Literature Survey – v2.0

By Jack Leyland

Introduction

Deep Learning, put simply, is a subfield of Machine Learning, using algorithms inspired by the structure and function of the brain called artificial neural networks[1]. I endeavour to experiment with a variety of Deep Learning methods, to learn the indicators of cyberbullying in social media messages and accurately detect if a given message is likely to be classed as cyberbullying or not.

I will be using numerous tools in my work towards a solution. For Deep Learning technologies I will utilise TensorFlow[2], an open source symbolic math library which can be utilised in Python, along with TensorBoard[3] to visualise the performance. I have sourced a hand-tagged for cyberbullying set of tweets[4], which (once pre-processed where appropriate) will form the input for my networks and I aim to create a network which accurately represents this dataset. There will also be the use of other libraries such as NumPy for efficient manipulation of my input data. I may also use the Twitter API to extract further Tweet information to enhance my datasets.

From the definition[5], cyberbullying is present if a message contains harmful or mean content about somebody else. Deep Learning models are relatively black-box, so I won't be able to extract the exact indicators of cyberbullying, but the model itself will learn these and - using these indicators - be able to predict with a reasonable accuracy the likelihood of unseen examples being classed as such.

My motivation for this project stems from my year abroad at Charles University, I studied the basics of Machine Learning and Deep Learning and this piqued an interest in Artificial Intelligence. My 3rd year project seemed like the perfect opportunity to probe further into this topic. I, personally, have had experiences with cyber-bullying on Twitter, and analysing variable-length text poses a new challenge for me, so this seemed like the perfect application to research into.

---

[1] Jason Brownlee, 'What is Deep Learning?', Machine Learning Mastery, 2016. https://machinelearningmastery.com/what-is-deep-learning/ [19/06/18].
[2] https://www.tensorflow.org/
[3] https://www.tensorflow.org/guide/summaries_and_tensorboard
[4] https://github.com/varmichelle/Anti-Bully/blob/master/datasets/new_data.csv
[5] stopbullying.gov, 'What is Cyberbullying', www.stopbullying.gov/cyberbullying/what-is-it/index.html [28/09/18].

Existing Solutions

There are many examples of previous work that have aimed to achieve the same (or similar) goal as me in this project. Each approach is different, and I hope to collate observations from all of this work to approach my project in the best possible way to achieve the best results.

In 2011, Reynolds et al. aimed to use Machine Learning to Detect Cyberbullying[6]. They used 4000 examples from Formspring data (a social media platform where you ask other users questions anonymously) which was hand-tagged for cyberbullying. Their approach included a significant amount of data pre-processing – the most effective method being to pre-compute features such as the number of 'bad' words with respect to the length of the text and then using decision trees to classify. This method achieved **78.5% recall**. From this, I have learnt that basic, traditional machine learning methods are relatively effective for this task. I could attempt to pre-process some features in a similar way, but apply a simple dense neural network as opposed to decision trees. Additionally, I will be setting 78.5% recall as my initial target for my models. Another interesting observation was that the positive cyberbullying examples were under-represented and composed only 10% of the dataset, so the model had limited data to work with to learn indicators of cyberbullying. Better result were seen when they repeated positive examples 8 times each in the development set.

In 2016, Bastidas et al. gave more insights to this field by attempting to apply Deep Learning methods in the cyberbullying context[7]. They created their own dataset (#HackHarrassment v1.0) which aimed to improve the size and quality of open source datasets for this task (although I have struggled to source it for myself). They hashed unigrams/bigrams/trigrams of the text, computed the frequency of term frequency for each hash value and these formed the features for classification with decision trees, 73.3% accuracy and **71% recall.** These figures are actually worse than those achieved by Reynolds. Furthermore, when applying a deep learning approach (details limited in the text), a slightly improved **recall of 73.3%** was achieved. Precision was down from 80% to 71%, however, so no real improvement was achieved with Deep Learning. From this paper, more than anything, I have learned that Deep Learning approaches don't automatically improve performance from traditional Machine Learning methods. I will need to take the right steps to ensure I am applying the methods on an input dataset that makes sense and allows us to exploit Deep Learning technologies more appropriately.

In 'Mean Brids: Detecting Aggression and Bullying on Twitter, 2017'[8] by Chatzakou et al, more advanced approaches were taken to increase performance compared to the previous 2 papers. The main improvements here cantered around the use of data in the solution. The task here was slightly different, to tag a user (using a batch of their tweets) as bully, spammer, or none. The dataset here had a huge 1.6M tweets, fetched with the Twitter API

---

[6] Kelly Reynolds, April Kontostathis, Lynne Edwards, 'Using Machine Learning to Detect Cyberbullying', , 2011. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.4788&rep=rep1&type=pdf [13/06/18]

[7] Alexei Bastidas, Edward Dixon, Chris Loo, John Ryan, 'Harassment Detection, a Benchmark in the #HackHarassment Dataset', Intel, 2016. https://pdfs.semanticscholar.org/c22d/fb9eff1c8c538d40a51609e7593cc9ded136.pdf [13/06/18]

[8] Despoina Chatzakou et al, 'Mean Birds: Detecting Aggression and Bullying on Twitter', Aristotle University of Thessaloniki, 2017. https://arxiv.org/pdf/1702.06877.pdf [13/06/18]

along with additional user profile features. This data was then cleaned (make text lower case, remove stop words, repeated characters etc.) and then classified with a Random Forest model. Again, we have a standard machine learning approach but the high quality input data helped achieve a huge **91.7% recall**, with 89.9% precision and 91% accuracy. I will keep this figure in mind as my reference for 'state of the art', which I ultimately aim to achieve, but as this is a different task it's not a directly comparable figure which is a slight disadvantage. This paper made it clear to me that this isn't just a Deep Learning task but also a Data Processing task, as simple changes to the way we use the data can show vast improvements in the performance of detecting hateful speech. Additionally, it made me aware of the Twitter API and its potential for data collection, and perhaps I will be able to use this tool to further my solution quality.

Finally, Badjatiya et al applied (and went into details with) Deep Learning technologies to detect hate speech in 2017. This paper furthers the work of the previous 3 papers and reaches performance levels above those previously seen, with precision and **recall at 93%**. The task was to classify examples of tweets (of which there were 16K) as racist, sexist, or neither. This performance was achieved when computing word embeddings and then classifying with a gradient-boosted decision tree classifier. Other (less successful) methods included randomly initiating word embeddings, applying CNN layers or running them through an LSTM cell and then classifying. These methods achieved approximately **84% recall**. I can take inspiration from these methods, and unlike Bastidas et al.'s work, we can see that applying Deep Learning methods correctly can improve our solution performance. Importantly, we can see the role of word embeddings and how they capture word relationships in text. I need to apply these to my solution if I expect to see promising results.

Comparison/Summary of Existing Solutions

From the first 2 of the above papers, we can see that an accuracy of approximately 70-80% is definitely achievable for detecting cyberbullying and this will be the figure I aim for in my project. The 3rd paper achieved a much higher accuracy, and I will use this as state of the art (91% accuracy, 91.7% recall) and my ultimate goal will be to reach something like this figure.

A common theme throughout the papers is data pre-processing. The first 3 papers manipulate the text data in some way beforehand, so that the models can be simplified as they are working with more discrete data. E.g. n-grams frequency as opposed to one piece of text with variable length. The final paper tackles this problem by working with word embeddings directly which aim to capture relationships between words in a sentence. Using RNNs on these introduce an idea of context, which will be vital in achieving the best results. I need to implement this into my solution to achieve good results.

However, Deep Learning methods didn't always improve the quality of these models (in the #HackHarrassment paper, performance was actually better with a simple decision trees algorithm). As a result, I plan on implementing both traditional machine learning and deep learning models to make my own comparison in their overall performance.

Finally, the very best solution I found in terms of precision and recall (both 0.930) actually *combined* deep learning methods (convolutional neural networks) and a gradient-boosted decision tree classifier.

Challenges

Finding a dataset

It was difficult to source a set of tweets, hand-tagged as cyber-bullying or not. Many weren't open source or were tagged slightly differently (bullying, spamming etc.).

Eventually, I found a dataset that I will be using which contains 8818 hand-tagged tweets.[9] This could be a sufficient number of tweets, I expect it to be since there are plenty of positive examples (~2505). Also, the 4th paper above achieved 0.930 recall with only 16K tweets, so I believe good results are achievable with this dataset.

I may want to enhance my dataset with some extra user/tweet information, as done in 'Mean Birds, 2017'. However, the users id's weren't provided in the dataset so at this time I am unsure if this dataset will be usable.

Overfitting

Since my dataset isn't huge (8818 tweets), my models would definitely be prone to overfitting. If the model over-fits to the training data, it won't generalize to unseen test tweets very well and this is bad for performance. If necessary, I will search for an even larger appropriate cyberbullying dataset. Additionally, each example is a maximum of 140 characters, so context within the examples are limited which may make cyberbullying detection more challenging. There are some approaches I can take to avoid overfitting (regularisation techniques for Deep Learning):

Dropout[10]

Dropout is a regularisation technique that 'drops' some nodes out of the network with a certain probability. This means that during training, batches of tweets will be run on many different sub-networks of the network as a whole. This will ensure that the node maximises its capacity with all sub-networks being trained to represent indicators of cyberbullying. TensorFlow provides methods that apply dropout to your networks easily.

Early Stopping[10]

Generally in Deep Learning, if we have a model with a high representational capacity, the training error will decrease continually as we fit it. Validation error decreases over time too but after a point, the validation error begins to increase. This is because we have started to overfit to the training data. I will counter this by recording the validation tweet set accuracy after each epoch, and saving the parameters where this accuracy was the highest.

---

[9] https://github.com/varmichelle/Anti-Bully/blob/master/datasets/new_data.csv

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning book,* (The MIT Press, 2016) [13/09/18].

Batch normalisation[11]

When backpropagating after a batch of tweets has been processed, the gradient tells us how to update each network parameter, under the assumption that the other layers do not change. Unexpected results can happen because many functions composed together are changed simultaneously when updating. Batch normalisation is a method of reparameterization that reduces this effect. TensorFlow provides methods which allows us to apply batch normalisation to a network easily.

How to evaluate my models

The papers above used precision and recall more frequently than the accuracy of the predictions to evaluate the models. I believe this is because in random samples of tweets, there will be a low proportion of positive cyber-bullying examples. As a result, recall is useful as this is a measure of how well we detect these specific examples.

As a result, when evaluating my models, I will follow suit and consider precision and recall as much as I do accuracy. Further to these, another metric I will evaluate is the F1 score, to give another value we can compare across models.

Processing informal/badly spelt text

Informal/badly spelt text, URLs etc. will add complexity to the text we are classifying and increase the vocabulary size that our network must capture. We want to normalise the text (without affecting the semantics of the messages) as much as possible. I will follow a similar method to that in 'Mean Birds', who pre-processed all text by removing repetitive characters (e.g. 'yesssssss' becomes 'yes'), removing URLs and making all text lower-case. I believe this will help the performance of my solution.

[11] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning book,* (The MIT Press, 2016) [13/09/18].

My Proposed Solutions Ideas

Based on the 4 papers I have discussed, I have come up with 4 high-level descriptions of potential models for my solution. They vastly vary in complexity and my prediction is that this will correlate with a variation in performance too. I felt a variation in complexity was key to further develop my understanding of these topics, and I am intrigued to see which changes lead to significant increases in performance. Of these methods, all are inspired by some previous work, and I have referenced this where applicable.

1. Logistic Regression (simulated in Neural Network) on n-grams.

Pre-compute frequency of n-grams for each tweet, use these as the features. Create a Neural Network with 1 hidden layer containing 1 node with no activation function, and a single output node with the sigmoid activation function.[12]

Of course, I could use a simple Machine Learning package to implement Logistic Regression, but I am interested in creating a Neural Network equivalent to aid my understanding of how they work.

2. Pre-compute some features, classify with dense network

Typical approach. Pre-compute some features such as number of swear words, number of exclamation marks etc. Could combine with n-grams. Run these through a dense network and then classify.

3. RNN on word embeddings, then classify with simple dense network

This is a method described in the TensorFlow documentation[13] (which is my framework of choice). The method is as follows: randomly initiate word embeddings, perform some convolutions, run through a RNN, then use resulting state from RNN as input to dense network to classify.

Another example here[14] uses the same methodology.

4. Randomly initiated word embeddings + CNNs

Using 'Deep Learning for Hate Speech Detection' as inspiration. (see Existing Solutions section)[15]. The method is to initiate word embeddings (randomly, or use existing embeddings found online), apply a series of 1D convolutions, run through a simple dense network and then classify.

---

[12] James D. McCaffrey, 'A Neural Network Equivalent to Logistic Regression', James McCaffrey, 2017. https://jamesmccaffrey.wordpress.com/2017/07/01/a-neural-network-equivalent-to-logistic-regression/, [11/09/18].

[13] https://www.tensorflow.org/tutorials/sequences/recurrent_quickdraw

[14] https://github.com/kamei86i/rnn-classifier-tf.

[15] For implementation details, see here: https://github.com/pinkeshbadjatiya/twitter-hatespeech

Conclusion

In conclusion, detecting cyberbullying in messages is a significant challenge and I can see this from previous work around the topic. There are numerous approaches to machine learning solutions, and traditional classifiers which are applied to simple pre-computed features (ngrams, swear word counts, etc.) provide relatively promising results. The best performance comes from a wealth of data pre-processing such as extracting user information from twitter to enrich a Twitter dataset, or to utilise word embeddings with Deep Learning technology to capture word relationships and context in sentences. I plan to experiment with solutions inspired by a number of these ideas to give a reliable comparison of the approaches and their effectiveness (see 'Proposed Solution Ideas' above).

I will face a number of generic Deep Learning challenges along the way such as overfitting, regularisation, dataset quality etc. but this all forms part of the challenge which I hope to solve effectively.