Using Machine Learning and Deep Learning Methods to Detect Cyberbullying in Messages

Student Name: Jack Leyland

Supervisor Name: Dr Noura Al Moubayed

Submitted as part of the degree of BSc Computer Science (with Year Abroad) to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

Context/Background –
Aims –
Method –
Results –
Conclusions –

Keywords —

I INTRODUCTION - 2/3 PAGES

A Context

The world is becoming increasingly technological, and with this, social media, online content and cyberbullying is becoming more prevalent than ever before.

As of 2017, almost 25% of 8-11-year-olds have a social media profile, and a huge 75% of 12-15-year-olds do (Ofcom 2017, p.5). 45% of these 12-15-year-olds say that they have seen hateful content online in the last 12 months - which is an increase on 2016's figures.

Worryingly, 12% of 12-15-year-olds have said that they have been bullied on social media. This means that cyberbullying is now as common as face-to-face bullying.

Due to its nature, cyberbullying is much harder to see and detect than physical bullying. As a result, we are reliant on manual reporting features provided by social media sites - with 75% of 12-15 year olds saying they are aware of these features (Ofcom 2017, p.5) and only 12.5% having used them. This highlights the scope for automatic cyberbullying detection in messages and the potential for Machine Learning and Deep Learning to achieve this.

This task is inherently difficult even for a human to accomplish; whether we classify something as cyberbullying or not is relatively subjective. The same meaning/semantics can be represented in different ways, and in reality context is important; looking at messages in isolation often doesn't provide enough information to make well-informed classifications. Furthermore, there are many different types of cyberbullying, such as racism, sexism, aggression etc. Natural language also provides more challenges than numerical data, since we must find meaningful representations of this information that Machine and Deep Learning models are able to exploit.

B Objectives

The research questions for my project are as follows:-
'What makes a good dataset for this task? How can datasets be pre-processed to ensure the models perform well? Which Deep Learning architectures give the best results for the classification of messages as cyberbullying? Is Deep Learning more appropriate than traditional Machine Learning for this task?'

The minimum objective of this project is to be able to implement a traditional Machine Learning model on one dataset that peforms better than the trivial classifier. The trivial classifier is one which either predicts randomly (would achieve 50% accuracy in a binary classification task) or makes the same prediction each time (for example, this could achieve 80% acuracy in a task where 80% of instances belong to one class). Surpassing this benchmark indicates that a model has learnt something about the underlying data which is being modelled. A prerequistie of this is the exploration of datasets; I will research into existing work around this topic, identify the current state of the art performance and investigate the datasets used if made publicly available, weighing up their suitability for this project. Furthermore, a method of evaluation will be established including F1 score, enabling comparison to previous work.

The intermediate objectives are to experiment with Deep Learning architectures and surpass the performance of the traditional Machine Learning methods. These models will be thoroughly evaluated, using F1 score. Also, training curves will be generated, providing insight into whether a given model is overfitting, underfitting, etc. These, combined, will allow for a conclusion as to whether Deep Learning performance reliably exceeds that of Machine Learning. The ability to save and re-load models for later use will be implemented, allowing us to recreate results and further train models which may be underfitting.

Finally, the advanced objectives for the project will be to extend this experimentation into numerous datasets. Results will be compared, providing an insight into the influence of the underlying data in Machine Learning and Deep Learning tasks. A wide range of architectures will be implemented and experimented, including state of the art technologies where existing research is limited such as ELMo deep contextualised word representations. Putting all of this together will hopefully, ultimately, result in results which equal or beat the state of the art performance for this task.

Conversely, there are a number of non-functional requirements associated with this task. For example, once completed, the code will be made public on *github.com* to aid further research into this field. The code must also be modular, scalable and maintainable, to increase efficiency for those who may adapt this code in the future. A comprehensive README will be provided to aid navigation of the wealth of files associated with the project.

C Achievements

TODO:

II RELATED WORK - 2-4 PAGES

Existing work into this field, datasets, and this project refer to the terms 'cyberbullying', 'racism', 'sexism', and 'neither racism or sexism', which are entirely subjective. Thus, these are defined below, with genuine examples from datasets used in this project.

Table 1: Definitions of subjective terms

Term	Definition	Example
Cyberbullying	A message where an individual or group uses the Internet to ridicule, harass or harm another person.	*Dear Mav: Thanks, bro! God bless you! Sincerely yours, **|
Racism	A message demonstrating prejudice or discrimination directed against someone of a different race based on the belief that one's own race is superior.	*@Lithobolos @PoliticalAnt @ZaibatsuNews So when are you going to admit that the Quran is wrong. I'm waiting.*
Sexism	A message demonstrating prejudice, stereotyping, or discrimination, on the basis of sex.	*RT @TheBigKahuna12 I'm not sexist, but I'm just not a fan of all these women rappers.*
Neither Racism nor Sexism	From the 3-class dataset. A message that falls under neither the racism or sexism category.	*you are right there are issues but banning Muslims from entering doesnt solve anything.*

A Machine Learning

There have been attempts in the past to use Machine and Deep Learning for detecting cyberbullying and hateful speech. For example, Reynolds and Kontostathis (2011, p.4) attempted to solve a binary classification task, either cyberbullying or not cyberbullying. This endeavour achieved 67% recall still meant that approximately 1/3 of bullying examples went undetected. Three things inspired me from Reynold and Kontostathis' (2011) work:

Firstly, they stress the importance of pre-processing the data. We cannot simply input text into machine learning models. Fixed length feature vectors must be generated for each instance and there are many ways to do this. Reynolds and Kontostathis opted for counting swear words in text, the density of swear words, and the number of swear words at numerous severity levels. Secondly, repeating positive cyberbullying examples in the training dataset (if the dataset is imbalanced) gives the model more motivation to learn to detect positive examples, increasing performance. Finally, I will evaluate my model based on precision/recall/F1 score instead of overall accuracy, as it's most important that I detect the positive examples and this is especially prevalent in imbalance datasets.

Furthermore, (Dixon 2018) has provided code accompanying chapters from Online Harassment by Golbeck (2018). Logistic Regression and Gradient Boosted Classifiers are applied to reddit comments, achieveing up to 0.61 F1 score. This dataset is the largest publically available cyberbullying dataset with approximately 70,000 instances.

Additionally, Bastidas et al. (2016, p.2/3) used a slightly different method. They tokenised the text (broke it into words), hashed resulting unigrams, bigrams and trigrams, and computed

TF-IDF for each hashed value. This provides another potential method for pre-processing data to extract features in my work. Here, using these features along with Gradient-Boosted decision trees, 0.8 precision, 0.71 recall and 0.75 F1 score was achieved.

Another notable paper with regards to this topic was from (Chatzakou et al. 2017). Here, a *slightly* different task is presented. Groups of tweets are sourced which belong to one user and then this *group* of tweets are labelled as bullying, spamming, agressive or neither. 89.9% precision and 91.7% recall were achieved with a Random Forest model, but my results are not directly comparable to this work since the application is different. One thing I have learnt from this paper, however, is the importance of a good dataset. There are 1.6M tweets, each enhanced with user profile information gathered through the Twitter API. This vast, enriched dataset allowed them to achieve high-performance results and I hope to do the same. Unfortunately, the dataset wasn't made public and thus wasn't directly applicable to this project.

B Deep Learning

Bastidas et al. (2016, p.3) also used some Deep Learning methods, however these actually gave worse results than their traditional Machine Learning methods. This shows that I need to represent my data appropriately to get the most out of these Deep Learning methods.

For example, Badjatiya et al. (Badjatiya et al. 2017) tried many architectures. It is possible to use Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) on a number of input features (Bag of Word Vectors (BoWV), Pre-trained word embeddings and character n-grams). I will consider many of these approaches in my project, evaluating and comparing as many different architectures as possible in an endeavour to find the best combination.

Badjatiya et al. (Badjatiya et al. 2017) attempted to solve a 3-class problem; individual tweets were classified as sexist, racist or neither sexcist or racist. Here, 0.93 weighted macro Recall, Precision and F1 score were achieved, by randomly initialising the input word embeddings, applying a LSTM and running the output through a Gradient Boosted Decision Tree classifier.

In conclusion, many have tried to apply traditional Machine Learning methods within the context of cyberbullying to varying levels of success, and Deep Learning applications in this field are relatively limited - with the exception of (Badjatiya et al. 2017), achieving impressive results in the aforementioned 3-class problem.

Therefore 0.93 weighted macro F1 score provides the **state of the art** figure that will be used to evaluate the overall quality of my achievements on the 3-class problem at the close of the project. For the 2-class problem, I will refer to 0.75 as the **state of the art** (Bastidas et al. 2016). However, I must consider that the exact results achieved in this project are likely to vary depending on the dataset being used.

C Datasets

Some of the aforementioned papers used publically available datasets. If so, these datasets were analysed and their suitability evaluated. The main factor was the strucutre of the data, ideally containing a list of instances (tweets/comments/etc.) with associated labels (ideally binary, cyberbullying or not cyberbullying). Three suitable datasets were found for use in this project.

The **Reddit** dataset came from Online Harassment (Golbeck 2018), with the accompanying F1 score coming from Dixon's work (2018) on this dataset[1]. This is a vast dataset containg

[1]https://github.com/EdwardDixon/Automation-and-Harassment-Detection

Table 2: Public datasets used in this project

Dataset	Structure	#examples	+ve instances	Avg. length	SOTA
Reddit	Text and binary label	69526	50.0%	593.0 words	0.61 F1
Twitter_small	Text and binary label	1066	40.1%	15.6 words	N/A
Twitter_big_ 3class	Text and 3-class label (racism, sexism, or neither)	16049	12.3% racist 19.7% sexist	28.1 words	0.93 (weighted macro F1)
Twitter_big_ 2class	Text and binary label (racism and sexism combined)	16049	32.0%	28.1 words	N/A

approximately 70,000 instances of average length 593 words, containing a huge amount of 'dirt' in the form of misspellings and excessive punctuation. The comments are not hand-tagged by humans, instead they are automatically assigned an attack value based on the frequency of certain swear words in the messages. This limits the quality of the dataset, but it will be interesting to see if our models can exploit this and perform particularly well.

The **Twitter_small** dataset came from a public online Git repository[2] and there are no published results accompanying this data. However, the data has already been cleaned, containing no hashtags, retweets tags or mentions. The only limitation to this dataset is that there are a small number of examples (only 1066) and they are short (15.6 words on average), meaning that models might overfit to this data and show poor performance when generalising to any unseen data.

Twitter_big_3class is the public dataset[3] originally created by Waseem et al. (2016) and used by Badjatiya et al.(2017) to achieve 0.930 weighted macro F1. F1 is only defined on 2 classes, thus, the 3-class variant weighted macro F1 is used to evaluate the performance on this dataset. 17K tweets were collected containing words and hashtags often present in racist/sexist tweets, so those tweets which fall under neither category contain such words - but are non-discriminatory in their use.

Furthermore, I will experiment with combining the racism and sexism classes in this dataset and approaching this as a binary task. I will hereby refer to this as the **Twitter_big_2class** dataset, where 32% of instances are positive. There are a reasonable number of examples but the short average length (28.1 words) could limit the amount of information we can extract from each tweet and thus make this a more difficult dataset to work with. Furthermore, since this is a raw Twitter dataset, there are a wealth of retweet tags, mentions, hashtags which add noise to the dataset; their removal must be considered.

III SOLUTION - 4/7 PAGES

All source code for this project is available on GitHub.[4]

[2]https://github.com/chantelmariediaz/Predicting-Cyberbulling-on-Twitter

[3]https://github.com/zeerakw/hatespeech

[4]https://github.com/jleyland96/Cyberbullying-Detection

A Tools Used

The nature of this task lead towards spending the majority of time in the implementation phase. There was research to be done regarding existing work and theory behind Machine and Deep Learning, however, most of the work centres around integrating the datasets, data pre-processing, creating models, evaluating their quality for cyberbullying detection, etc.

As a result, I opted for a language with a wealth of support for these tasks; allowing me to get into the implementation part of my project as soon as possible. I am developing my system in **Python** since the documentation is plentiful, the code is easy to read, and there exists frameworks and libraries for implementation of my models, and a number of forums.

For example, Python library **Sci-kit Learn** (Sklearn) provides a number of functions for easy data processing and machine learning model creation. This allows me to focus on the implementation of models, instead of any low-level mathematical details. Furthermore, **Keras** is suitable for Deep Learning approaches, providing more data pre-processing functions and support for a number of architectures such as Recurrent and Convolutional Neural Networks, Dense Networks, etc.

Additionally, Python libraries such as **NumPy** (for mathematical operations) and **CSV** (for reading in the datasets) will be used.

B Pre-processing the data

Each dataset contains a considerable amount of dirt. All contain excessive punctuation and inconsistent capitalisation which adds noise and would make it harder for the models to make informed decisions on unseen data. For example, we would want a training example containing *'Can't'* to inform decisions on a testing example containing *'cant'*. As a result, I simply removed all punctuation and converted all text to lower case in each dataset.

The **Twitter 1K** dataset has been cleaned for us, however the **Twitter 16K** dataset contains raw tweets with a number of hashtags, RT tags, mentions and URLs. I removed RT tags, mentions and URLs as I believe that these have no bearing on if a given instance is cyberbullying, but I left in hashtags since these are meaningful pieces of text which could be cyberbullying, e.g. *#deathtokat*.

One measure of the cleanliness of the data is the number of words in our dataset that have an associated GloVe representation. This is a method of feature extraction (see section below) which converts words in our data to vectors. The more of our data that has an associated vector, the more information we have as inputs to our models and the better the models are likely to perform. Figure 3 shows how this cleaning method greatly increased the percentage of our input words that have an associated GloVe representation.

Table 3: Input words with GloVe representations, before and after data cleaning.

Dataset	GloVe hit % before cleaning	GloVe hit % after cleaning
Reddit	65.6	95.5
Twitter1K	64.9	92.7
Twitter16K	59.4	88.4

Finally, in inbalanced datasets such as the **Twitter_small** dataset where only 28% of examples

are cyberbullying, repeating the positive examples in the dataset gives the models more motivation to fit the positive examples and might improve performance (Reynolds & Kontostathis 2011). This forms another part of the experimentation of this project.

C Feature Extraction

Traditional Machine Learning models require fixed length numerical vectors as inputs. Therefore, the input to the functions provided by Sci-kit Learn cannot simply be the text (of varying length). This textual data must be represented numerically in some way and then used as the input features.

This project consists of experimentation with 6 methods of feature extraction.

- **GloVe —** Global vectors for word representation (Pennington et al. 2014). These are pre-trained vectors that represent many words in the English language. They are pre-trained on a huge dataset and the motivation for using them is that similar words ('cat' and 'dog') have similar vectors, different words ('cheese' and 'running') have different vectors.

  Concatenating all vectors for each word in a message could be effective. The motivation is that the resulting vectors of cyberbullying instances may be similar as they could contain similar words.

  GloVe vectors have 300 elements for each word, therefore our feature vectors are of length $300\times$ the number of words we pad/truncate our input sentences to. If the feature space is too large (observed by particularly poor F1 score), we could average all vectors corresponding to all words in a given instance, fixing our feature vectors to length 300.

- **TF (Term Frequency) —** Each message is represented as a vector. Each vector element represents a word that appears in any document (the dataset). The value is the number of occurences of this word in this document (text message) divided by length of message. The motivation behind this (and the following 2 methods) is that there may be sets of words that occur more often in cyberbullying examples.

- **TF-IDF (Term Frequency-Inverse Document Frequency) —** Same as TF, but then each value is divided by the number of times the corresponding word appears in the whole dataset. This reduces the influence of common words in the dataset. (e.g. 'the', 'a', etc.)

- **Term Counts —** Same as TF, except the values are not divided by the length of the document. Simply just the number of occurrences of each word from the dataset in this particular sentence.

- **Character bigrams —** Each element of vector represents a character bigram ('aa', 'ab', 'ac',...,'zz'), and the value is the number of times this bigram appears in this message. The motivation behind this method (and trigrams) is that there may be combinations of characters that occur more often in cyberbullying examples.

- **Character trigrams —** Each element of vector represents a character trigram ('aaa', 'aab', 'aac',...,'zzz'), and the value is the number of times this trigram appears in this message.

D Word Embeddings

When using Deep Learning, input (messages) were represented with Word Embeddings. This is where words are represented as points in an n-dimensional vector space, in a way such that words often used in similar contexts (e.g. 'cat' and 'dog') are close to each other in this space and vice versa (Goodfellow et al. 2016, p.464). This, for example, allows messages containing the word 'cat' to help inform predictions on messages containing 'dog'.

GloVe (Pennington et al. 2014) - as already mentioned - is an example of a general-purpose pre-trained word-embedding. This embedding is applied to the input data and then fixed, training and tuning only the succeeding model that uses these inputs. Alternatively, **not** fixing GloVe vectors would allow them to be trained along with the model, tuning them to fit the given training dataset more than they already do.

Alternatively, word embeddings can be randomly initiated and then trained with the rest of the model to fit the training dataset from scratch. The model will take longer to train; however, the resulting embedding will be more specific to the given dataset and the cyberbullying context and thus *could* lead to better performance.

E ELMo

GloVe is a widely-used method of word representation, but the latest cutting-edge advancements in this field have led to Contextualised WordVectors (CoVe), specifically, Embeddings for Language Models (ELMo) (Peters et al. 2018). This word representation models synatx and semantics, as well as polysemy (how word use varies across linguistic contexts). Peters et al. pre-trained a deep bidirectional LSTM with a coupled language model objective on a large text corpus, and for any given word, its ELMo representation is a function of the entire input sentence in all of the internal layers of the LSTM. They have been used to improve the state of the art in six challenging NLP problems including question answering and sentiment analysis - so they are likely to show high performance in this project.

F Other Data Considerations

The public datasets available for this project are all relatively small (the largest dataset has only 17K tweets), therefore the decision was made to use the testing dataset as the validation dataset during training, with a 90:10 split. This maximised the amount of data that could be used for training, and since the testing data is completely independent of the model during training, it still makes a valid validation set.

The pad-length of the input data must also be considered. Inputs must be of the same length, so short inputs are padded with zeros (which the model learns to ignore) and long inputs are truncated. Obviously, this pad length will vary based on the input dataset. Twitter_small and Twitter_big are padded to length 30 and 32 words respectively, which captures information from the entirety of the vast majority of instances (see Figure 1, showing the distribution of the length of data in each dataset). Conversely, in the Reddit dataset, pad length 500 was used, to capture as much information as required but not exceed the computational resources available.

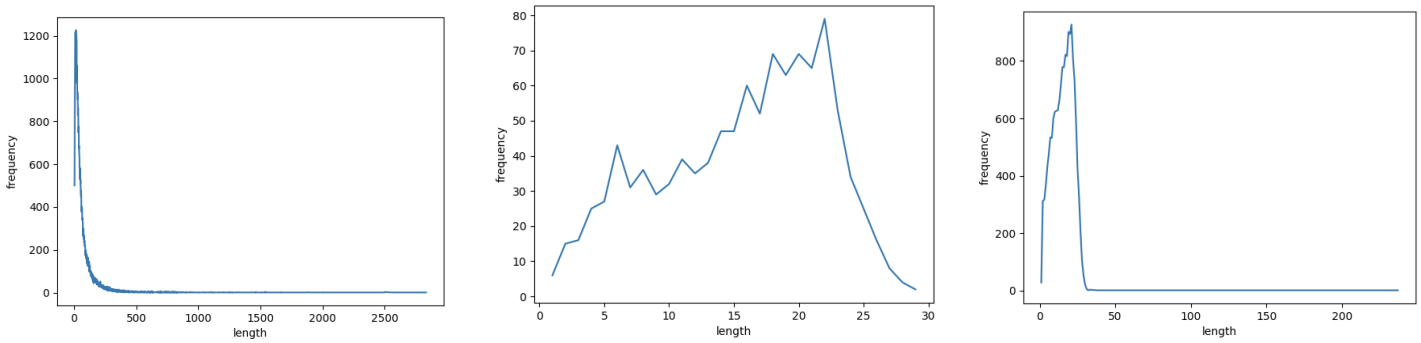

Figure 1: Length distribution of data: **Left**: Reddit, **Centre**: Twitter_small, **Right**: Twitter_big

G Machine Learning methods

Experimenting with traditional Machine Learning approaches provides a baseline system performance. This gives motivation for experimenting with Deep Learning, and it becomes possible to conclude if Deep Learning is more appropriate for detecting cyberbullying in messages than traditional Machine Learning.

Machine Learning models used in this project (and the ideas behind them) are:

- **Naive Bayes —** the features are represented as length n vectors, $x = (x_1, x_2, ..., x_n)$, and for each instance, $P(Bullying|x)$ is calculated using Bayesian statistics. 'Naive' comes from the fact that this assumes the independence of each variable in the feature vectors.

- **Logistic Regression —** Here, a function is defined over each variable in the feature vectors (assuming they are independent) and ultimately provides an output between 0 and 1 which represents the probability that this instance is an example of cyberbullying or not.

- **SVM —** Each instance is represented as an n-dimensional vector point $[x_1, x_2, ..., x_n]$ in an n-dimensional space. The aim is to find a (n-1)-dimensional hyperplane which separates these points, with one class (bullying instances) on one side and the other class (non-bullying instances) on the other side. Ideally, a hyperplane is found which separates the two classes effectively, but also doesn't over-fit the training data, which would reduce generalisation performance.

- **Gradient-Boosted Classifier —** A method of producing numerous weak prediction models, and combining them into an ensemble to make one high-performing model in an iterative fashion, minimising a particular loss function by using gradient descent. Over time, instances that are difficult to classify are focused on more to improve the performance of the weak classifiers.

- **K-Nearest-Neighbours —** an **unsupervised** method. Again, each instance in the training set is represented as an n-dimensional vector point $[x_1, x_2, ..., x_n]$. Then to classify an instance of the test set, the classes of the K nearest points (neighbours) to this instance are considered and a majority vote is taken. These are, in essence, the K most similar messages to the given instance.

Each of these models is tested with each possible input feature representations to find the best Machine Learning model. These models are tested with a number of parameter combinations. This isn't an exhaustive parameter search which would take an unlimited amount of time, but a representative set is manually tested which aims to find at least an adequately accurate estimate of the best possible achievable performance with Machine Learning. This project will then experiment with if

H Deep Learning methods

Three different representations of the text data are being used for Deep Learning experimentation: GloVe word embeddings, randomly initiated word embeddings which we will train as part of our model, and ELMo word representations. These are each used in experimentation with a number of Deep Learning architectures, as follows.

H.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are applied to the word embeddings. RNN cells apply the same function (creating a recurrence) to all of the inputs (the embedding vector for each word in the message), while maintaining a hidden internal state that is passed through the recurrence. (Goodfellow et al. 2016, p.376)

This 'state' at any given point is some lossy representation of the past information in the sentence, used to inform the calculations that the model makes at this point. For example, when processing a cyberbullying message, this state could be a notion of context.

The final 'state' of the system is taken and the sigmoid function is applied to get the final output - a number between 0 and 1 representing the probability that the input sentence was a cyberbullying instance or not.

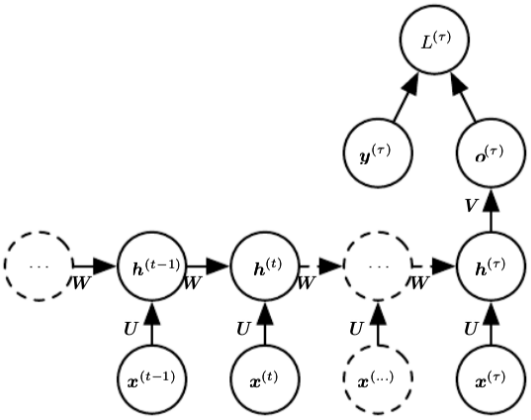

Figure 2: **RNNs.** h**=state,** x**=input,** y**=gold label,** $\sigma(T)$**=predicted label,** t**=time/index**

Specificaly, Long-Short Term Memory (LSTM) cells are used in this project (Goodfellow et al. 2016, p.410). Traditional RNNs suffer from the vanishing gradient problem. In other words, RNNs struggle to learn long-term dependencies in temporal data or text which makes it hard to train them. LSTM units introduce a memory cell, which are self-loops, giving them the ability to maintain information for a long time as they process the input data. They can learn what information they should remember and what information they should forget as they train, which means that they typically show much better performance than traditional RNNs.

Bidirectional LSTMs are also used in this project. These process sequential data the same in a regular LSTM, but both forwards and backwards at the same time. This means that when processing any one word, an LSTM cell can use both past and future information to inform decisions at this timestep. This enriches the amount of information that the LSTM can draw upon and thus could show an improvement in detecting cyberbullying.

H.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) employ the mathematical operation **convolution**. They allow us to process data with structure, (e.g. 2D images, 3D images, 1D sequences) thus they can be applied to text too (Goodfellow et al. 2016, p.330).

Convolutions are invariant to translation, and so this allows us to exploit the feature that the same couple of words would indicate cyberbullying, regardless of where in the sentence they occur. For example, 'you d**k' and 'wow, you d**k' would both be classed as cyberbullying.

A 1D 'kernel' is defined and moved across the text, followed by a 'pooling layer' which essentially converts the input to an output which is smaller by taking the most prominent features in local areas of the text. (Goodfellow et al. 2016, p.339)

CNNs are utilised in 3 ways in this project:

- **Traditional CNN —** Numerous CNN layers and pooling layers are applied, followed by a traditional dense neural network to eventually classify output into my two classes.
- **CNN + LSTM —** One CNN and one Pooling layer is applied to reduce feature size and extract some meaningful features of the text. The output from the pooling layer is applied to an LSTM. Finally, the sigmoid function is applied to the final state of the LSTM cell to classify the sentence.
- **Multi-channel CNN —** Inspired by Kim (2014). The idea is that multiple convolutional layers are applied to the input words in parallel with different kernel sizes. The layer outputs are concatenated, and then classified with a dense neural network. This is therefore extracting information at different n-gram sizes; groups of words. This idea is illustrated in Figure H.2.

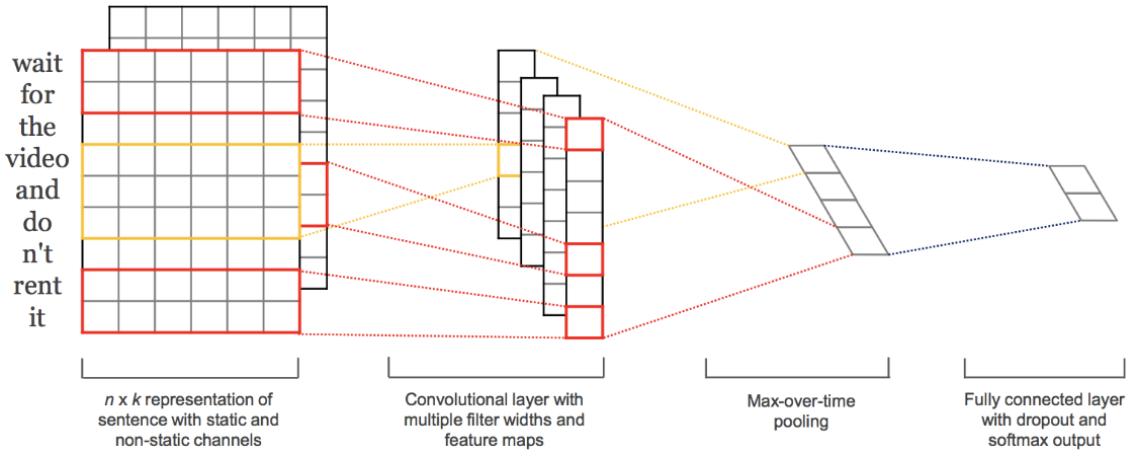

Figure 3: Multi-channel CNN. NOTE: I apply sigmoid for the 2-class problem, not softmax as in this diagram.

H.3 Loss functions

Binary crossentropy is used as the loss function of the models in this project, and the objective in training a model is to minimise this. However, it also seems logical to maximise the metric that

we are using to evaluate our models (F1 score) directly. Therefore, using $(1 - F1)$ as the loss function and minimising this, is synonymous with maximising the F1 score of our model.

This loss function isn't supported by Keras by default, but a custom metric can be designed. F1 must be written using built-in Keras backend functions - this allows the loss function to be differentiable, which is required for backpropagation, thus the network parameters can be shifted accordingly.

In the experimentation phase of this project, both loss functions were used alongside all aforementioned Deep Learning architectures and their performance evaluated.

H.4 Regularization

In any Deep Learning task, the objective is to minimise the generalization error, i.e. to ensure that the model which effectively fits the training data also effectively generalises to unseen test data. There is a danger that overfitting occurs, which is where the training dataset is modelled *too* specifically and so the model demonstrates poor performance on unseen data.

During experimentation in this project, the following methods of regularization are used:

- **Dropout —** On each training batch, each cell of the network is removed (activation is set to zero) with a given probability. The loss is therefore backpropagated through a different sub-network for each batch, meaning that over many batches a bagged ensemble of many neural networks is trained. This provides a computationally cheap way of regularising our model (Goodfellow et al. 2016, p.273). In our RNNs, dropout is applied on the inputs to each cell, however we can apply **recurrent dropout** which is where the connections between the recurrent cells are removed at random for each batch, again with a given probability.

- **Batch Normalization —** The layer where Batch Normalization is applied is reparametrized which normalizes the activations of each cell across a batch. The primary effect of this is to improve optimization but it can also have a regularizing effect on models in practice (Ioffe & Szegedy 2015, p.1).

- **Early-stopping —** When we train a model for a long time, we might find that validation error starts to increase after a model starts to ovetfit to the training data. This means that we are likely to obtain a model with better generalization by saving the parameter setting that achieves the lowest validation set error. When training ceases, we return these parameters, rather than the latest parameters in training.

IV RESULTS - 2/3 PAGES

A F1 score

Once a model has been trained, it is evaluated by observing predictions made on the test set. Throughout this project, F1 score is used as a metric to evaluate the performance of any given Machine Learning or Deep Learning model. Keeping this metric consistent allows meaningful direct comparison between different models.

The following equations define Precision, Recall and F1 score metrics (TP=number of true positives, FP=number of false positives, FN=number of false negatives). These metrics are more

appropriate than accuracy in this project since three of the four datasets contains an imbalanced number of positives and negatives, i.e. most tweets will not be cyberbullying/racism/sexism. Therefore, a high accuracy is achieved if negative is predicted every single time. F1 score ensures that a good balance is achieved; correctly identifying positive examples, but not wrongly identifying cyberbullying too often (predicting 'cyberbullying' every time). As a result, the aim is to maximise F1 score.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F1 = 2(\frac{Precision * Recall}{Precision + Recall}) \quad (3)$$

In the 3-class problem, F1 score isn't defined this way since there isn't a simple notion of 'positive' and 'negative' instances. For the **Twitter_big_3class** problem, SOTA is calculated as a weighted-macro F1 score (Badjatiya et al. 2017). This is synonymous with the micro F1 score provided by sklearn. One method of calculating this is to calculate the F1 score independently for each of the 3 classes, by treating one class at a time as positive and every other instance as negative. The average of the 3 F1 scores is calculated, weighted by the number of instances in that class. In the 3-class problem, the weighted macro F1 score is used.

B Training curves

Deep Learning training is much more complex than traditional Machine Learning. Firstly, there are many more network parameters to tune. Secondly, these models train over many epochs, shifting their weights with each batch to fit the training data. Accuracy, f1 score and loss can all be observed over time to help infer things about the current model architecture.

As a result, the training accuracy, training loss, validation accuracy and f1 score on the validation set were calculated at the end of each epoch when training my Deep Learning models, and plotted on a graph - these are known as training curves. From these, it is inferred where the maximum performance is achieved (which epoch's parameters were saved with early stopping) and if a model is overfitting or underfitting - changes to the model can be made accordingly. For example, in Figure B we can see that the model is overfitting after approximately 60 epochs as the test accuracy starts to go down and the generalization error becomes significant.

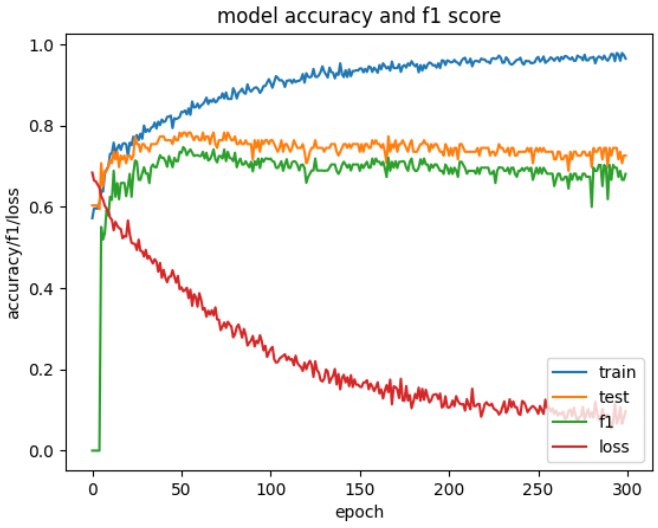

Figure 4: Example training graph, LSTM on **Twitter_small** dataset

C Final results

The tables below show the best results achieved in the 2-class problem with both Machine Learning and Deep Learning methods, for each dataset. The F1 score shown is that achieved through the predictions on the test dataset.

Table 4: Results on **Reddit** dataset

	Model	**F1 score**
Machine Learning	Average GloVe vector + SVM	0.7147
	Term Counts + SVM	0.7288
	TF-IDF + Multinomial Naive Bayes	0.7318
	TF + Logistic Regression	0.7332
	TF-IDF + Logistic Regression	**0.7450**
Deep Learning	GloVe + CNN + LSTM + Dropout	0.7533
	ELMo + LSTM (300 units) + Dropout	0.7604*
	GloVe + LSTM (400 units) + Dropout	0.7620
	GloVe + LSTM (500 units) + Dropout	**0.7645**

* due to the computational resources that a large dataset requires to use ELMo, inputs were padded to length 100 rather than 500 and only 20K of the 69K dataset examples were used.

As you can see....

Table 5: Results on **Twitter_small** dataset

	Model	F1 score
Machine Learning	TF-IDF + Gradient Boosted Classifier	0.7253
	Positives repeated in data + TF + Logistic Regression	0.7253
	Positives repeated in data + Term Counts + Logistic Regression	0.7263
	Average GloVe vector + Gradient Boosted Classifier	0.7342
	Positives repeated in data + TF + Gradient Boosted Classifier	**0.7978**
Deep Learning	ELMo + LSTM (256 units)	0.7784
	Trained word embeddings + LSTM (50 units) + Dropout	0.7789
	GloVe + LSTM (100 units) + Dropout	0.8140
	GloVe + LSTM (150 units) + Dropout	0.8269
	GloVe + Multi-channel CNN + Batch Normalization	**0.8330**

Table 6: Results on **Twitter_big_2class** dataset

	Model	F1 score
Machine Learning	Term Counts + Logistic Regression	0.7405
	Term Counts + Multinomial Naive-Bayes	0.7421
	TF-IDF + Bernoulli Naive-Bayes	0.7424
	TF + Bernoulli Naive-Bayes	**0.7551**
Deep Learning	ELMo + Bidirectional LSTM (512 units) + Dropout	0.7141
	One-hot inputs + Multi-channel CNN	0.7430
	Trainable GloVe + LSTM (100 units) + Dropout	0.7515
	GloVe + LSTM (150 units) + Dropout	0.7645
	GloVe + LSTM (50 units) + Dropout	**0.7716**

From the 2-class problem results, it is clear that our models are performing at quite a high level, beating the identified state of the art 0.75 F1 (Bastidas et al. 2016). However, I must consider that the exact results achieved are dataset dependent, varying approximately 7% between datasets in this project.

D Binary Cross-Entropy Loss vs F1 loss

As mentioned in the Solution, Binary Cross-Entropy is used as the loss function for the majority of experimentation in this project - which means we are minimising the Cross-Entropy between the test labels and test predictions. However, since F1 score is the metric being used to evaluate my models' performance, there is motivation to attempt to maximise the F1 score directly. In practice, $(1 - F1\_score)$ is implemented as a custom less function and then minimised. My best results with each loss function are as follows:

Table 7: Binary Cross-Entropy vs F1 loss function

Dataset	Model	Loss function	F1 score
Reddit	GloVe + LSTM (500 units) + Dropout	Binary Cross-Entropy	0.7645
	XXXX	Custom $(1 - F1)$	0.XXX
Twitter_small	GloVe + Multi-Channel CNN + Batch Normalization	Binary Cross-Entropy	0.8330
	GloVe + Multi-Channel CNN + Batch Normalization	Custom $(1 - F1)$	0.7227
Twitter_big_2class	GloVe + LSTM (50 units) + Dropout	Binary Cross-Entropy	0.7716
	GloVe + Bidirectional LSTM (50 units) + Dropout	Custom $(1 - F1)$	0.7535

In no case does directly maximising F1 increase the model's performance on the test data.

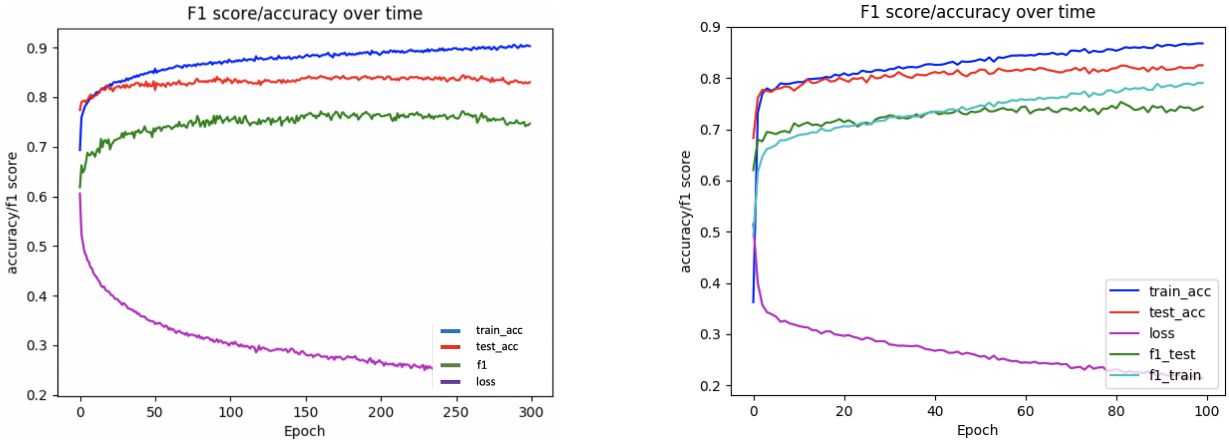

Figure 5: Training curves on **Twitter_big_2class**. **Left:** Cross-Entropy loss, **Right:** F1 loss.

Figure 5 shows 2 training graphs on the **Twitter_big_2class** dataset. The left shows an LSTM applied with Cross-Entropy loss, the second shows a Bidirectional LSTM applied with custom $(1-F1)$ loss; the best results achieved with the respective loss functions on this dataset. There is little difference between the two graphs, except that the generalization error is larger with Cross-Entropy loss, which is inferred from the larger gap between the train and test accuracy curves. Perhaps, directly maximising F1 score is directly informing the model to prioritise generalization across the 2 classes rather than overfitting to one class.

Although there was no increase in performance here, it was an interesting experiment.

E 3-class problem

With successful results in the 2-class problem, experimentation continued but now around the 3-class problem on the **Twitter_big_3class** dataset. The aim was to beat the state of the art for this problem, 0.93 weighted-macro F1 (Badjatiya et al. 2017), and get an insight into how well models can distinguish between different *types* of offensive messages, namely, racism and sexism. Additionally, categorical cross-entropy is used as the loss function, since we now have 3 classes.

Table 8: Results on **Twitter_big_3class** dataset

	Model	**Weighted Macro F1**
Machine Learning	Trigrams + SVM	0.8342
	Trigrams + Gradient-Boosted Classifier	0.8377
	TF + Logistic Regression	0.8399
	TF + Bernoulli Naive Bayes	0.8408
	Term Counts + Logistic Regression	**0.8465**
Deep Learning	Trainable GloVe + LSTM (100 units) + Dropout	0.8370
	GloVe + Bidirectional LSTM (100 units) + Dropout	0.8384
	ELMo + LSTM (256 units) + Dropout	0.8394
	GloVe + LSTM (50 units) + Dropout + Batch Normalization	0.8411
	ELMo + LSTM (512 units) + Dropout	**0.8419**

Interestingly, in the 3-class problem there is no differentiation between the performance of our Machine Learning and Deep Learning models; there are a large number of completely differently models all achieving weighted macro f1 apprximately 0.84.

Confusion matrices allow visualisation of a model's predictions against the test labels. Figure 6 shows an example of a confusion matrix for a well-performing model in the 2-class and 3-class tasks for the **Twitter_big** dataset. A reminder that **Twitter_big_2class** is the same dataset as **Twitter_big_3class** but with the sexism and racism classes combined.

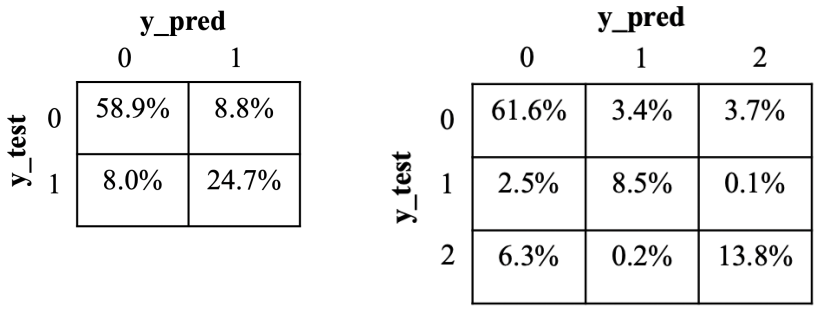

Figure 6: Confusion matrices.
Left: 2-class problem, **Right:** 3-class problem (0=None, 1=Racism, 2=Sexism)

The 2-class matrix shows that misclassification occurs both ways, our model often predicting positive instead of negative and vice versa. However, the 3-class matrix shows that although the model often wrongly predicts 0/None instead of racism or sexism and vice versa, the model almost never misclassifies between racism and sexism classes, only 0.1% of examples being classified as sexism instead of racism, and 0.2% as racism instead of sexism.

From this, we can conclude that Deep Learning models are much more effective at distinguishing between the different *types* of hateful speech, as opposed to detecting in a binary task whether hateful speech is present or not.

V EVALUATION - 2/3 PAGES

A Strengths and Limitations

B Is Deep Learning worth it?

It can be concluded that Deep Learning models exceed the performance of traditional Machine Learning for the 2-class problem when detecting Cyberbullying, since for all 3 datasets, we see a 0.0195, 0.0333 and 0.0352 increase in the best f1 score achieved respectively.

However, Deep Learning architectures are far more varied, contain more network parameters and the time and computational resources required for experimentation are far greater than that of traditional Machine Learning models. This extra difficulty must be considered for what is a relatively marginal (approx. 3% on average) performance increase.

VI CONCLUSIONS - 1 (OR 2) PAGES

Achievements...

A Future work

References

Badjatiya, P., Gupta, S., Gupta, M. & Varma, V. (2017), *Deep Learning for Hate Speech Detection in Tweets*, Hyderabad, Microsoft.

Bastidas, A., Dixon, E., Loo, C. & Ryan, J. (2016), *Harrassment detection: a benchmark on the #HackHarrassment dataset*, Intel.

Chatzakou, D., Kourtellis, N., Blackburn, J., Cristofaro, E. D., Stringhini, G. & Vakali, A. (2017), *Mean Birds: Detecting Agression and Bullying on Twitter*, Aristotle University of Thessaloniki and Telefonica Research and University College London.

Dixon, E. (2018), Automation and harassment detection. https://github.com/EdwardDixon/Automation-and-Harassment-Detection.

Golbeck, J. (2018), *Online Harassment*, Springer International Publishing.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. http://www.deeplearningbook.org.

Ioffe, S. & Szegedy, C. (2015), *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, Google Inc.

Kim, Y. (2014), *Convolutional Neural Networks for Sentence Classification*, New York University.

Ofcom (2017), Children and parents: Media use and attitudes report. unpublished.

Pennington, J., Socher, R. & Manning, C. D. (2014), Glove: Global vectors for word representation. Online research document.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018), *Deep contextualized word representations*, University of Washington.

Reynolds, K. & Kontostathis, A. (2011), *Using Machine Learning to Detect Cyberbullying*, Ursinus College.

Waseem, Z. & Hovy, D. (2016), *Hateful Symbols or Hateful People?Predictive Features for Hate Speech Detection on Twitter*, NAACL-HLT 2016.