Project Description

1 Introduction

With the advent of social media online, information, thoughts and opinions have never before been shared as freely and globally. The potential of this is largely beneficial; it allows us as a society to organise public affairs, communicate with people around the globe and educate almost anyone regardless of wealth or social status. However, with the positive strides it allows us to make as a society, it would be remiss to ignore the negative consequences.

As the ability to freely express ourselves escalates, so too does the exposure of harmful expression such as hate speech, which promotes illegal discrimination, causes distress to targeted individuals and groups, and – in some cases - incites violence and harm. Given the massive amount of user generated data uploaded to social media every day, it` has become extremely important that social media companies develop tools that automatically detect hate speech using natural language processing for commercial, regulatory and ethical reasons as human detection and intervention is not realistically possible with the immense stream of user generated content uploaded online every day. Therefore, to have an automatic NLP model to detect hateful posts will serve as a good first line of defence against hateful user-generated content online

It is often difficult for even humans to distinguish a ground truth as to what constitutes as hate speech [Sue et al., 2007], as there are many differing opinions as to what exactly is hate speech. Unlike other types of pernicious activity, such as spam or malware, the twitter accounts posting hate speech online are controlled by humans, not bots. This makes identifying this type of language less predictable and thus less identifiable. The final NLP model that will be created will aim to identify hate speech broadly based off the following definition - Hate Speech (HS) is commonly defined as any communication that disparages a person or a group on the basis of some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics [Nockleby, 2000].

2 Software environment and modelling framework

Python will be used to complete this project using Jupyter notebooks and Google Colab. Google Colab is essential to this project as it provides access to a Cloud TPU (Tensor Processing Unit) on which we can train our model. Without access to this virtual memory we cannot train deep neural networks and indeed the best version of BERT, which is BERT large, (having 24-layer, 1024-hidden, 16-heads, 340M parameters) as we get out-of-memory errors on our local CPU and even on the computer science building computer's GPUs.

2.1 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a language representation model developed by Google and it is designed to pretrain deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks [Devlin et al., 2018] so it is perfect for the creation of my NLP model.

BERT has been developed with the Tensorflow framework and so it is this main machine learning package we will use. Google Cloud's TPUs have been developed with Tensorflow computations in

mind and optimised accordingly, therefore for the foreseeable future Tensoflow will be used in this project.

Pytorch may be explored in the future as this library has much praise in the machine learning community with its plaudits for it's simple to use functionality, conciseness and efficiency, as well as sometimes better performance with it's additions to the BERT architecture – see RoBERTa developed by Facebook [Liu et al., 2019]. However, for the moment mastering Tensorflow is the priority as I'm a relative novice to it

3 Data

The data so far collected has come from numerous sources. To optimise the performance of the final classifier, our language data must be generalised. Therefore, we will use the single data source of Twitter, so all the language that will be used to train the NLP model will be collected from tweets. As we plan to use a neural network model to classify tweets, getting as much tweet data containing genuine hate speech as possible is the goal.

3.1 Strategy on data collection and pretraining

My strategy initially was to collect tweets from sources that annotated either abuse/offensiveness or obviously hate speech. The reason for this is that because I believed from the outset that the biggest obstacle my model would have to overcome in classification was to learn the separation between hate speech and other instances of offensive language, as often automatic hate speech detection programs are over-sensitive to sarcasm or sentences containing particular terms and this results in many false positives [Davidson et al., 2017]. Also, given the legal and moral implications of hate speech, it is important that we develop a model capable of accurately distinguishing between the two.

Thus, my goal from the outset was to use the methodology of transfer learning to fine-tune BERT to detect offensive tweets and from there fine tune my system again to extract what offensive tweets were hate speech. However, I now realise that while this strategy still may be effective – and may be used down the line still, it's the wrong place to start and it's not considering the shortcomings of BERT when addressing twitter vernacular in general.

The pre-training corpus for BERT was from two sources BooksCorpus (800M words) [Zhu et al., 2015] and English Wikipedia (2,500M words). While this corpus may cover much of the lexical of the correctly spoken English word, it likely does not adequately understand the parlance of people on twitter e.g. *@JuanYeez shut yo beaner ass up sp*c and hop your f*ggot ass back across the border little n*gga.* (from ICVSM 2017), language like this would be remiss on an official site like wikipedia.

To hopefully incorporate this into my future model, my plan is to undertake further pre-training of the large BERT model in the target domain of twitter. This method has been explored in detail [Sun et al., 2019] which demonstrated that further pre-training on in-domain and within task data could significantly boost performance on a target task. I will attempt to gather tweets where more colloquial terminology is used as this seems to coincide often with Hate Speech online, especially in such an informal social medium as twitter. This model will benefit from transfer learning from BERT which will provide some knowledge to more conventional language at least, which is also undeniably useful in any NLP task. 100k training examples seem to be the recommendation of the authors of the paper, but I will attempt to pre-train on more tweets as well and see for myself the effect this has.

3.2 Training Data

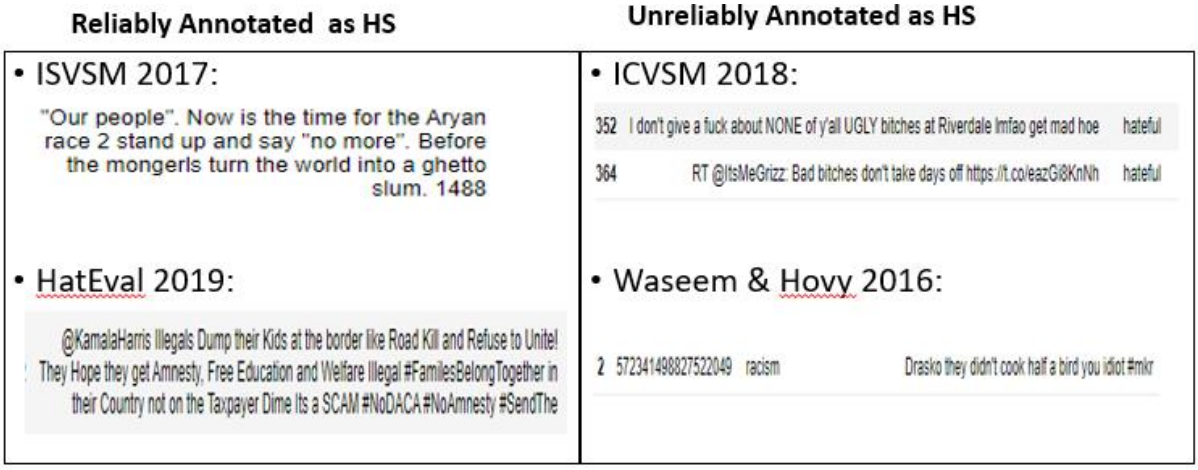
In total 148,057 tweets have been collected so far, the sources for this data presently include data from the SemEval 2019 Shared Task on Multilingual Detection of Hate competition on Codalab [Basile at al 2019], which is a hate speech dataset with the targets of abuse being women and immigrants, also included is data from a related competition by Codalab; OffensEval 2018 [Zampieri et al 2019]. However, this dataset did not specify whether tweets were hate speech they did specify whether the content of tweets was offensive and they were directed at a whole host of targets – not just women and children, which is important because Hate speech can be directed at a spectrum of targets.

Also included were the datasets from ICVSM 2017 [Davidson et al], ICVSM 2018 [Founta et al 2018] and Waseem & Hovy 2016 [Waseem and Hovy 2016]. ICVSM 2018 was a dataset with unreliable hate speech annotation as it was annotated with abusive and offensive based content in mind, but it contained 99,996 tweets so I believed it would be a valuable resource for learning. ICVSM 2017 was labelled for hate speech but among an informal sampling of opinions among friends I found that it sometimes lacked the consistency in it's labelling of hate speech compared to the initial SemEval 2019 dataset, which was much more often clear-cut hate speech.

Figure 1: Summary and analysis of the collected datasets

Source	Hate Speech	Not Hate Speech	Total	Reliably Annotated	Notes
HateEval 2019	4,210	5,790	10,000	Yes	Very reliably annotated. Examples of hate speech are clear cut
OffensEval. Offensive Target Identification 2018	0	12,166	12,166	Not when concerning HS	No tweets labelled as Hate speech. But many labelled as offensive. Some tweets (1074 total) taken out to avoid possible contamination
ICVSM 2017	1,430	23,353	24,783	Mostly yes	For the most part reliably annotated. A few strangely labelled HS here and there though
ICVSM 2018 Abusive Behaviour Database	0	87,967	87,967	Not when concerning HS	There are 4965 tweets marked as 'hateful', however these do not qualify as hate speech. Again, these tweets are removed to avoid contamination. Also there were 7,064 duplicate tweets
Waseem & Hovy 2016	3,534	9,638	13,172	Not certain	Tweet datset with just tweet IDs, had to be retrieved using twitter API. A lot of the racism and sexism wasn't clear cut, seemed to be contextually based often which is hard to distinguish and could throw off classifier
Total	**9,174**	**138,914**	**148,057**		**In all, 7,122 tweets were removed from the overall dataset because of duplicate tweets and because some tweets were unreliably annotated as hate speech**

Figure 2: As mentioned above, some of the provided datasets have often unreliably characterised some tweets as Hate speech, which could confuse our model in the training phase. Demonstrated below is the dichotomy between reliably classified tweets and unreliably classified tweets:



3.3 Evaluating and testing

I have not yet developed a satisfactory and practical method to evaluate, and later test, the performance of my model. So far I have used the evaluate functionality in Tensorflow's estimator function, although this gives very unsatisfactory evaluation as it only returns the accuracy metric - and not precision, recall, auc, or indeed F score, which give us far more insight into our model's performance and where it can be improved. (Although I am working on developing a method to calculate these metrics)

A future approach may be to use cross-validation within my training set – however this may not be a fair and true test of my model's performance. Another plan of action may be to enter my predictions on unlabelled test data provided by the semEval 2019 competition on which this project is loosely based – but it has two further sub-tasks of identifying whether the tweet was targeted and whether it was aggressive, and I believe I may have to classify for these two sub-tasks if I am to receive an evaluation.

The makeshift method of evaluation I'm utilizing presently is by entering a contest on the Indian based machine learning website analyticsvidhya (https://datahack.analyticsvidhya.com/contest/practice-problem-twitter-sentiment-analysis/). This contest has 12,166 participants as of writing, it has been ongoing since the start of the year and it will end on the 30th December this year.

The tweets are very unreliably marked as being hate speech but nevertheless achieving a high score compared to other competitors would demonstrate value in my BERT fine-tuned model. The training dataset contained around 32,000 tweets with 2,242 annotated as hate speech. (7% of overall dataset) and the test dataset was unlabelled.

Figure 3: The public leaderboard for the analyticsvidhya, currently I am in 6th place with an F score of ~0.827 out of 12,166 participants.

Public Leaderboard - Practice Problem : Twitter Sentiment Analysis

#		Name	Score	Submission Trend	Participant's approach
1		kysmet	0.8583815029		
2		becnic	0.8575498575		
3		saurabh502	0.8571428571		
4		robertvici	0.8514056225		
5		m0baxter	0.8311688312		
6		fionn49	0.8267045455		Add approach
7		akash780	0.8237037037		
8		swapnilg	0.8223495702		
9		pagadi	0.8190184049		
10		oahmia	0.8130311615		

4 Pre-Processing Tweets

4.1 Text Pre-Processing

I created a basic function which accepts a text string and it:

1) Removes URLS
2) Replaces excess whitespace with one instance
3) Removes mentions
4) Uses the html.unescape() method to convert unicode to text counterpart
5) Replaces & with and
6) Remove the fact the tweet is a retweet if it is

Figure 4: Text pre-processing has evidently improved the F-score of my model (F-score on test predictions in the right column)

Sun Dec 08 2019 19:09:07 GMT+0000 (Greenwich Mean Time)	no preprocess, same params	0.764444444444444
Sun Dec 08 2019 17:52:48 GMT+0000 (Greenwich Mean Time)	Preprocessing added	0.8

Currently I am operating under the assumption that mentions and URLs in tweets do not give us useful information from an NLP standpoint. I may later experiment with converting these instances in tweets to a common tag like “@user” and “url” respectively (as they could provide contextual information) and analyse if they improve the model’s effectiveness.

I also have created a function which translates emojis to words, however it remains to be seen if this improves the performance of our model as the tweet corpus we’re currently evaluating on (analyticsVidhya) does not contain any emojis. Also tweets laden with emojis may be converted to a much larger sequence – which can be troublesome for BERT if it exceeds 512 in sequence length.

Figure 5: Pre-Processing Demonstration

I may also experiment with other text pre-processing techniques common in NLP such as

- Punctuation removal - although may not be much use as BERT has been pre-trained on a corpus containing punctuation
- Contraction – again, may not be useful for the same reason as above
- Hashtag segmentation – isolating hashtags that occur often in hate speech tweets could prove a very useful and simple method in identifying hateful behaviour
- Unsupervised Data augmentation for a neural network approach – (via paraphrasing tools and via backtranslating) [Xie et al., 2019]

I considered using lexical detection methods by using the hate speech lexicon containing words and phrases identified by internet users as hate speech, compiled by Hatebase.org. However, upon research, I discovered that this method would result in false positives, especially in our task of hate speech detection as terms like h*e and b*tch may be pejorative in some contexts, but not all. Although perhaps a more restricted lexicon may prove useful (one is provided by Davidson et al., 2017 in their github repository)

4.2 BERT Pre-Processing

We must also pre-process our text so that it resembles the text that BERT was trained on. For this reason we must use a BERT tokenizer, which was a pre-processing method the original BERT model was pre-trained with. Tokenization uses Wordpiece tokenization, the vocabulary is initialized with all

the individual characters in the language, and then the most frequent/likely combinations of the existing words in the vocabulary are iteratively added. Our existing knowledge of vocabulary is retrieved from the BERT model's vocab file, which has each word on a new line, with some subwords like -ing and -ed represented as ##ing and ##ed. This method allows BERT to identify a much wider variety of words, for example the word "tokenizer" is not in the original vocab file, but it is instead broken down into sub-words greedily. Each word is mapped to a unique Id by BERT so it is more interpretable by the system

We also convert our tweets into features so BERT can interpret them. The inbuilt function *convert_examples_to_features()* adds the [CLS] and [SEP] tokens to denote the beginning and end of a unique tweet and it also appends "index" and "segment" tokens to each input

Figure 6: Example of tokenization

```
['this',
 'here',
 "'",
 's',
 'an',
 'example',
 'of',
 'using',
 'the',
 'bert',
 'token',
 '##izer']
```

Figure 7: Converting tweet text to features

```
INFO:tensorflow:tokens: [CLS] hu ##rra ##y , saving us $ $ $ in so many ways # lock ##the ##mu ##p # bui
INFO:tensorflow:input_ids: 101 15876 11335 2100 1010 7494 2149 1002 1002 1002 1999 2061 2116 3971 1001 5
INFO:tensorflow:input_mask: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
INFO:tensorflow:segment_ids: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Also, to fully utilize the TPU power, we need to feed the training data in the most efficient way possible, for which I plan to use a TFRecordDataset by encoding our training examples into tfrecord files.

5 Fine -Tuning BERT

As mentioned before, a key strategy moving forward in this task is to further pre-train BERT on unsupervised data - namely twitter language, specifically from individual users who use slang and perhaps speak in a more colloquial dialect – as this type of speech is often used along with hate speakers (although not always).

Also to be considered is transfer learning on a related task before actually fine-tuning on the target task as mentioned briefly before. I have already collected a wealth of supervised data on offensive language and this could be a good pre-cursor to the actual task at hand and improve performance. This method (multi-task fine-tuning) [Sun et al., 2019] has demonstrated performance improving potential in NLP before such as natural language inference [Conneau et al., 2017] and machine translation [McCann et al., 2017].

More research is needed on my end to determine the best practice for tuning hyperparameters and what type of models to pursue when coding the final layers to fine-tune the model for my target task and indeed possibly for multi-task fine-tuning. Bi-LTSM models seem to be the consensus best type of classifier with the fleeting research I've done on this.

I must be cognizant on common problems with transfer learning such as catastrophic forgetting, where the pre-trained knowledge is erased during learning of new knowledge. An appropriate layer-wise decreasing learning rate has been shown to mitigate this.

6 Solution Approach

Summarising my findings above, I can distil my future approach to this task in a simple TODO list:

- Become much more familiar with the TensorFlow framework. Also explore pytorch huggingface library as an optional framework
- Develop a better method of evaluation. Attempt to return an F score, precision and recall to better evaluate the performance of our model on the dev set. Explore using cross validation also - time and processing ability willing.

- Explore further text pre-processing techniques such as hashtag segmentation, punctuation removal and translating emojis into their textual representation. Analyse what effect this has on performance

- Use unsupervised data augmentation. It's ability to boost performance has been demonstrated. Use paraphrasing tools and backtranslation

- Convert data to tensorflow dataframe object to better take advantage of the TPU runtime. This promises to boost processing speed and perhaps even model performance.

- Utilize the method of within-task and in domain pre-training. Research the best method to do this – perhaps consider how RoBERTa was pretrained as it boasts better performance than BERT on most tasks.

- Perhaps fine tune task to classify offensive tweets and then use transfer learning from there to classify for hate speech. Research online if this methodology has any merit.

My criterion for success would be to beat the performance of the leading system on the semEval competition on hate speech – which is the contest this task is based off. The highest F score attained was on an SVM with an RBF kernel which achieved ~0.65. This benchmark does not seem that high at a glance, but other systems with much more advanced methods, even with BERT like architecture, failed to beat this score. My hypothesis for this is that the tweets in this database contain many of the terms associated with hate speech, but aren't hate speech because of the context of the tweet. It isn't a clear lexical difference in the sample of tweets between not hate speech and hate speech like the analyicsvidhya competition mentioned above.

I also plan in gaining more insight and understanding into BERT and what methods of utilizing it, achieve the best results. A comprehensive study on this accompanied with visualisations is also a goal of mine. An in-depth error analysis on what type of tweets my

final model's false positive and false negative tweets are will also need to be included so I know where I can improve.

Work Plan

Month	Phase	More details	Project Deliverables
December	Learn more about tensorflow library and develop better evaluation metrics and methods (cross-validation). Experiment with huggingface's pytorch BERT library	Research problem to get a more cohesive strategy moving forward based off of research paper findings online. Implement the metrics of F-score, precision and recall into my sytsem	
January	Improve pre-processing of data. Obtain more hate speech data (scarce and hard to find online) by performing unsupervised data augmentation (UDA). Convert data to tensorflow dataset for better TPU compatibility	Pre-processing methods include; hashtag segmentation, punctuation removal and translating emojis into text. Examples on how to perform UDA are detailed on their github, likewise for converting data to tensorflow dataframe	
Febuary	Collect tweets from twitter (in-domain) and tweets from hate speech datasets (within task) and other related tasks to further pre-train BERT	This method has been detailed in research papers so it's not a novel idea, there will be explanations online on how to do this and the best strategy to employ. Look to the RoBERTa paper for insight into how it was pre-trained also	
March	Find the best method (CNNs, bi-LTSM etc.) to fine tune BERT for hate speech classification.	Much more research needed here. Perhaps compare model performances with various types of SOTA models	
April	Evaluate results and complie final report		Dissertation (40%) and demonstrate final system (40%)

Link to the github repo (HTTPS):
(https://gitlab.eeecs.qub.ac.uk/40156103/CSC3002_Detecting_Hate_Speech_On_Social_Media)

Link to the github repo (SSH):
(git@gitlab.eeecs.qub.ac.uk:40156103/CSC3002_Detecting_Hate_Speech_On_Social_Media.git)

References

Derald Wing Sue, Christina M Capodilupo, Gina C Torino, Jennifer M Bucceri, Aisha Holder, Kevin L Nadal, and Marta Esquilin. 2007. Racial microaggressions in everyday life: implications for clinical practice. American Psychologist, 62(4).

John T. Nockleby, Hate Speech. In Encyclopedia of the American Constitution (2nd ed., edited by Leonard W. Levy, Kenneth L. Karst et al., New York: Macmillan, 2000), pp. 1277-1279 (see http://www.jiffynotes.com/a_study_guides/book_notes/eamc_03/eamc_03_01193.html)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar S. Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. ArXiv, abs/1907.11692, 2019.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019). Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Predicting the type and target of offensive posts in social media. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 1415–1420, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In Proceedings of ICWSM.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. AAAI.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL Student Research Workshop, pages 88–93, San Diego, California, USA, June. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In Advances in neural information processing systems, pages 3294–3302.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? arXiv preprint arXiv:1905.05583, 2019.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised Data Augmentation. arXiv e-prints, art. arXiv:1904.12848, Apr 2019

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In Advances in Neural Information Processing Systems, pages 6294–6305