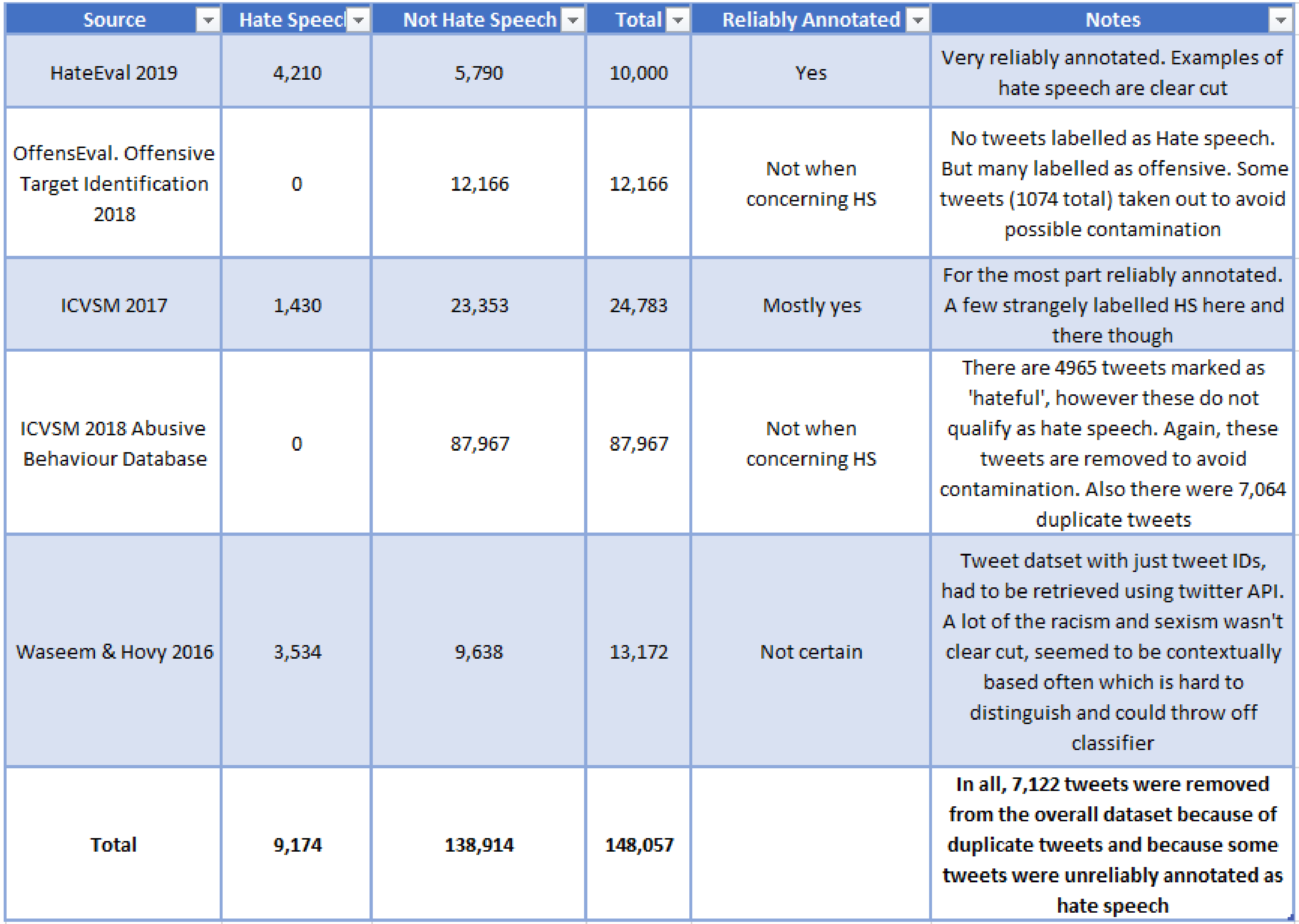
CSC3002 Initial Demo

Assembling the Data

- For my project, as it is a neural network based task, My initial goal was to source as many hate speech datasets off twitter as possible.
- This project was loosely based off of the competition on HatEval 2019 on codalab, so I started there.
- After further research I gathered more datasets from widely respected datasets in the NLP community online. Some datasets I found did not match the criterion for hate speech, but nevertheless I kept them as they could be useful down the line for my task – (namely the fact that some were labelled as offensive and if my model learned the distinction between what is offensive and what is hate speech then it could avoid many false positives).
- The datasets I used are in a table on the slide overleaf

Overview of the Data

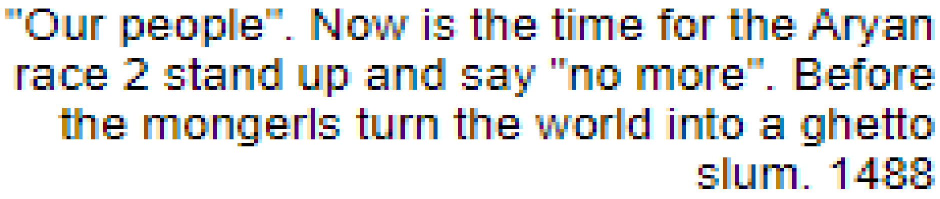


Source	Hate Speec	Not Hate Speech	Total	Reliably Annotated	Notes
HateEval 2019	4,210	5,790	10,000	Yes	Very reliably annotated. Examples of hate speech are clear cut
OffensEval. Offensive Target Identification 2018	0	12,166	12,166	Not when concerning HS	No tweets labelled as Hate speech. But many labelled 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**

What do you mean by reliably annotated?

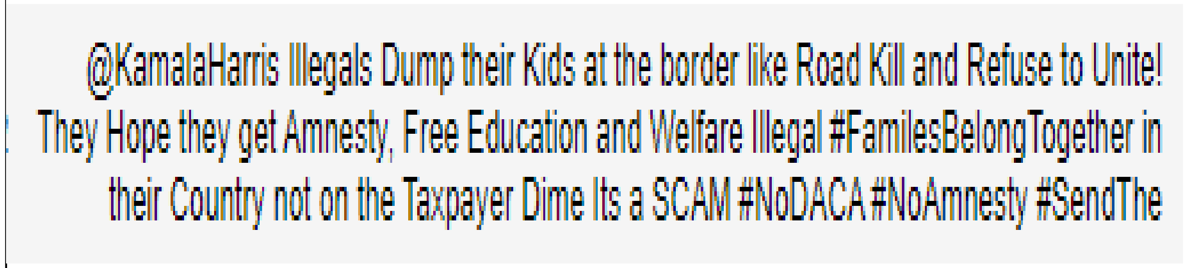
Reliably Annotated as HS

- ISVSM 2017:



"Our people". Now is the time for the Aryan race 2 stand up and say "no more". Before the mongerls turn the world into a ghetto slum. 1488

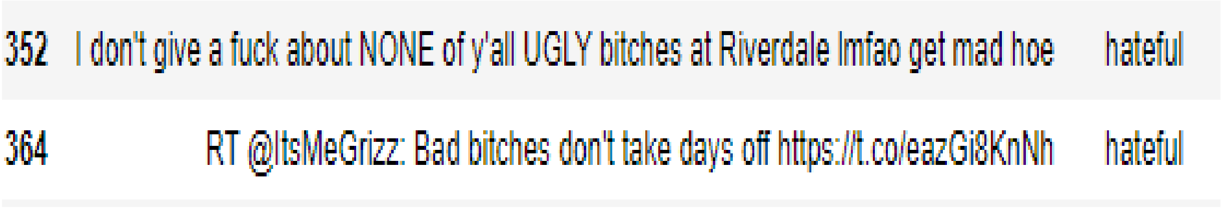
- HatEval 2019:



@KamalaHarris Illegals Dump their Kids at the border like Road Kill and Refuse to Unite! They Hope they get Amnesty, Free Education and Welfare Illegal #FamilesBelongTogether in their Country not on the Taxpayer Dime Its a SCAM #NoDACA #NoAmnesty #SendThe

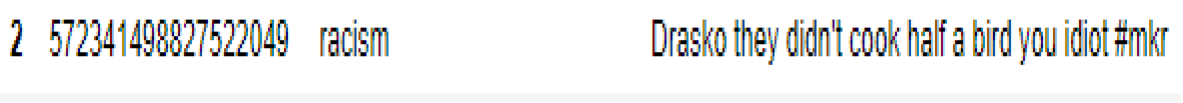
Unreliably Annotated as HS

- ICVSM 2018:



352	I don't give a fuck about NONE of y'all UGLY bitches at Riverdale lmfao get mad hoe	hateful
364	RT @ItsMeGrizz: Bad bitches don't take days off https://t.co/eazGi8KnNh	hateful

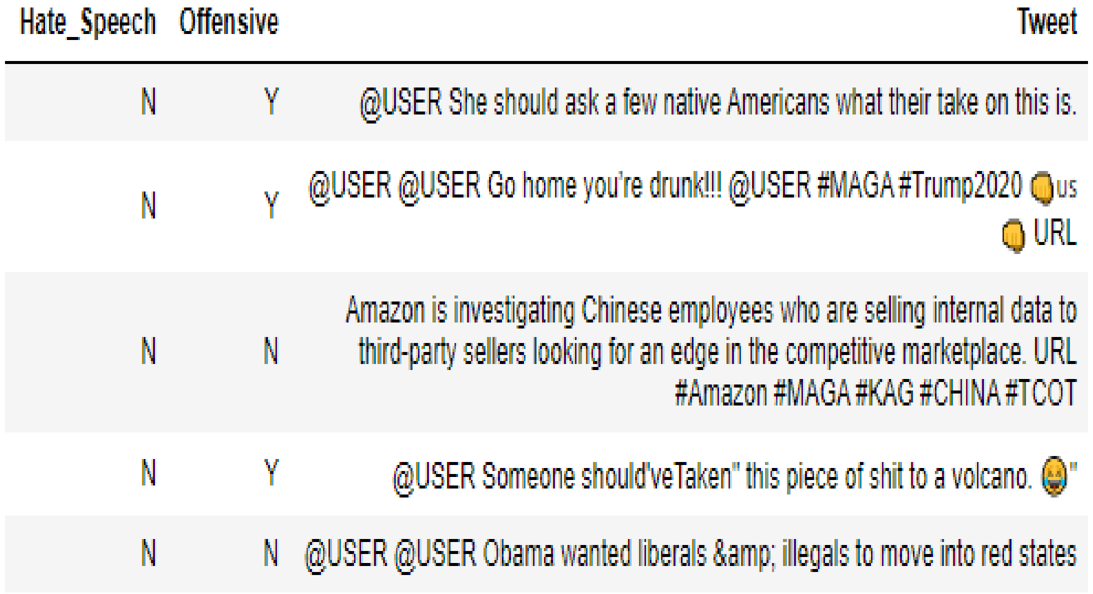
- Waseem & Hovy 2016:



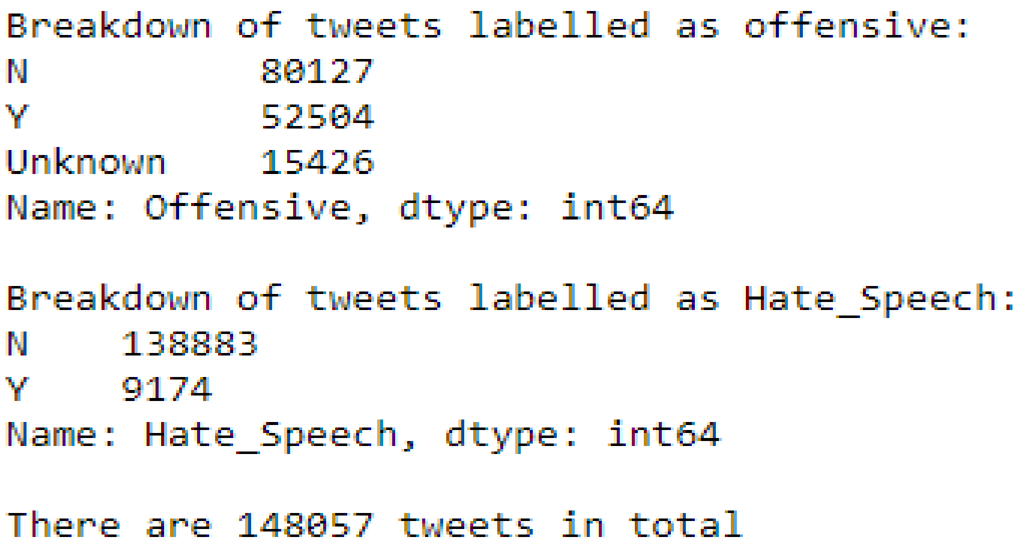
2	572341498827522049	racism	Drasko they didn't cook half a bird you idiot #mkr

How my final, combined dataset looks

- The dataset has 148,057 tweets in total. The tweets have a binary classification as hate speech or not hate speech and a three-way classification of the offensive column as offensive, not offensive or unknown – this is because the semeval 2019 database does not have a column denoting whether a tweet is offensive or not.



Hate_Speech	Offensive	Tweet
N	Y	@USER She should ask a few native Americans what their take on this is.
N	Y	@USER @USER Go home you’re drunk!!! @USER #MAGA #Trump2020 👊us 👊 URL
N	N	Amazon is investigating Chinese employees who are selling internal data to third-party sellers looking for an edge in the competitive marketplace. URL #Amazon #MAGA #KAG #CHINA #TCOT
N	Y	@USER Someone should'veTaken" this piece of shit to a volcano. 😂"
N	N	@USER @USER Obama wanted liberals &amp; illegals to move into red states



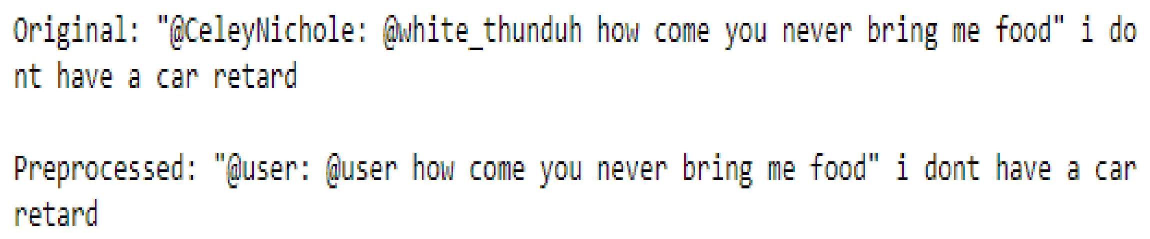
```
Breakdown of tweets labelled as offensive:
N          80127
Y          52504
Unknown    15426
Name: Offensive, dtype: int64

Breakdown of tweets labelled as Hate_Speech:
N    138883
Y    9174
Name: Hate_Speech, dtype: int64

There are 148057 tweets in total
```

Text cleaning and Pre-processing

- Admittedly, I jumped the gun in this regard as I started pre-processing my tweets in my database, before knowing how this may affect performance down the line.
- For this reason, onwards I'll be performing pre-processing in the google colab notebook where I build and evaluate my model. This is so I can judge what pre-processing methods aid performance and what methods hinder performance
- I've built a pre-process method and the next slide demonstrates it's many functions in action.
- Also I've created a function that replaces emojis with words, but it does make the sequence length quite long for some emoji laden tweets, so it remains to be seen if this method is used

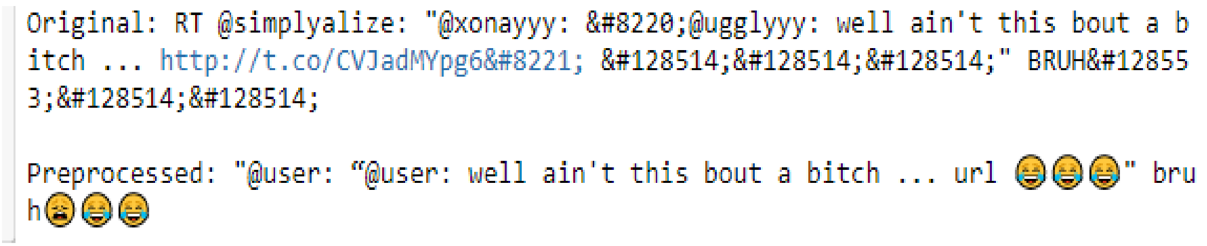
Pre-Process Demonstration

- On mentions:



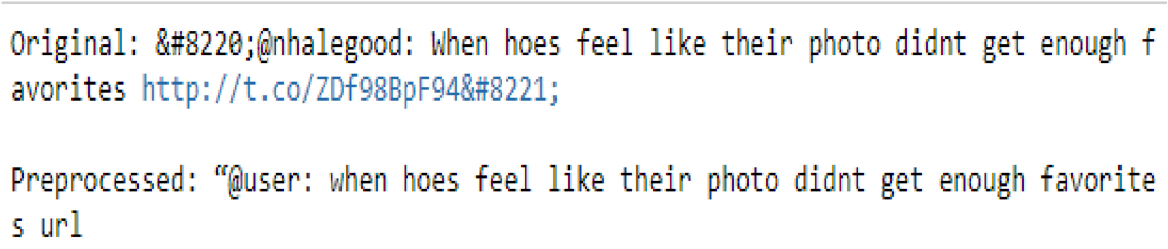
```
Original: "@CeleyNichole: @white_thunduh how come you never bring me food" i do
nt have a car retard

Preprocessed: "@user: @user how come you never bring me food" i dont have a car
retard
```

On retweets:



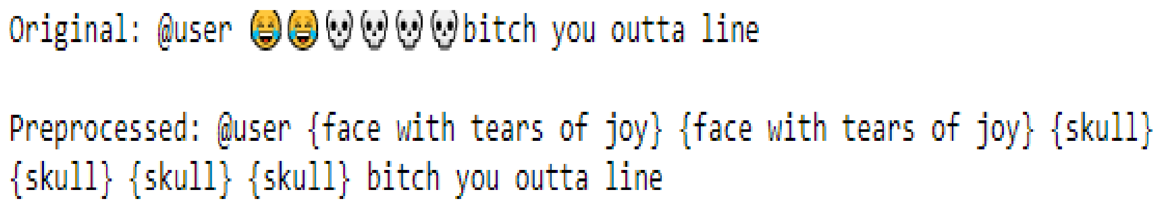
```
Original: RT @simplyalize: "@xonayyy: &#8220;@ugglyyy: well ain't this bout a b
itch ... http://t.co/CVJadMYpg6&#8221; &#128514;&#128514;&#128514;" BRUH&#12855
3;&#128514;&#128514;

Preprocessed: "@user: “@user: well ain't this bout a bitch ... url 😂😂😂" bru
h😩😂😂
```

- On URLs and tweets with Unicode:



```
Original: &#8220;@nhalegood: When hoes feel like their photo didnt get enough f
avorites http://t.co/ZDf98BpF94&#8221;

Preprocessed: “@user: when hoes feel like their photo didnt get enough favorite
s url
```

Replacing emojis with text:



```
Original: @user 😂😂💀💀💀💀bitch you outta line

Preprocessed: @user {face with tears of joy} {face with tears of joy} {skull}
{skull} {skull} {skull} bitch you outta line
```

- Also included in the pre-processing is making words uncased, removing excess whitespace and replacing & with and. Might experiment with not using common tags for URLs and users, but instead removing them altogether. Maybe this strategy is unwise though because these things can give context to words.

Moving Forward…

- I retained the “offensive” column for my final database, as my strategy at the time was to use transfer learning from BERT to classify if tweets were offensive or not, and then from the offensive tweets learn which of them were hate speech.
- I believed the distinction between hate speech and offensive tweets was critical and I could avoid many false positives by teaching my neural network the difference between the two. However I’m not sure how useful this strategy is going forward, but I’ll keep it in mind.
- For now my goal will be to collect as many tweets as possible for my new strategy of developing a model like BERT .
- Instead of being pre-trained on proper English and grammar via the entire Wikipedia corpus like the original BERT, instead it will be a fine tuned version of BERT pre-trained on millions of tweets so it can understand the vernacular of twitter.

Google Colab – Fine Tuning BERT on TPUs

- Google Colab kindly grants us memory access to both cloud TPUs and GPUs on it’s colab service. So it is necessary to train my neural network model there as my local machine does not have the required memory
- TPUs are a tensorflow only accelerator for deep learning and the cloud GPUs provided get out of memory errors when they attempt to fine tune my model on top of BERT Large – which has 24-layers, 1024-hidden, 16-heads, 340M parameters.
- We make sure our runtime is connected to a TPU and we store it’s address which we’ll use later
- Also, notably we import bert-tensorflow.

Online storage – Google bucket

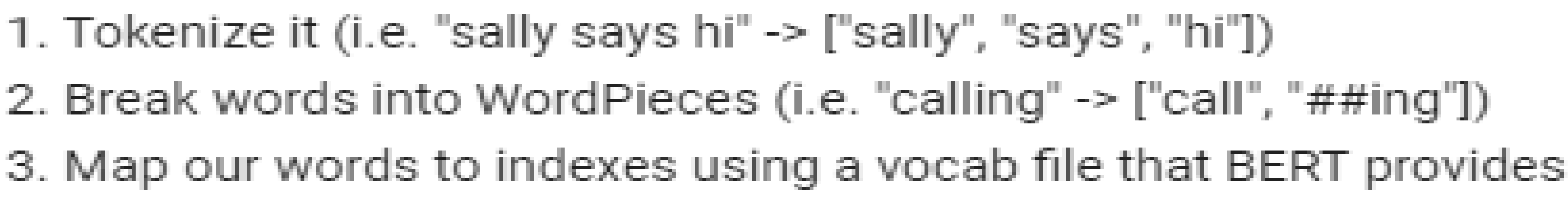
- We have various versions of BERT stored in online storage in our google bucket. We store our output in a sub-directory /output because we need our output to be stored in the same directory as BERT for our functions to work later
- We can choose to delete the directory and create a new one if we wish to overwrite an existing fine-tuned model. Likewise we can load an existing model and train further

The Data + Analytics Vidhya competition

- Because of the vast size of my data and also the uncertainty surrounding the authenticity of evaluating on it – (as it came from many different sources with differing opinions on what constitutes hate speech), I decided to enter an Indian based, twitter hate speech detection competition.
- The benefit of this is that it’s a task exactly like mine and I could truly evaluate the performance of my model on an unlabelled test dataset by submitting my prediction results online and comparing my results to other contestants
- This set was unreliably annotated in my opinion, but nevertheless it would give me an idea of performance. It contained around 32,000 tweets with 2,242 annotated as hate speech. (7% of overall dataset)

BERT Pre-processing

- We use the inbuilt BERT tokenizer to pre-process our data so it matches the data BERT was trained on, the tokenizer does the following to our tweets:



1. Tokenize it (i.e. "sally says hi" -> ["sally", "says", "hi"])
2. Break words into WordPieces (i.e. "calling" -> ["call", "##ing"])
3. Map our words to indexes using a vocab file that BERT provides

- We then convert our tweets into features so BERT can interpret them. The inbuilt function 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

Create Fine-tuned layer and build overall model

- We then create our own fine-tuned layer, which is trained exclusively on our task. It is only a single layer network so there’s definitely room for improvement. This network, through transfer learning, obtains the knowledge BERT has on NLP and uses it to it’s advantage.

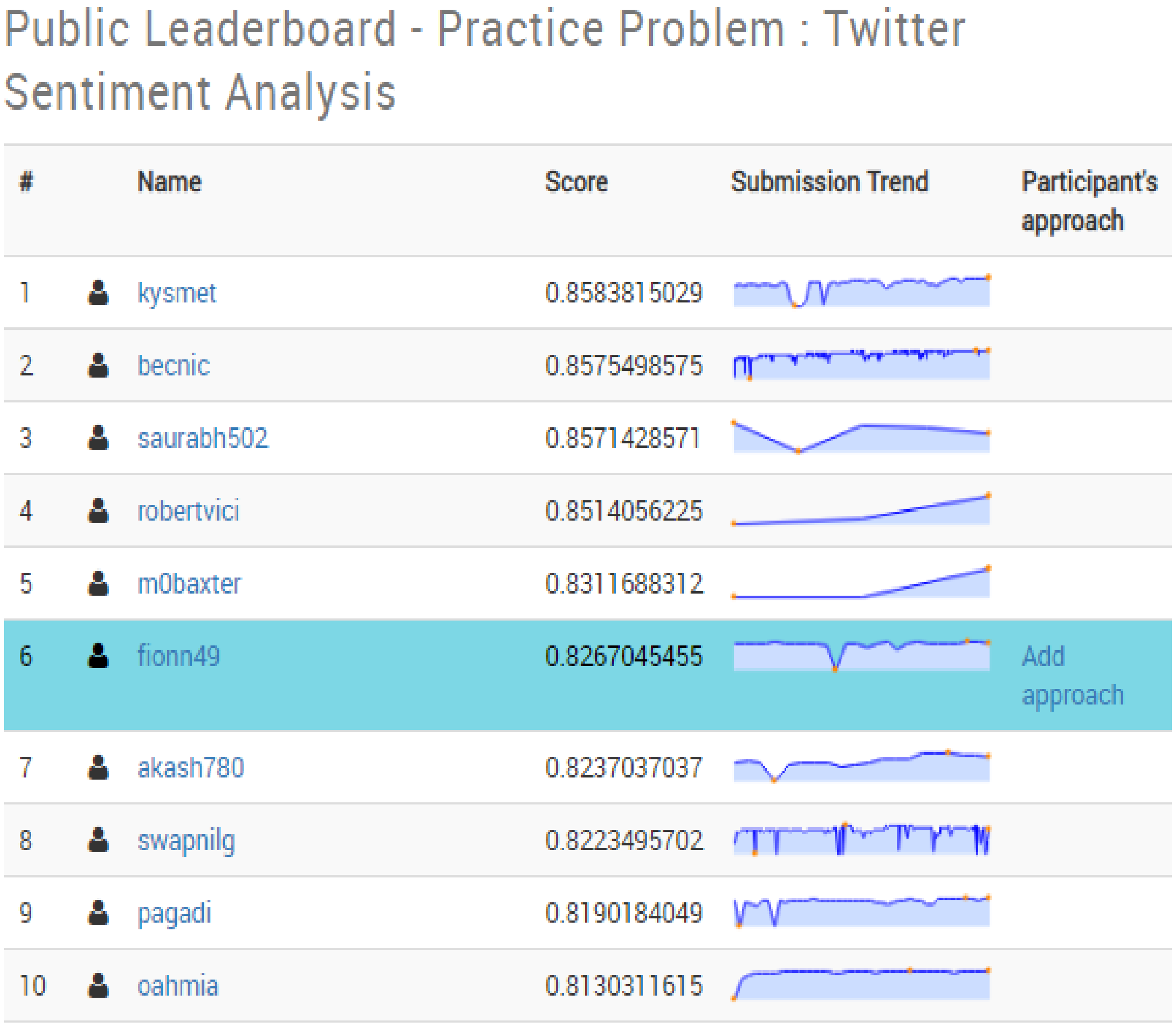
- We have a model function so that we can code for different behaviours the network may have, i.e. whether it’s training, evaluating or predicting
- We can tune our model hyperparamters if we wish

Training and evaluating BERT

- Using inbuilt functions, we specify that this model is to be trained on TPUs so the training can be optimised for it. We also specify our hyperparameters accordingly
- Training can take anywhere between 15 mins and an hour on a TPU
- We can also just load an existing fine tuned mode if we skip the training bit and we have a model in our output directory already
- Evaluating takes a few minutes on average, I’m working on making it output other metrics than accuracy as we all know this is a flawed measure or performance. Especially when the examples of hate speech in this dataset are relatively scarce (~7%). Still the results are promising

Submission to the competition

- Because I can’t get the metrics I’d like for my model, and also because it’s available as often as I’d like, I use the submission to the competition as a temporary method of evaluation.
- I plan to use cross validation in the future also
- But anyways alongside are the current standings for the competition

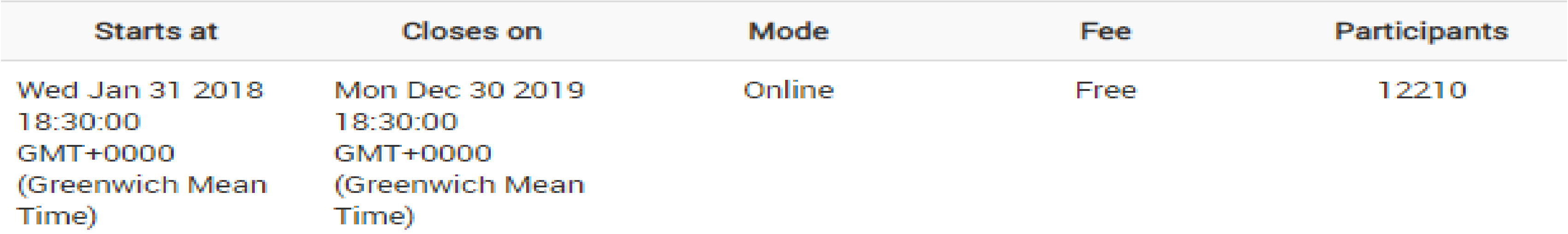


Public Leaderboard - Practice Problem : Twitter Sentiment Analysis

#		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		

Results Discussion

- Currently I’m sixth in this competition which has 12,210 participants, it’s been going since January 2018 and ends at the end of the year. My guess is that it’s for novices, still it’s quite promising



Starts at	Closes on	Mode	Fee	Participants
Wed Jan 31 2018 18:30:00 GMT+0000 (Greenwich Mean Time)	Mon Dec 30 2019 18:30:00 GMT+0000 (Greenwich Mean Time)	Online	Free	12210

- This temporary method of evaluation has enabled me to better learn the relationship between a model’s parameters and performance
- I’ve seen performance improvements through tuning parameters and introducing pre-processing techniques.
- Going forward I’ll attempt to add more data from an outside source, or from within through unsupervised data augmentation for training and see if it improves.

Any Questions?