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

Student Name: J. A. Leyland

Supervisor Name: Noura Al-Moubayed

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

Board of Examiners in the School of Engineering and Computing Sciences, Durham University

Abstract —

Context/Background —

Cyberbullying has affected an estimated 14.9% of high-school students in the last 12 months alone. It is defined as bullying (unwanted, aggressive behavior that involves a real or perceived power imbalance), that takes place over digital devices like mobile phones, computers, etc. through text, social media or forums. The behaviour is repeated, or has the potential to be repeated, over time.

Aims —

Unlike bullying in general, since cyberbullying takes place over electronic media, we may not see it taking place; it may go un-noticed. Furthermore, electronic messages lack context/intonation, which may make it hard to recognise cyberbullying. To attempt to solve this problem, the aim of this project is to create Machine Learning and Deep Learning models which reliably predict if a given message is an instance of cyberbullying or not. I will begin with a text messages dataset, and expand to social media platforms, investigating into whether a successful model for text messages can generalise to other datasets.

Method —

In this project, I will experiment with a variety of Machine and Deep Learning models to see if I can detect cyberbullying (or not) in unseen text messages. The input to my models will be some representation of the messages and the output will be one of two classes, cyberbullying (positive) or not cyberbullying (negative). Textual data can be represented using term frequency (with and without inverse document frequency), ngrams and word embeddings (amongst other forms), and I will evaluate the performance of many different text representation-learning model pairs, to see which create the most reliable predictions. Finally, I will judge the generalisation capabilities of my models.

Proposed Solution —

I will first search for an open-source dataset online which has already been hand-tagged for cyberbullying, and will explore this dataset to produce statistics to help me understand the dataset I am working with. There will then be some data pre-processing to clean the data and create an input that is easier to work with. Then, using Python and a number of libraries such as Sklearn and Keras for Machine and Deep Learning respectively, I will train various models on a subset of the aforementioned data and test them on the remainder of the data to evaluate the quality of each model I create. This will enable me to conclude if I can reliably detect cyberbullying. I will use these models to form predictions on a new unseen dataset (e.g. Tweets) to evaluate if my models can generalise to other datasets.

Keywords — Cyberbullying, Machine Learning, Deep Learning, Neural Networks, Natural Language Processing (NLP), Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs).

I INTRODUCTION

A *Purpose of project*

The research questions for my project are as follows:-
'What makes a good dataset for this task? How can I pre-process given datasets 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? Can this model generalise to other datasets and show high performance there?'

The world is becoming increasingly technological, and with this, social media and online content 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 is now equal to the figure of those who have been bullied face-to-face, which highlights the importance of being able to detect such behaviour online.

A.1 Cyberbullying

Cyberbullying is much harder to see and detect than physical bullying. As a result, we are reliant on online 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). Only 12.5% of these have used this reporting feature and so this highlights the scope for automatic cyberbullying detection in messages.

A.2 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) whose 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.

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 and 0.71 recall was achieved.

A.3 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, comparing as many different architectures as possible.

Badjatiya et al. (Badjatiya et al. 2017) achieved 0.930 Recall, Precision and F1 score - which represents the **state of the art** for my project area. I ultimately aim to get as close to this figure as possible (or surpass it).

B Specification of Deliverables and Aims

The overall aim is to attempt to create a model which detects cyberbullying reliably and effectively. I will begin with some simple Machine Learning models to set a benchmark for performance in this particular field. I expect these results to be imperfect, giving motivation to explore some multi-level architectures - Deep Learning. This is where the majority of my experimentation will take place.

Depending on the best performance I manage to achieve, I will be able to conclude whether Deep Learning can be used effectively in the detection of cyberbullying in messages.

B.1 Aims

I aim to experiment with a number of methods, falling under the bracket of both Machine Learning and Deep Learning. If I manage to find a successful solution, I will store these models and create a simple executable program that takes user input (an example of a message), and the model (quickly) returns a value, indicating whether the input message was cyberbullying or not.

B.2 Non-Functional Requirements

Table 1: Solution Requirements

Requirements
Final executable for classifying messages will be a working Python script which can be executed on multiple platforms
Executable can generalise reliably to numerous datasets, (but allow for slightly reduced performance).
Executable comes with instructions on how to run the program on your own instances of messages
Once the project is completed, provide a public online repository to help other researchers in future.
Code is well-written, modular, scalable and maintainable.

B.3 Deliverables

Table 2: Basic, Intermediate and Advanced project deliverables

Deliverable	**Description**
	Basic
Explore Datasets	Search online for datasets that are hand-tagged for cyberbullying. Analyse the datasets, clean them up, judge their effectiveness for this project. Decide on a dataset to use for the project.
Research existing work	Take inspiration from existing work for data pre-processing and model architecture ideas. Identify state-of-the-art (SOTA) for this task
Create first model	Create a traditional Machine Learning model better than a trivial classifier (predicts 0 every time or 1 every time). This can be observed if I get a good F1 score. This forms a performance baseline.
Evaluate	Evaluate this model. Look at precision, recall and F1 score. How does this compare to existing work?
	Intermediate
Word embeddings	Why should I use them? How can I implement them? Explore pre-trained embeddings such as GloVe and/or Word2Vec
Improved Model	Improve my existing Machine Learning model. Start experimenting with Deep Learning architectures
Evaluate and Explain	Once I have better performance with my Deep Learning models (compared to Machine Learning), evaluate their performance. How does this compare to existing work? How can I improve these models?
Save models	Implement the ability to save models, and load them later to make predictions again without retraining the model
	Advanced
Equal/Beat SOTA	Keep experimenting with Deep Learning architectures and data pre-processing to see if I can reach the state-of-the-art performance, or even beat it. Evaluate the best model I find.
Explore Architectures	Experiment with some unseen architectures. Justify rationale behind them. They may not perform perfectly, but would help develop understanding of natural language processing and Deep Learning.
Generalise	Once the best model has been found, see if this generalises to other datasets. Is the performance still high? If yes, why? If no, why?
Create Dataset	Create my own dataset for task for other researchers to use in the future. As I found it difficult to find the perfect public dataset for this.

II DESIGN

A Requirements

As mentioned in the Introduction section, this project has associated with it a number of requirements (mostly non-functional, such as maintainable code, a cross-platform executable, see Table 1) and then a number of deliverables (see Table 2) which define basic, intermediate and advanced milestones that I need to reach during the course of this project.

I have created a Gantt chart which visualises the deliverables of the project and the timescale in which I would expect to complete each task. This is vital to ensure that I remain on track; visualising the progress makes it much easier to interpret and work with.

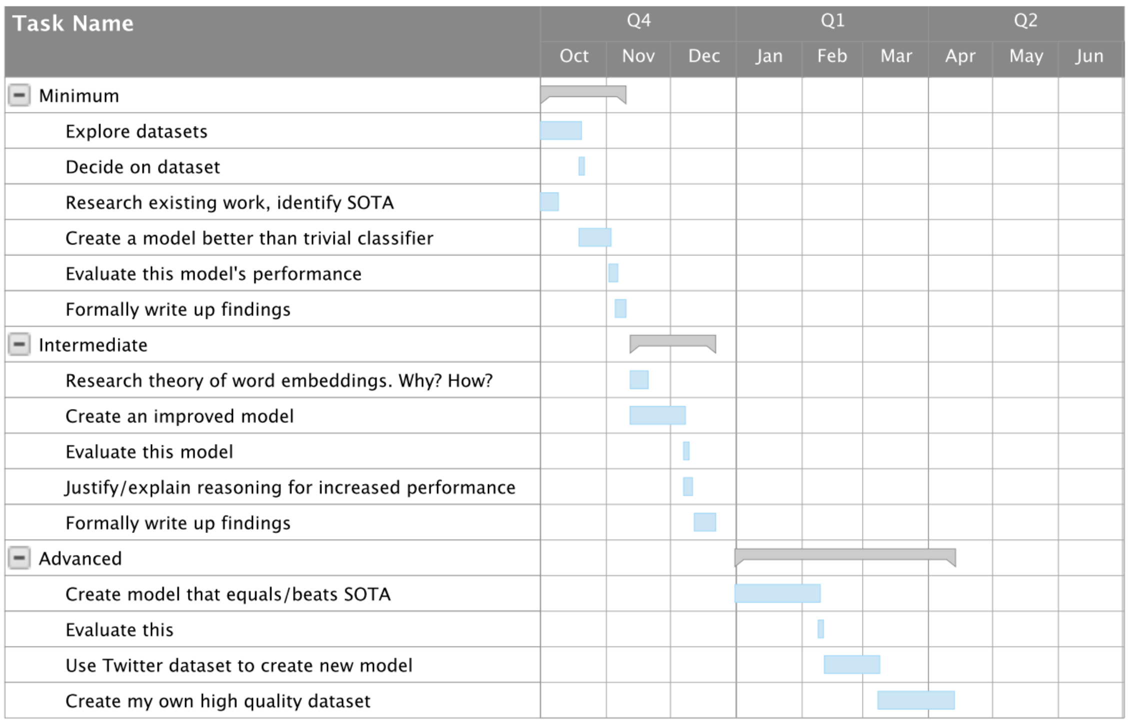

Figure 1: Gantt chart - visualising the timeframe expected to complete each deliverable in during the project.

B Tools Used

The nature of this task leads me towards spending the majority of my time in the implementation phase. There exists some research to be done regarding existing work in this field as well as the theory behind Machine and Deep Learning. however, most of the work centres around integrating my dataset with these models, experimenting with data pre-processing, creating models, evaluating their quality for cyberbullying detection, improving them 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** (ensuring that my code works in both version 2.7 and 3.4), 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, there exists a Python library called **Sci-kit Learn** (Sklearn), which provides a number of functions for easy data processing and machine learning model creation. This allows me to focus on a number of approaches and models, instead of the low-level mathematical details. Textual data needs to be converted to a fixed-length feature vector in *some way* in order for us to process it with Machine Learning Models; Sklearn offers fantastic support for this. For example, there are functions provided for tokenising and vectorising messages, then splitting data into train, development and testing sets. Additionally, there are a wealth of classifiers such as Naive Bayes, Support Vector Machines (SVMs), Decision Trees, Random Forests and more.

Furthermore, there is a library called **Keras** for my Deep Learning approaches, providing alternative data pre-processing functions, optimiser functions, and support for a number of architectures such as Recurrent and Convolutional Neural Networks, simple dense networks etc.

I will also use a number of common Python libraries such as **NumPy** (for mathematical operations) and **CSV** (for reading in the datasets).

My project might benefit from the modularity of Java, but this requires a considerable amount of planning regarding the system design. C++ enforces good programming practices and quicker running times, but also requires focus on the very low-level implementation details, reducing the time I can spend on the development my models. Other languages are typically less-widely used for this purpose and thus lack the support that Python has. As a result, all things considered, Python is the correct language for this project.

C Life-Cycle

A real-world project with an aspect of collaboration would require a large amount of forward-planning, documentation to keep on track, and constant communication to ensure smooth-running of the project. In this case, however, I am working completely on my own and thus I can operate with a more rapid-development kind of approach, similar to Agile. This will lead to creating executable programs extremely quickly relative to a plan-driven approach - only relying on finding an appropriate dataset beforehand.

I can jump into the implementation phase and work on a number of personal sub-tasks at once (as long as I use Git for version control). This allows me to design and implement at the same time, leading to rapid progress towards my deliverables throughout the course of the project. I can create predictive models and evaluate each model for the success of cyberbullying detection as I go along and record my results.

Additionally, if I hit any issues within my programs I can instantly edit my code as much as I want to and this won't cause any major issues - there are no documented system plans which would need updating if I employed a plan-driven approach.

D Searching for a Dataset (p.7)

The one pre-requisite for starting the implementation phase, as mentioned, is finding an appropriate dataset for this task. Since there exists previous work on the topic of automatic cyberbullying detection, I searched for the datasets used in these research projects to see if they were appropriate for me.

D.1 What is a Good Dataset?

I want a dataset that has been hand-tagged, forming the gold-standard data which we will be trying to model.

Ideally, the data will contain a column representing the text that I am classifying, with a corresponding 1 or 0 representing if this is a cyberbullying example. In (Chatzakou et al. 2017), they are classifying groups of tweets as coming from an aggressive or bullying *user* (or neither). This doesn't easily convert to the single-message 0/1 classification. (Badjatiya et al. 2017) classifies tweets as racist, sexist, or neither, which poses the same problem.

This dataset[1] is in the perfect format, containing 8000 text messages (an adequate number of examples) for use in a cyberbullying Twitter-bot. This is the dataset I will go ahead with.

I found 2 further datasets here[2] (used by (Reynolds & Kontostathis 2011)) and here[3], which are also appropriate. I will refer to these datasets as 'Formspring' and 'Tweets' respectively. They are ideal for the advanced deliverable of investigating if a successful cyberbullying model can generalise to unseen datasets effectively. The text messages and Tweets datasets contain relatively clean text, whereas the Formsping dataset contains lots of slang and mis-spelt text, which may make it more difficult to model.

Table 3: Dataset figures

Dataset	**Structure**	**#examples**	**+ve instances**	**Avg. length (words)**
Text Messages	Text and binary label	8817	28.4%	15.3
Tweets	Text and binary label	1065	40.1%	24.0
Formspring	Question text, Answer text 3 binary labels from 3 human labellers	12773	6.1%	25.4

I have gone ahead with the Text Messages dataset, since this has an adequate number of examples and isn't too imbalanced, contrary to the Twitter dataset and the Formspring dataset.

E System/Architectural Overview (p.7)

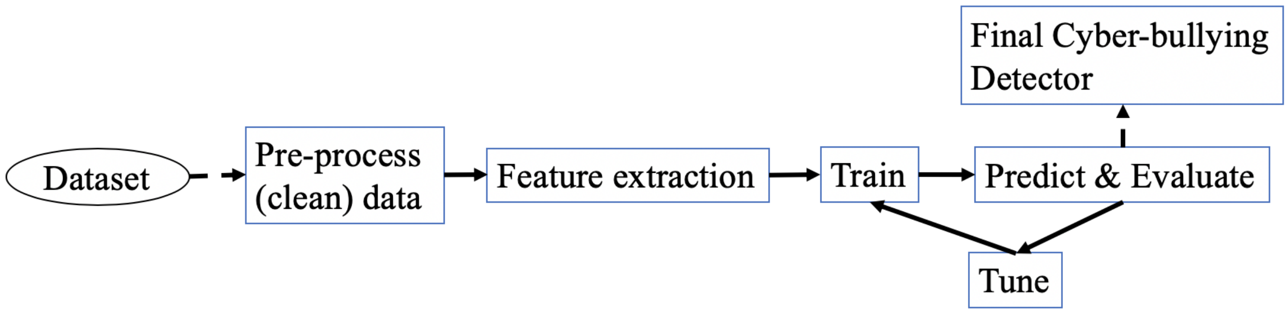

Figure 2: My system overview / pipeline

[1]https://github.com/varmichelle/Anti-Bully
[2]https://www.kaggle.com/swetaagrawal/formspring-data-for-cyberbullying-detection/data
[3]https://github.com/chantelmariediaz/Predicting-Cyberbulling-on-Twitter/blob/master/cleanprojectdataset.csv

F Specification and design (p.8/9)

F.1 Pre-processing

Since I am dealing with text data written by real-life human beings, there is a significant amount of noise and dirt within the data.

I have chosen the text messages dataset for my project. This dataset seems cleaner than a typical dataset of tweets/Facebook posts etc, as it doesn't contain any mentions ('@username'), hashtags ('#trends') or Twitter vocabulary ('RT') and doesn't contain many URLs either; only 17 out of 8000 text messages contain URLs.

Excessive punctuation, slang words and misspellings make the data dirtier, expand the vocabulary and reduce the amount of words that we have GloVe embeddings for ('see Feature Extraction below'). For example, over 200 of the text messages have no GloVe vector representation. This all ultimately makes the dataset more difficult to model and means that it is harder to correctly classify examples of cyberbullying.

My attempts to clean the data have included to remove comments with less than 3 words (too short to meaningfully classify), remove punctuation, remove repeated letters ('yessss' becomes 'yes') and make all text lower case.

If I manage to extend my research into a twitter dataset or another social media platform, I will need to remove hashtags, mentions, quote tweets etc. Also, many tweets will come from spam account or bots, which I will need some heuristic for detecting. For example, Wang (2010) notes that there are a much higher occurrence of mentions from spam Twitter accounts.

F.2 Feature Extraction

Machine Learning models require fixed length numerical vectors as inputs. Therefore, I cannot simply input text (of varying length) to the functions provided by Sci-kit Learn. I need some way to represent this textual data numerically and use these as the feature vectors.

I plan to experiment with 6 methods of feature extraction but may expand this depending on the results I find.

- **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 word vectors for each word in a message could be effective. The resulting vectors of cyberbullying messages may be similar as they could contain similar words.

- **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.

F.3 Training and Predicting

Once I have a representation of my input data, I need to train my models to fit the dataset and thus start making predictions on unseen messages.

There are a large number of possible Machine Learning models, as well as a theoretically infinite number of hyperparameters that I can use to tune these models. Tuning could theoretically go on forever, so I am going to use Sklearn's 'Grid Search' functionality, which allows us to train models on a number of parameters and take the best one. I will grid search across hyperparameters with a large range so that I am extremely likely to at least be close to achieving the best possible model with shallow ML methods. This will form my baseline performance, which I hope to improve on with Deep Learning models.

Once I have trained the Machine Learning models, I will undertake a large amount of experimentation with Deep Learning (which has even more scope than traditional ML) and see what kind of performance I can achieve.

Once I have trained a particular model, I can evaluate this model's performance based on predictions on a test set of data (data held out independent of the data that I am training on).

F.4 Evaluating

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 tasks such as this since our dataset contains an imbalanced number of positives and negatives, i.e. most text messages will not be cyberbullying. Therefore, I would get a near-perfect accuracy if I just predict 'not cyberbullying' every single time. F1 score ensures that I get a good balance between correctly identifying positive examples, but not wrongly identifying cyberbullying too often (predicting 'cyberbullying' every time).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

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

As a result, I want to maximise our F1 score.

G Potential Learning Architectures (p.10/11)

G.1 Traditional Machine Learning

In my path to finding a solution, I will experiment with traditional Machine Learning approaches for a considerable amount of time to find a baseline system performance. This will provide motivation for experimenting with Deep Learning, and I will be able to conclude afterwards if Deep Learning is more appropriate for detecting cyberbullying in messages than traditional Machine Learning.

Machine Learning models that I plan to experiment with (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.

- **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.

I will most likely explore more models during implementation.

G.2 Deep Learning

Machine Learning methods will perform a baseline performance which I aim to beat with Deep Learning. The following sections will detail aspects of my Deep Learning solution designs.

G.3 Word Embeddings

When using Deep Learning, I will represent the input (messages) with Word Embeddings. This is where we represent words in a vocabulary 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). For example, this allows messages containing the word 'cat' to help inform predictions on messages containing 'dog'.

GloVe (Pennington et al. 2014) is an example of a general-purpose pre-trained word-embedding based on factorising a matrix of co-occurrence statistics. I can fix this embedding and apply it to the input data, training and tuning only the remainder of the model.

Alternatively, I can randomly initiate a word embedding and then train the word embeddings along with the rest of the model to fit the training dataset. The model will take much longer to train; however, the resulting embedding will be more specific to a cyberbullying context and thus *could* lead to better performance in this case.

G.4 Recurrent Neural Networks (RNNs)

I will apply networks known as Recurrent Neural Networks to the word embeddings in my Deep Learning models. 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, indicating if there has been offensive language, or targeted language etc.

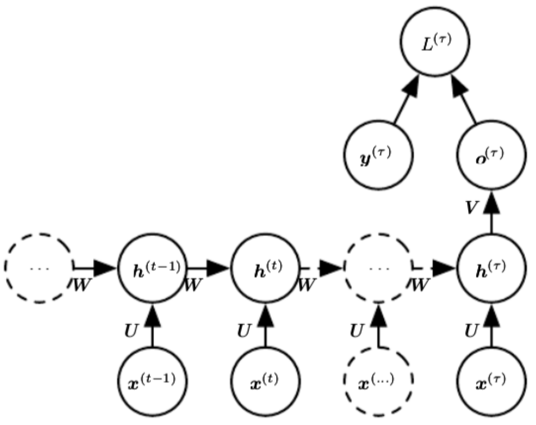

Figure 3: **RNNs.** h**=state,** x**=input,** y**=gold label,** $\sigma(T)$**=predicted label,** t**=time/index**

I will take the final 'state' of the system and apply the sigmoid function to get the final output - the probability that the input sentence was a cyberbullying instance or not.

I specifically, will be using a type of RNN known as Long-Short Term Memory (LSTM) cells. (Goodfellow et al. 2016, p.410) Traditional RNNs suffer from the vanishing gradient problem, which makes it hard to train them. In other words, RNNs struggle to learn long-term dependencies in temporal data or text. 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.

G.5 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs), clearly, 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)

I will utilise CNNs in 2 ways in my proposed solutions:

- Apply numerous CNN layers and pooling layers, followed by a traditional dense neural network to eventually classify output into my two classes.

- Apply CNN layer + Pooling to reduce feature size. Run the output from the pooling layer through an LSTM cell. Finally, apply sigmoid function to the final state of the LSTM cell to classify the sentence.

H Creating a new Dataset

At the close of my project, I aim to create a large, reliable, human-labelled dataset for further research into this topic. One issue in cyberbullying research is that the number of datasets are limited, often due to privacy/data protection, so I would need to strip all names and identification from the data. Additionally, hand-labelling takes a huge amount of time since we ideally want 10/100K examples in a dataset for this task.

Therefore, I would aim to utilise crowd-sourcing services to share the workload on hand-tagging this data. Figure-eight[4] and Amazon Web Turk[5] are appropriate tools for this. However, they would pose problems such as the aforementioned privacy considerations and also cost.

I References

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.

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

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.

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

Wang, A. H. (2010), *Spam Detection in Twitter*, Pennsylvania State University.

[4]https://www.figure-eight.com/
[5]https://www.mturk.com/