class, which would account for its limited success in mitigation.

4.4 Adding "love"

Finally, we present the results from our attack combining the two most powerful approaches we experimented with: whitespace removal and word appending.

Whitespace removal turns the sentence into a single <¡*unk*¿>, making the classification entirely dependent on the model's prediction of this particular token. Models might behave differently with respect to it, and hence the effects of whitespace removal can be uncertain. This problem can be remedied by adding words strongly indicative of the non-hate class, effectively forcing the model to prefer it.

Furthermore, instead of using a long list of non-hateful words, we minimize the hindrance on readability by appending the text with only one word: "love". We choose this word because, intuitively, it is likely to negatively correlate with hate speech. Our results support this hypothesis.

As expected, the attack completely broke all word-models, and significantly hindered character-models (Table 9). Based on the

Model (data, classes)	Original	"Love"
LR char (W, 2)	0.75	0.52
MLP char (W, 2)	0.75	0.55
CNN+GRU (T1*, 2)	0.36	0.01
CNN+GRU (T2, 2)	0.79	0.00
CNN+GRU (T3, 2)	0.78	0.00
LSTM (T2, 2)	0.74	0.00
LR word (T1, 3)	0.50	0.00

Table 9: F1-scores (macro) from the attack of removing word boundaries and adding "love".

word appending results, the performance on character-models would decrease with more appended non-hateful words. Since the original message can relatively easily be recovered, and the single unrelated word has a minimal effect on readability (assuming it to be separable from the rest by the reader), we consider this attack highly successful against state-of-the-art hate speech classification.

We additionally experimented with Google Perspective, using example sentences from Hosseini et al. [11], reproduced below.

- **S1:** Climate change is happening and it's not changing in our favor. If you think differently you're an idiot.
- **S2:** They're stupid, it's getting warmer, we should enjoy it while it lasts
- **S3:** They are liberal idiots who are uneducated
- **S4:** idiots. backward thinking people. nationalists. not accepting facts. susceptible to lies.
- **S5:** They are stupid and ignorant with no class
- **S6:** It's stupid and wrong
- **S7:** If they voted for Hilary they are idiots
- **S8:** Anyone who voted for Trump is a moron
- **S9:** Screw you trump supporters

Hosseini et al. provide examples of introducing manual typos, punctuation and spaces in each sentence, and demonstrate its effects on Google Perspective. We applied the same modifications to the same example sentences, but received much higher toxicity scores. This indicates that Perspective has been updated, and now performs better on adversarial data. Nevertheless, we managed to reduce the toxicity score to below 0.40 on all example sentences with our attack of removing word boundaries and adding "love" (Table 10). This number is Perspective's upper limit for the label "unlikely to be considered toxic".

Sentence	Original	Modified	
		[11]	"Love"
S1	0.95	0.94 (0.20)	0.35
S2	0.92	0.38 (0.02)	0.37
S3	0.98	0.79 (0.15)	0.37
S4	0.95	0.90 (0.17)	0.37
S5	0.97	0.53 (0.11)	0.37
S6	0.88	0.82 (0.17)	0.35
S7	0.99	0.70 (0.12)	0.15
S8	0.96	0.64 (0.13)	0.35
S9	0.90	0.78 (0.17)	0.35

Table 10: Google Perspective "toxicity" scores for S1–S9. Scores from manual modifications [11] in third column; results reported in the original paper in parentheses.

5 DISCUSSION

We evaluated the performance of seven state-of-the art hate speech classifiers presented in prior work. We showed that these techniques work roughly equally with different datasets, provided that the training and testing are based on the same dataset. However, we identified three main deficiencies in the models: (i) lack of effective transferability across datasets, (ii) conflation of hate speech and offensive ordinary speech, and (iii) susceptibility to simple text modification attacks. The first two arise from the problematicity of the concept of "hate speech", which can differ across datasets, and may or may not include all offensive or inappropriate material depending on the context.

Our attacks were much more effective against word-models than character-models. Most effective was the very simple "love" attack, which managed to completely break all word-models, and severely hinder the performance of character-models. We further demonstrated the attack's ability to reduce Google Perspective's toxicity score to below the threshold of 0.40 in all our example sentences.

In this section we present some consequences for future work that, we suggest, are implied by our findings.

5.1 Transferability and false positives

No two-class model performed well on other datasets, and all of them classify most offensive ordinary speech as hateful. Manual experimentation showed that Google Perspective functions similarily, as adding curse words to otherwise benign text drastically increases the toxicity score. These results are indicative of two related problems.

First, the standards of ground-truth labeling likely varied across different datasets. For example, catagories like "sexism/racism" (T2) might be appropriate for some comments which are not "personal attacks" (W), or vice versa. This problem may not be fatal to institutions that wish to target particular subtypes of hate speech, as long as appropriate labels are available for sufficient training data. Our cross-application results further indicate that, for classification performance, model type matters less than dataset. However, the problem is more serious for the task of more general hate speech detection, as undertaken in law enforcement or academic research.

Second, with the exception of Davidson et al. [7], the distinction between hate speech and more generally "inappropriate" material is typically not made clear. Google Perspective does not distinguish between hate speech and offensive speech, characterizing their "toxicity" metric as the means to identify a "rude, disrespectful or unreasonable comment that is likely to make you leave a discussion".[8] Hence, the problem is not only that offensive ordinary speech can constitute a false positive. Rather, it is not clear where the boundary between true and false positives should lie.

We can, however, assume that a datapoint constitutes a genuine false positive at least when the only reason it is allocated to the hate

[8] https://www.perspectiveapi.com

class is because it contains curse words not used to offend any particular person or group. For future work, we therefore recommend experimenting with methods that discard curse words at least on the unigram level.

5.2 Evasion attacks

Our results show that character-models are much more resistant to simple text-transformation attacks against hate speech classification. While this is not theoretically surprising, to our knowledge it has not been taken into account in prior work on the topic. In particular, space-removal during training has only a minor negative effect on performance, mitigates all tokenization attacks, and is only available for character-models. Further, while we received the best results overall on CNN+GRU, the simple character-models fared reasonably well in comparison to word-based DNNs. We conclude that using character-models instead of word-models is the most effective protection against our attacks.

Nevertheless, all models were susceptible to the word appending attack. This was expected, since the other attacks are built around the transformation of word identities, but the appending attack has a different basis. It takes advantage of a fundamental vulnerability of all classification systems: they make their decision based on *prevalence* instead of *presence*. The status of some text as hateful is a matter of it containing some hateful material; whether this material constitutes the majority of the text is irrelevant. Classification, in contrast, looks at the entire sentence and makes its decision based on which class is more represented on average. In principle, any text classification system can be forced to make a particular prediction by simply adding enough material indicative of one class.

We can re-phrase the problem by referring to the distinction between classification and *detection*, where the latter consist in finding some features of a relevant sort, regardless of the prevalence of other material in the datapoint. In particular, we suggest reconceptualizing hate speech detection as *anomaly detection*, where hate speech constitutes an anomalous variant of ordinary speech.

As our attacks target commonly used model architectures in text classification, they are generalizable beyond the task of hate speech detection. Possible other attack scenarios involve e.g. fooling sentiment analysis, author anonymization [2, 3], or avoiding content-based text classification to escape censorship.

On the other hand, our attacks only concern textual content, and hence do not hinder hate speech detection methods based around meta-features concerning user behavior [24]. Given both the simplicity and effectiveness of our attacks, focusing on meta-level approaches instead of text classification can be a useful direction for future research.

6 ETHICAL CONSIDERATIONS

For our replication and cross-application, we only used freely available online datasets and models. We collected no data ourselves, and none of our tests involved human subjects. Since our original code constitutes a series of attacks, we do not provide it open source. However, we will make it available for *bona fide* researchers to facilitate reproducibility.

7 RELATED WORK

In their survey on hate speech detection, Schmidt and Wiegand [24] categorize the features used in prior research into eight categories:

(i) simple surface features
(ii) word generalization
(iii) sentiment analysis
(iv) lexical resources
(v) linguistic features
(vi) knowledge-based features
(vii) meta-information
(viii) multimodal information

Focusing only on linguistic features, we disregard (vii)–(viii).

Of simple surface features, character n-grams have been argued to perform better than word n-grams, since they can detect similarities between different spelling variants [16]. These results are in line with ours. Word generalization has traditionally involved word clustering, i.e. assimilating similar words [26], and more recently word embeddings. However, the superiority of embeddings over n-grams is not empirically attested, as both character and word n-grams have performed better when compared with embeddings in hate speech classification [20].

DNNs typically include a word embedding layer in the beginning of the network, which is also true of the models we experimented with (LSTM, CNN+GRU). Prior to training, the embedding layer can be initialized randomly, or initialized by pre-trained embeddings like word2vec [19] or GloVe [21] Of the models we used, the LSTM [1] embeddings were randomly initialized, whereas the CNN+GRU [29] embeddings were initialized with Google embeddings trained on a news corpus.[9]

Sentiment analysis can be incorporated to the process either as a prior filter [9], or as a feature used directly for hate speech classification. Of the models we experimented with, the three-class LR-model of Davidson et al. [7] includes sentiment as a textual feature. One important domain of future work involves applying our attacks on state-of-the-art sentiment classifiers to see if they can be broken to the same extent with simple text transformation methods.

The remaining (linguistic) feature-types (iv)–(vi) consist of more traditional, often hand-crafted, features and rules. Lists of hateful words are available online,[10] and can be appended to other features in aid of hate speech detection. As stand-alone features, their performance is weak in comparison to n-grams [20, 24]. Of linguistic features applied to hate speech classification, the most common have been part-of-speech tags and dependency relations [5, 6, 20]. Knowledge-based approaches based on automatic reasoning can help in detecting particular patterns related to hate speech [8], but do not scale beyond those patterns.

A general trend within NLP in recent years has been a shift toward using DNNs as opposed to more traditional keyword- and rule-based methods, or traditional machine learning approaches building on simple sparse features like n-grams [10]. However, our attack results indicate that reconsidering some older methods

[9] https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[10] The most extensive of such lists are currently found in https://www.hatebase.org/.

could be useful, as they may be more resistant toward the word appending attack. In particular, keyword-based approaches are not vulnerable to the class asymmetry problem, as the mere presence of hate-indicative keywords is relevant, irrespective of the presence of other words.

Outside of hate speech detection, text obfuscation -based evasion attacks have been conducted to avoid *spam detection*, especially Bayesian models [25]. In particular, our word appending attack bears a close similarity to what have been called "good word attacks" on spam filters [14, 30]. Here, the spammer injects words that are treated as indicative of legitimate text by the filter, with the goal of flipping the model's class prediction. Despite these well-known attacks on spam detection, analogical cases for hate speech have so far been neglected in the literature. We hope to alleviate this problem.

8 CONCLUSIONS AND FUTURE WORK

Our replication and cross-application results suggest that model architecture has no major impact on classifier performance. Additionally, the simplest model (LR-char) performed comparably to more complex models, indicating that the positive effect of using more complex architectures is only minor. Cross-application further demonstrated that model complexity did not help to improve scalability across datasets. Instead, the problem stems from the labels themselves, the grounds of which can differ between the datasets.

We therefore suggest that future work should focus on the datasets instead of the models. More work is needed to compare the linguistic features indicative of different kinds of hate speech (racism, sexism, personal attacks etc.), and the differences between hateful and merely offensive speech.

The effectiveness of our simple attacks is indicative of the vulnerability of proposed hate speech classifiers based on state-of-the-art machine learning approaches. Future work should take such attacks into consideration in addition to mere classification accuracy. In particular, we demonstrated the superiority of character-models against attacks, which provides a significant case in favor of using them in real-world applications.

The appending attack presents a fundamental problem with treating hate speech detection as classification. The classes are asymmetrical in that ordinary speech can be transformed into hate speech by adding hateful material, but hate speech should not be transformed into ordinary speech by adding benign material. This asymmetricity is not built into classification, but it should be a foundational principle of hate speech detection. We recommend focusing on this problem in future research, and seeking detection methods that are not based on mere classification. One possibility is to reintroduce more traditional keyword-based approaches, where only hate-indicative words are sought, disregarding the presence or absence of other words.

Additionally, building on our adversarial training experiments, we suggest training data augmentation as a method to help classification remain more resistant against appending attacks. This is a well-known approach to making classifiers more resistant to noise. Adding benign text to hate speech datapoints helps the classifier find those aspects that are relevant for the text belonging to the hate class, and decreases the effect of irrelevant word-class correlations.

In summary, we make four recommendations for future work on hate speech detection:

- Focus should be on the datasets instead of the models, and more qualitative work is needed to understand different categories the fall under the umbrella of "hate speech".
- Simple character-models are preferable to word-based models (including DNNs) with respect to resisting simple evasion methods based on text transformation.
- Detection methods should not be vulnerable to the asymmetricity between the classes, which invites using methods that only target the presence of hate-indicative features and remain indifferent to other features.
- Training data augmentation can reduce the effect of benign words on classification.

REFERENCES

[1] Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion* (2017), pp. 759–760.

[2] Brennan, M., Afroz, S., and Greenstadt, R. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security 15*, 3 (2011), 12:1–12:22.

[3] Brennan, M., and Greenstadt, R. Practical Attacks Against Authorship Recognition Techniques. In *Proceedings of the Twenty-First Conference on Innovative Applications of Artificial Intelligence* (2009), K. Haigh and N. Rychtyckyj, Eds., pp. 60–65.

[4] Brown, A. What is hate speech? Part1: The myth of hate. *Law and Philosophy 36*, 4 (2017), 419–468.

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

[6] Chen, Y., Zhou, Y., Zhu, S., and Xu, H. Detecting offensive language in social media to protect adolescent online safety. In *Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust and of the 2012 International Conference on Social Computing, PAS-SAT/SocialCom '12* (Amsterdam, 2012), pp. 71–80.

[7] Davidson, T., Warmslay, D., Macy, M., and Weber, I. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th Conference on Web and Social Media* (2017), pp. 512–515.

[8] Dinakar, K., Jones, B., Havasi, C., Lieberman, H., and Picard, R. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Transactions on Interactive Intelligen Systems 2*, 3 (2012), 18:1–18:30.

[9] Gitari, N. D., Zuping, Z., Damien, H., and Long, J. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering 10*, 4 (2015), 215–230.

[10] Goldberg, Y. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research 57*, 1 (2016), 345–420.

[11] Hosseini, H., Kannan, S., Zhang, B., and Poovendran, R. Deceiving Google's Perspective API Built for Detecting Toxic Comments. *CoRR abs/1702.08138* (2017).

[12] Howard, J., and Ruder, S. Fine-tuned language models for text classification. *CoRR abs/1801.06146* (2018).

[13] Kumar, S., and Shah, N. False information on web and social media: A survey. *CoRR abs/1804.08559* (2018).

[14] Lowd, D., and Meek, C. Good word attacks on statistical spam filters. In *CEAS* (2005).

[15] Marpaung, J., Sain, M., and Lee, H.-J. Survey on malware evasion techniques: State of the art and challenges. In *14th International Conference on Advanced Communication Technology* (2012), pp. 744–749.

[16] Mehdad, Y., and Tetreault, J. Do characters abuse more than words? In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2016), pp. 299–303.

[17] Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer Sentinel Mixture Models. In *Proceedings of the International Conference on Learning Representations* (2017).

[18] Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Team, T. G. B., Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M. A., and Aiden, E. L. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science 6014*, 331 (2011), 176–182.

[19] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints* (2013).

[20] Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*.

[21] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1532–1543.

[22] Perea, M., nabeitia, J. A. D., and Carreiras, M. R34D1NG W0RD5 W1TH NUMB3R5. *Journal of Experimental Psychology: Human Perception and Performance 34* (2008), 237–241.

[23] Rayner, K., White, S., Johnson, R., and Liversedge, S. Raeding wrods with jubmled lettres: there is a cost. *Psychological Science 17*, 3 (2006), 192–193.

[24] Schmidt, A., and Wiegand, M. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media* (2017), pp. 1–10.

[25] Stern, H., Mason, J., and Shepherd, M. A linguistics-based attack on personalised statistical e-mail classifiers. Tech. rep., Dalhousie University, 2004.

[26] Warner, W., and Hirschberg, J. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media, LSM fi12* (2012), pp. 19–26.

[27] Waseem, Z., and Hovy, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop* (2016), pp. 88–93.

[28] Wulczyn, E., Thain, N., and Dixon, L. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web* (2017), pp. 1391–1399.

[29] Zhang, Z., Robinson, D., and Tepper, J. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Proceedings of ESWC* (2018), pp. 745–760.

[30] Zhou, Y., Jorgensen, Z., and Inge, W. M. Combating good word attacks on statistical spam filters with multiple instance learning. *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007) 2* (2007), 298–305.

Appendix B

Media Coverage

The paper has a good media coverage, such as Wired (https://www.wired.com/story/break-hate-speech-algorithm-try-love/), or Repubblica (https://www.repubblica.it/tecnologia/social-network/2018/09/12/news/odio_online_che_disastro_i_filtri_automatici_basta_una_lettera_per_ingannarli-206266171/).

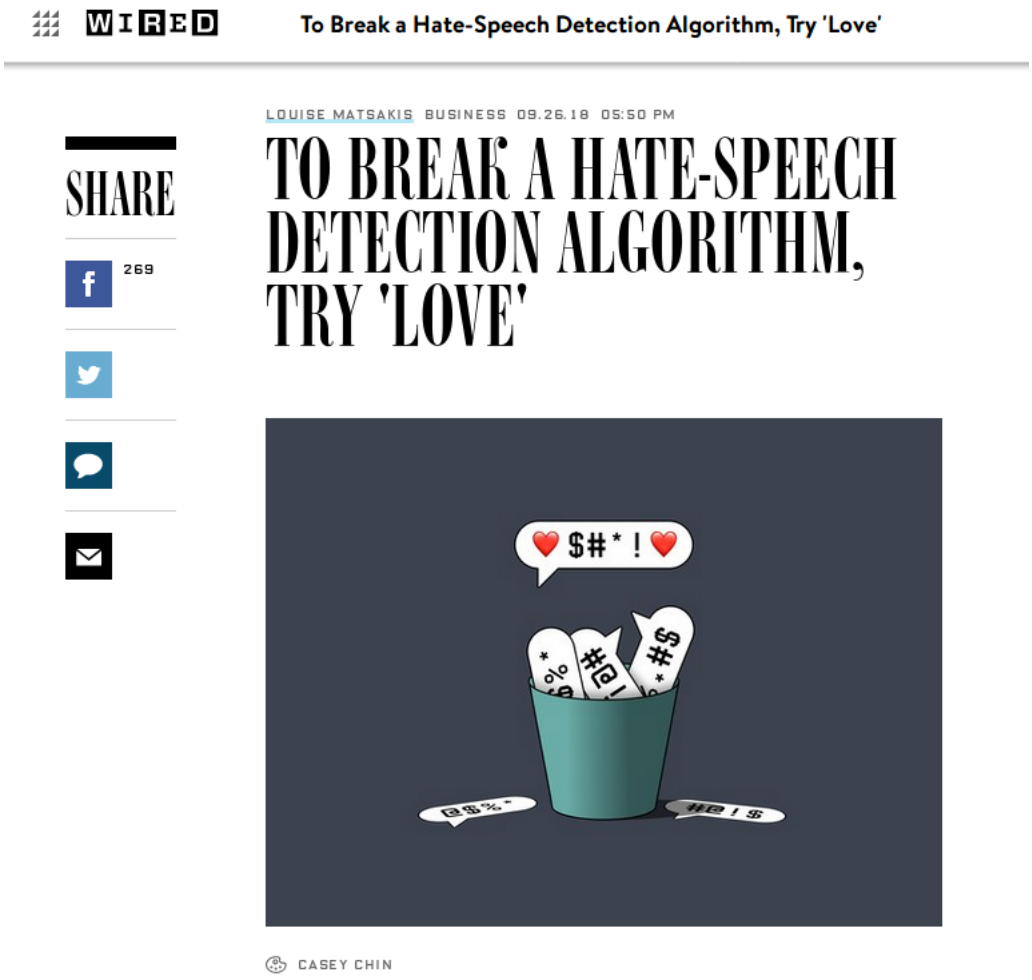


FIGURE B.1: Wired - 26.9.18

The updated and full list is available in https://ssg.aalto.fi/research/projects/deception-detection-via-text-analysis/results/.

R.it | Social Network

Home | News | Speciali | Mobile | Social Network | Sicurezza | Prodotti | Interattivi | Video

97

Odio online, i filtri automatici? Un disastro: basta una lettera per ingannarli

Lo dimostra un'analisi condotta su sette programmi, tra cui quello utilizzato da Google: i sistemi funzionano bene solo sulla stessa banca dati su cui sono stati allenati

FIGURE B.2: Repubblica - 12.9.18

Bibliography

Badjatiya, Pinkesh et al. (2017). "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, pp. 759–760. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3054223. URL: https://doi.org/10.1145/3041021.3054223.

Biggio, Battista et al. (2013). "Evasion Attacks against Machine Learning at Test Time". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Hendrik Blockeel et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 387–402.

Bird Steven, Edward Loper and Ewan Klein (2009). "Natural Language Processing with Python". In: *O'Reilly Media Inc*.

Blum, Avrim L. and Ronald L. Rivest (1992). "Training a 3-node neural network is NP-complete". In: *Neural Networks* 5.1, pp. 117 –127. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80010-3. URL: http://www.sciencedirect.com/science/article/pii/S0893608005800103.

Brennan, Michael, Sadia Afroz, and Rachel Greenstadt (2011). "Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity". In: *ACM Transactions on Information and System Security* 15.3, 12:1–12:22.

Cho, Kyunghyun et al. (2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: http://www.aclweb.org/anthology/W14-4012.

Chollet, François, et al. (2015). *Keras*. URL: https://keras.io.

Chung, Junyoung et al. (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In:

Davidson, Thomas et al. (2017). "Automated Hate Speech Detection and the Problem of Offensive Language". In: *11th Conference on Web and Social Media* abs/1703.04009. arXiv: 1703.04009. URL: http://arxiv.org/abs/1703.04009.

ECHR (1976). *Case of Handyside v. The United Kingdom*. http://hudoc.echr.coe.int/eng?i=001-57499.

— (2006). *Case of Erbakan v. Turkey*. http://hudoc.echr.coe.int/eng?i=001-76234.

Fox, Jesse, Carlos Cruz, and Ji Young Lee (2015). "Perpetuating online sexism offline: Anonymity, interactivity, and the effects of sexist hashtags on social media". In: *Computers in Human Behavior* 52, pp. 436 –442. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2015.06.024. URL: http://www.sciencedirect.com/science/article/pii/S0747563215004641.

Gareth James Daniela Witten, Trevor Hastie Robert Tibshirani (2015). *An Introduction to Statistical Learning. with Applications in R*. Springer.

Gers, Felix A., Jürgen A. Schmidhuber, and Fred A. Cummins (2000). "Learning to Forget: Continual Prediction with LSTM". In: *Neural Comput.* 12.10, pp. 2451–2471. ISSN: 0899-7667. DOI: 10.1162/089976600300015015. URL: http://dx.doi.org/10.1162/089976600300015015.

Grave, Edouard, Armand Joulin, and Nicolas Usunier (2016). "Improving Neural Language Models with a Continuous Cache". In:

Gröndahl, T. et al. (2018). "All You Need is "Love": Evading Hate-speech Detection". In: Toronto, Canada: the 11th ACM Workshop on Artificial Intelligence and Security (ACM CCS 2018 workshop: AISec 2018).

Grosse, Kathrin, Michael T. Smith, and Michael Backes (2018). "Killing Three Birds with one Gaussian Process: Analyzing Attack Vectors on Classification". In:

Hinduja, S. and J. W. Patchin (2010). "Bullying, Cyberbullying, and Suicide". In: pp. 206–221.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-term Memory". In: 9, pp. 1735–80.

Hosseini, Hossein et al. (2017). "Deceiving Google's Perspective API Built for Detecting Toxic Comments". In:

Hosseinmardi, Homa et al. (2015). "Detection of Cyberbullying Incidents on the Instagram Social Network". In:

Human Rights, European Court of. *European Court of Human Rights*. URL: https://www.echr.coe.int/Documents/Convention_ENG.pdf.

— (2018). "Hate Speech". In:

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Jay, Timothy (2009). "The Utility and Ubiquity of Taboo Words". In: *Perspectives on Psychological Science* 4.2. PMID: 26158942, pp. 153–161. DOI: 10.1111/j.1745-6924.2009.01115.x. eprint: https://doi.org/10.1111/j.1745-6924.2009.01115.x. URL: https://doi.org/10.1111/j.1745-6924.2009.01115.x.

Joyce, James M. (2011). "Kullback-Leibler Divergence". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327. URL: https://doi.org/10.1007/978-3-642-04898-2_327.

Judd, J. Stephen (1990). *Neural Network Design and the Complexity of Learning*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-10045-2.

Kingma, Diederik and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In:

Korde, Vandana (2012). "Text Classification and Classifiers:A Survey". In: 3, pp. 85–99.

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio (2016). "Adversarial Machine Learning at Scale". In: *CoRR* abs/1611.01236.

L. Breiman J. Friedman, R. Olshen and C. Stone (1984). "Classification and Regression Trees". In:

Levy, L.W., K.L. Karst, and A. Winkler (2000). *Encyclopedia of the American Constitution*. Macmillan library reference USA v. 2. Macmillan Reference USA. ISBN: 9780028648804. URL: https://books.google.fi/books?id=lDMOAQAAMAAJ.

Mehl, Matthias R. and James W. Pennebaker (2003). "The Sounds of Social Life: A Psychometric Analysis of Students' Daily Social Environments and Natural Conversations". English (US). In: *Journal of Personality and Social Psychology* 84.4, pp. 857–870. ISSN: 0022-3514.

Merity, Stephen et al. (2016). "Pointer Sentinel Mixture Models". In: *CoRR* abs/1609.07843. arXiv: 1609.07843. URL: http://arxiv.org/abs/1609.07843.

Michel, Jean-Baptiste et al. (2011). "Quantitative Analysis of Culture Using Millions of Digitized Books". In: *Science* 331.6014, pp. 176–182. ISSN: 0036-8075. DOI: 10.1126/science.1199644. eprint: http://science.sciencemag.org/content/331/6014/176.full.pdf. URL: http://science.sciencemag.org/content/331/6014/176.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

Olweus, D. (1993). *Bullying at School: What We Know and What We Can Do*. Understanding Children's Worlds. Wiley. ISBN: 9780631192411. URL: https://books.google.fi/books?id=4qNLY13mkDEC.

Pan, Sinno Jialin and Qiang Yang (2010). "A Survey on Transfer Learning". In: *IEEE Trans. on Knowl. and Data Eng.* 22.10, pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. URL: http://dx.doi.org/10.1109/TKDE.2009.191.

Paszke, Adam et al. (2017). "Automatic differentiation in PyTorch". In:

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Perea, Manuel, Jon Andoni Duñabeitia, and Manuel Carreiras (2008). "R34d1ng w0rd5 w1th numb3r5." In: 34, pp. 237–41.

Powers, D. M. W. (2011). "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation". In: *Journal of Machine Learning Technologies* 2.1, pp. 37–63.

Rayner, Keith et al. (2006). "Raeding Wrods With Jubmled Lettres: There Is a Cost". In: 17, pp. 192–3.

Razavi, Amir H. et al. (2010). "Offensive Language Detection Using Multi-level Classification". In: *Advances in Artificial Intelligence*. Ed. by Atefeh Farzindar and Vlado Kešelj. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 16–27.

Richardson, Alice (2011). "Logistic Regression: A Self-Learning Text, Third Edition by David G. Kleinbaum, Mitchel Klein". In: 79, pp. 296–296.

Schmidt, Anna and Michael Wiegand (2017). "A Survey on Hate Speech Detection using Natural Language Processing". In: *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. Valencia, Spain: Association for Computational Linguistics, pp. 1–10. URL: http://aclweb.org/anthology/W17-1101.

Smith, Peter K. et al. (2008). "Cyberbullying: its nature and impact in secondary school pupils." In: *Journal of child psychology and psychiatry, and allied disciplines* 49 4, pp. 376–85.

Sood, Sara, Judd Antin, and Elizabeth Churchill (2012). "Profanity Use in Online Communities". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, pp. 1481–1490. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208610. URL: http://doi.acm.org/10.1145/2207676.2208610.

Sood, Sara Owsley, Elizabeth F. Churchill, and Judd Antin. "Automatic identification of personal insults on social news sites". In: *Journal of the American Society for Information Science and Technology* 63.2, pp. 270–285. DOI: 10.1002/asi.21690. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.21690. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21690.

Spertus, Ellen (1997). "Smokey: Automatic Recognition of Hostile Messages". In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth*

Conference on Innovative Applications of Artificial Intelligence. AAAI'97/IAAI'97. Providence, Rhode Island: AAAI Press, pp. 1058–1065. ISBN: 0-262-51095-2. URL: http://dl.acm.org/citation.cfm?id=1867406.1867616.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: 15, pp. 1929–1958.

Stern, Henry, Justin Mason, and Michael Shepherd (2004). *A linguistics-based attack on personalised statistical e-mail classifiers*. Tech. rep. Dalhousie University.

T. Beck, Aaron, Maria Kovacs, and Arlene Weissman (1979). "Assessment of suicidal intention: The Scale of Suicide Ideation". In: 47, pp. 343–52.

Van Hee, Cynthia et al. (2015). "Detection and fine-grained classification of cyberbullying events". eng. In: *Proceedings of Recent Advances in Natural Language Processing, Proceedings*. Ed. by Galia Angelova, Kalina Bontcheva, and Ruslan Mitkov. Hissar, Bulgaria, pp. 672–680.

Warner, William and Julia Hirschberg (2012a). "Detecting Hate Speech on the World Wide Web". In: *Proceedings of the Second Workshop on Language in Social Media*. LSM '12. Montreal, Canada: Association for Computational Linguistics, pp. 19–26. URL: http://dl.acm.org/citation.cfm?id=2390374.2390377.

— (2012b). "Detecting Hate Speech on the World Wide Web". In: *Proceedings of the Second Workshop on Language in Social Media*. LSM '12. Montreal, Canada: Association for Computational Linguistics, pp. 19–26. URL: http://dl.acm.org/citation.cfm?id=2390374.2390377.

Whittaker, Elizabeth and Robin Kowalski (2014). "Cyberbullying Via Social Media". In: 14, pp. 11–29.

Wu, Xindong et al. (2008). "Top 10 algorithms in data mining". In: *Knowledge and Information Systems* 14.1, pp. 1–37. ISSN: 0219-3116. DOI: 10.1007/s10115-007-0114-2. URL: https://doi.org/10.1007/s10115-007-0114-2.

Wulczyn Ellery, Thain Nithum Dixon Lucas (2017). "Ex Machina: Personal Attacks Seen at Scale". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, pp. 1391–1399. ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052591. URL: https://doi.org/10.1145/3038912.3052591.

YANG, QIANG and XINDONG WU (2006). "10 CHALLENGING PROBLEMS IN DATA MINING RESEARCH". In: *International Journal of Information Technology & Decision Making* 05.04, pp. 597–604. DOI: 10.1142/S0219622006002258. eprint: https://doi.org/10.1142/S0219622006002258. URL: https://doi.org/10.1142/S0219622006002258.

Zeerak, W. and D. Hovy (2016). "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter". In: *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, pp. 88–93. URL: http://www.aclweb.org/anthology/N16-2013.

Zhang Ziqi Robinson David, Tepper Jonathan (2018). "Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network". In: *The Semantic Web*. Cham: Springer International Publishing, pp. 745–760.

Zhong, Haoti et al. (2016). "Content-driven Detection of Cyberbullying on the Instagram Social Network". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA: AAAI Press, pp. 3952–3958. ISBN: 978-1-57735-770-4. URL: http://dl.acm.org/citation.cfm?id=3061053.3061172.

Zhu, Ji et al. (2006). "Multi-class AdaBoost". In: 2.