Project Documentation
Racism and hate speech detection in Twitter

Course: DS8013 | **Instructor**: Prof. Kanchana Padmanabhan | **Student**: Raad Al-Husban

Table of contents

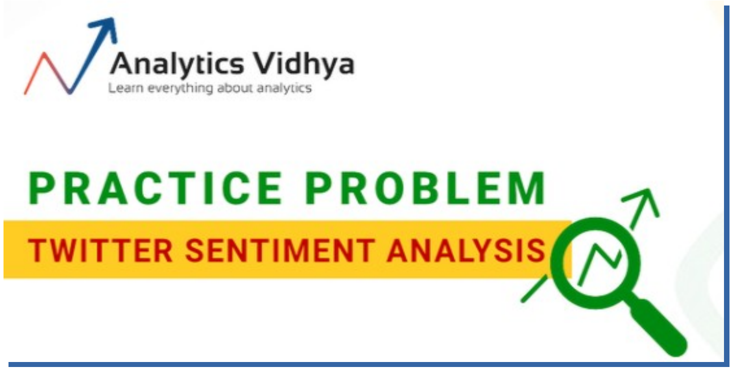
Project Documentation
Racism and hate speech detection in Twitter

Course: DS8013 | **Instructor**: Prof. Kanchana Padmanabhan | **Student**: Raad Al-Husban

1. **Problem statement**

Social media is having an increasing influence on the world, and in the past years we've seen violence instigated by social media. Twitter is a well known platform that helped people spread news around the world in an unprecedented manner. However, at the same pace, hate, violent, and racism was equally propagated by some users of Twitter. Companies such as Twitter and Facebook strive to improve their hate speech / racism detection algorithms to flag and remove any tweets / posts containing such language. Therefore, an efficient design of such model would be beneficial overall to counter the spread of hate speech and racism on social media.

2. **Dataset**



I used a Twitter tweets dataset provided by Analytics Vandhya as a competition. The dataset contains 32'000 labelled tweets which will be used to train and test our model. The highest achieved score (F-1) in the competition is ~87%. The Data set is highly unbalanced – containing only 7% training data with hate speech / racism while the rest is regular tweets.

3. **Literature Review:**

I went thru several blogs and papers to get inspired with the embedding method implemented for this project:

3.1 Contextual Topic Identification – Steve Shao: This blog used an unconventional implementation of BERT text embedding combined with the probabilistic LDA topic assignment. The authoer achieved impressive results in separating clusters of user comments on STEAM – an online game selling platform

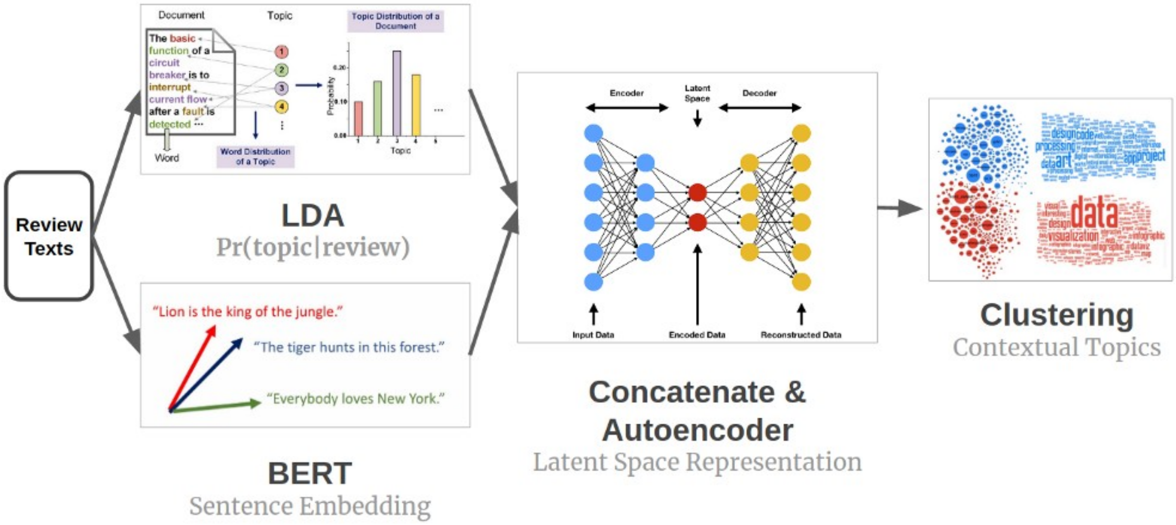
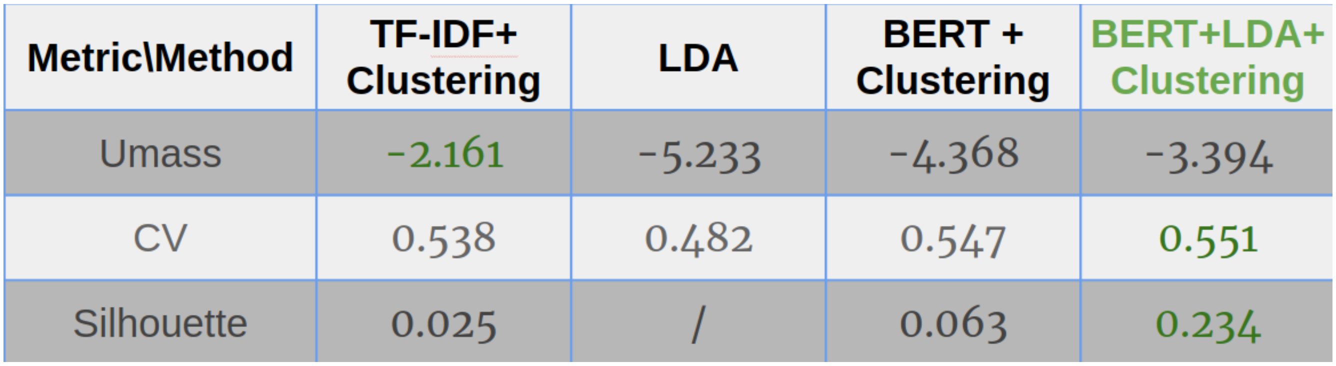

Review Texts
Document
Topic
Topic Distribution of a Document
The basic function of a circuit breaker is to interrupt current flow after a fault is detected ...
Word
Word Distribution of a Topic
Probability
Topic
LDA
Pr(topic|review)
"Lion is the king of the jungle."
"The tiger hunts in this forest."
"Everybody loves New York."
BERT
Sentence Embedding
Encoder
Latent Space
Decoder
Input Data
Encoded Data
Reconstructed Data
Concatenate & Autoencoder
Latent Space Representation
Clustering
Contextual Topics

Figure 1: Contextual Topic Identification - Steve Shao



Metric\Method	TF-IDF+ Clustering	LDA	BERT + Clustering	BERT+LDA+ Clustering
Umass	-2.161	-5.233	-4.368	-3.394
CV	0.538	0.482	0.547	0.551
Silhouette	0.025	/	0.063	0.234

Figure 2: Evaluation Metrics for 4 methods of topic modelling - Steve Shao

3.2 Topic Modelling with BERT **– Maarten Grootendorst:**

Another inspiring blog on how to average probabilistic and context aware methods for easily interpreter topics. What was special about this blog is their use of UMAP for dimensionality reduction and eventually visualization in a similar fashion like TSNE is used for visualizing high-dimensional data.

The author used HDBSCAN instead of the more traditional K-Means clustering approach that normally used in clustering problems.

The author tested this approach on the famous 20NewsGroup dataset often used to benchmark classification and clustering models.

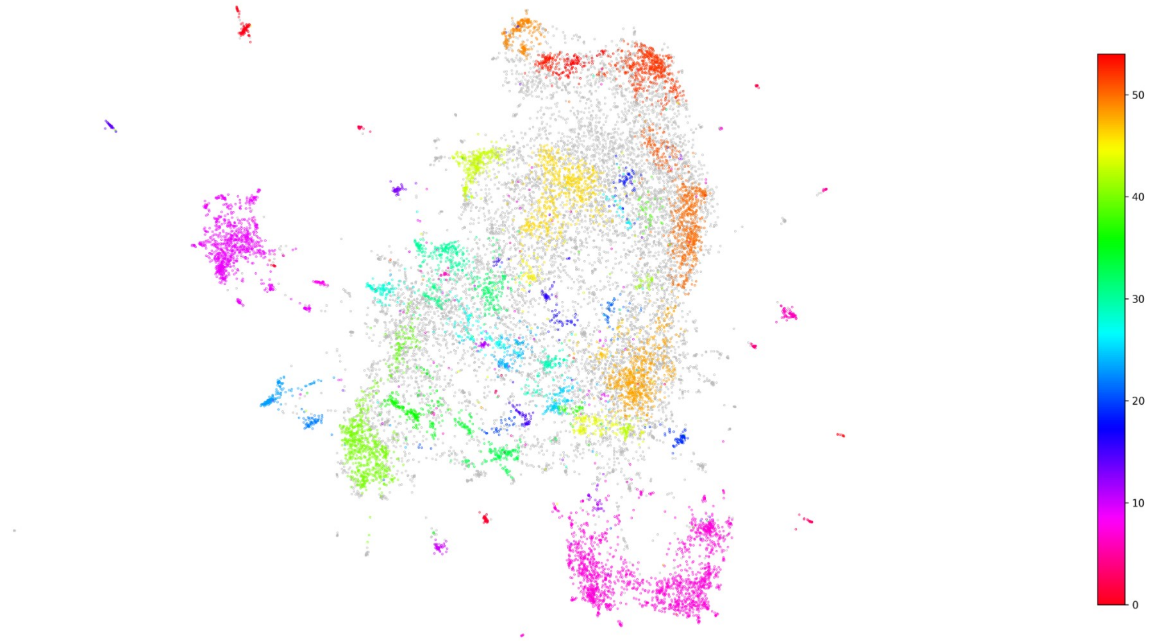
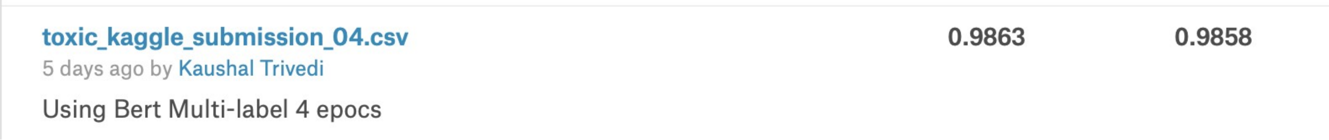


Figure 3: Topics visualized by reducing sentence embeddings to 2-D - Maartin Grootensdorst

The model showed the ability to extract clear keywords for various topics from the 20 NewsGroup dataset.

3.3 Multi-label Text Classification using BERT-The Mighty Transformer – Kaushal Trivedi:

This blog was a very useful guide on how to implement pre-trained BERT models using the HuggingFace Library and apply it to multi-label text classification problems. The author used ROC-AUC evaluation metric, which is a standard for Kaggle competitions according to the author. The blog ran its model with impressive results on the Toxic Comment Classification Challenge to benchmark the performance of the BERT model.



toxic_kaggle_submission_04.csv 0.9863 0.9858
5 days ago by Kaushal Trivedi
Using Bert Multi-label 4 epocs

Figure 4: Kaggle Competition results submitted by Kaushal Trivedi

3.4 Text Classification with TF: BERT, XLNet – David Mraz

The author compared older approaches for text classification using language methods based on RNNs such as LSTMs and demonstrated how the fail to capture longer contextual dependencies compared to transformer architecture.

The blog doesn't use a pre-trained model but rather trains a BERT model from scratch but also shows how a pre-trained model could be fine-tuned.

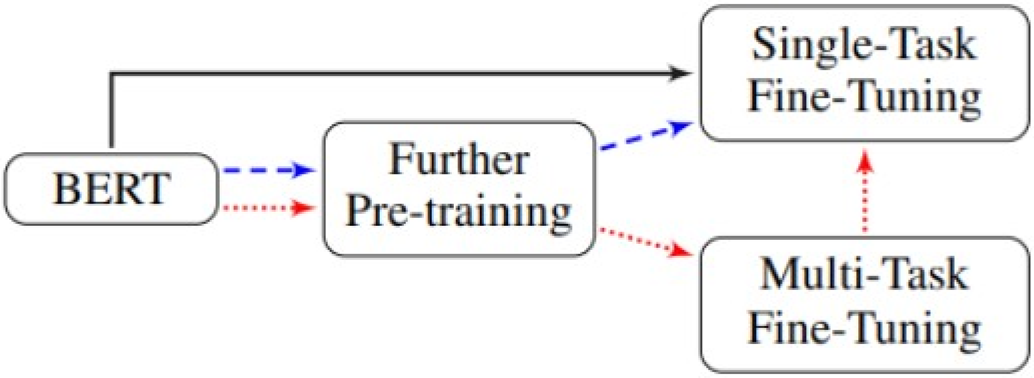
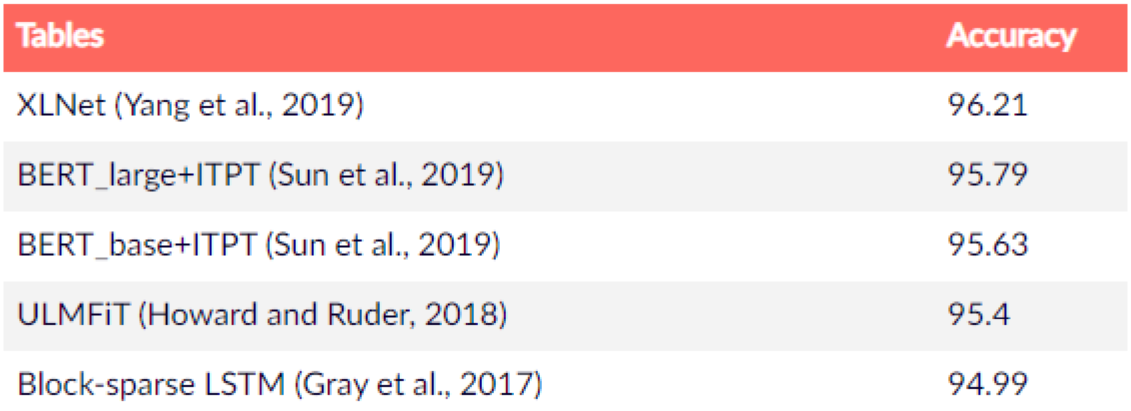


Figure 5: Three general ways for fine-tuning BERT - David Mraz

The author tested his model on the IMDB dataset and achieve about 90% classification accuracy compared to other models shown-below.



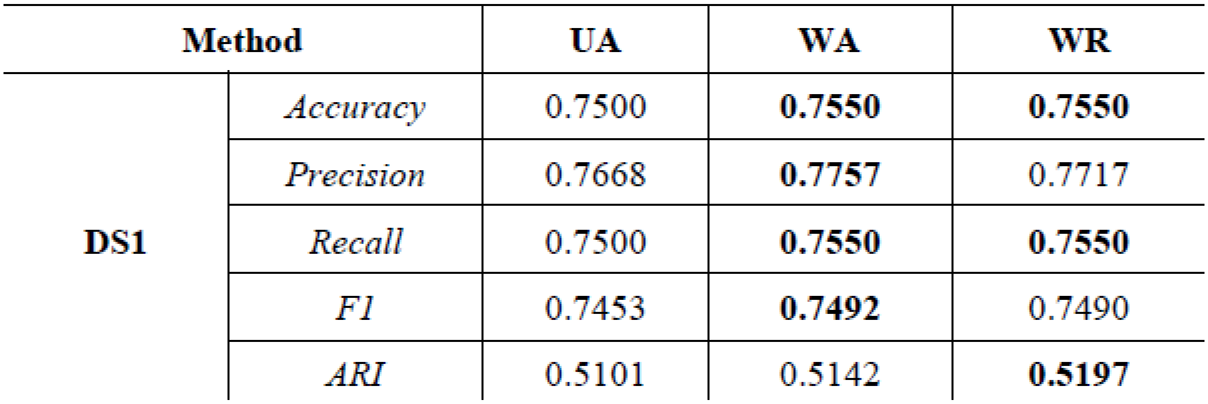
Tables	Accuracy
XLNet (Yang et al., 2019)	96.21
BERT_large+ITPT (Sun et al., 2019)	95.79
BERT_base+ITPT (Sun et al., 2019)	95.63
ULMFiT (Howard and Ruder, 2018)	95.4
Block-sparse LSTM (Gray et al., 2017)	94.99

Figure 6: Model accuracy on IMDB dataset - nlpprogress.com

3.5 A Text Document Clustering Method Based on Weighted BERT Model – Yutong Li, Jingling Wang:

This paper, which was accepted in IEEE's ITNEC 2020, proposed an interesting approach of using a modified BERT method by using Part of Speech (PoS) tags to apply a weighing method for document embedding prior to clustering. The authors applied their model on the Reuters-21578 and used four different subsets with 4,5,8, and 16 topics. The have compared a standalone BERT and two variants of their weighing approachs.

This resulted in marginal improvement in clustering performance. The main take-away from this paper was that every single 'tweak' we make to transformer models can improve the performance of a given model on clustering and equally text classification problems.



Method		UA	WA	WR
DS1	*Accuracy*	0.7500	**0.7550**	**0.7550**
	Precision	0.7668	**0.7757**	0.7717
	Recall	0.7500	**0.7550**	**0.7550**
	F1	0.7453	**0.7492**	0.7490
	ARI	0.5101	0.5142	**0.5197**

Figure 7: Performance of text clustering with 4 topics - Yutong Li et al

3.6 Sentiment Analysis with BERT and Transformers by Hugging Face using PyTorch and Python – Venilin Valkov:

This useful blog guides on how to use BERT to conduct sentiment analysis on Google Play Reviews, data preprocessing, and how to use transfer learning using the HuggingFace library.

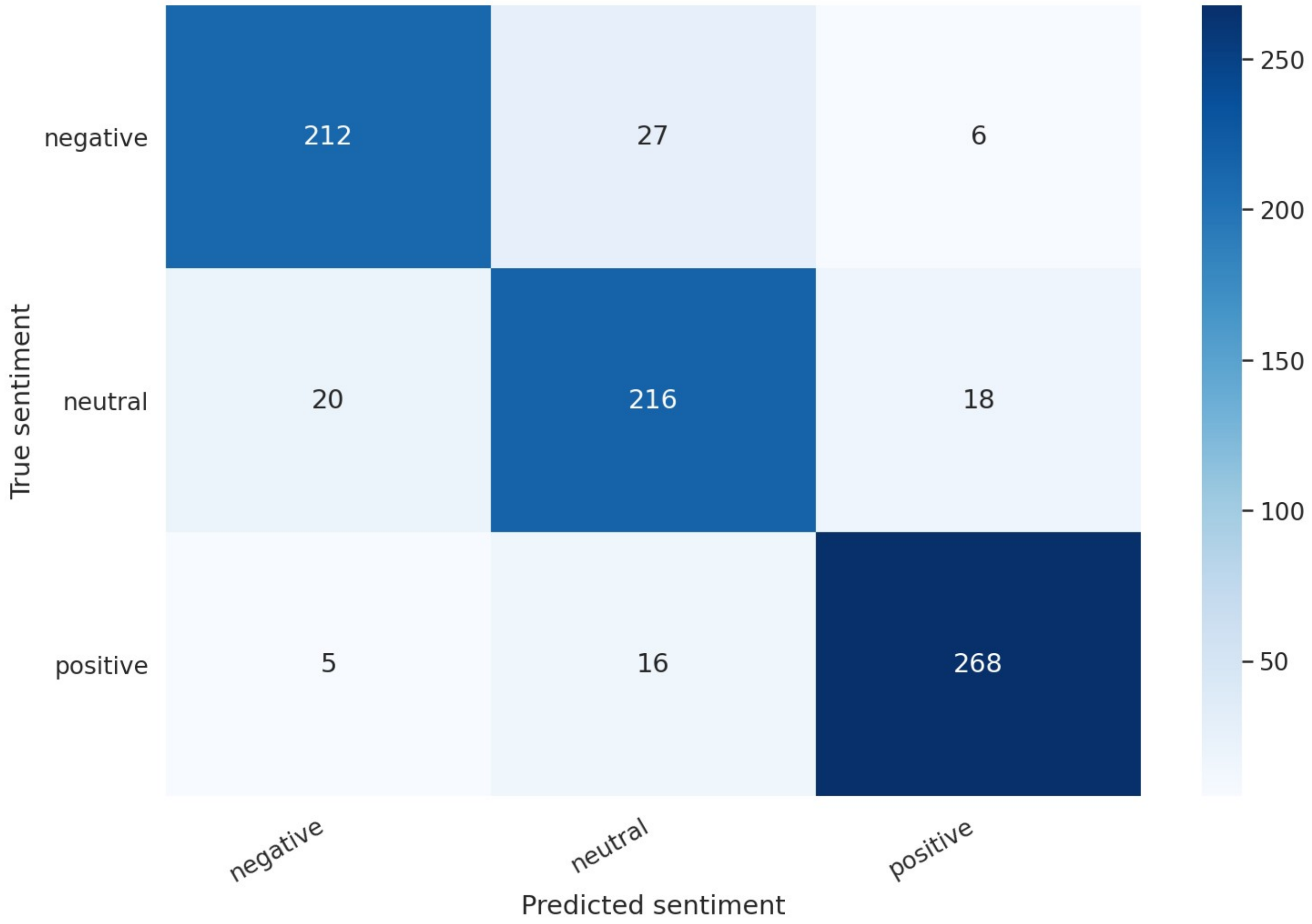


Figure 8: Confusion matrix of the sentiment classification model – Venilin Valkov

Valkov’s model did a good job classifying positive and negative sentiments, however it had more challenge doing the job for neutral sentiments.

3.7 BERT Explained – Rani Horev:

Obviously its best to start from Google's original BERT publication, however this blog walks thru the architecture and use of BERT in a simplified manner. It compares its significant advantages to all previous language models and they way its architecture is easily adaptable to various language problems.

It also walks the reader thru differences of using untrained and pre-trained models in addition to fine-tune, which is essentially training only the last layer of the architecture with freezing the weights of previous layers.

3.8. Text classification models for the automatic detection of nonmedical prescription medication use from social media – Mohammed Al-Garadi et al.

The authors discuss using pre-transfer learning with BERT, RoBERTa, XLNet, Albert, and DistilledBERT) and compare them with conventional Machine Learning Models on text classification problems.

The authors concluded that the Deep Learning architecture performed significantly better than traditional models - ((F1-score [95% CI]: 0.67 [0.64–0.69] vs. 0.45 [0.42–0.48]). They also noted that deep learning architecture required less annotated data and generalized better and was better suited to classify non-medical prescriptions due to the unique way medical transcriptions are written.

One contradictory claim by the author is that the Deep Learning architecture required less labelled data as DNNs normally require a lot of training samples compared to conventional machine learning models.

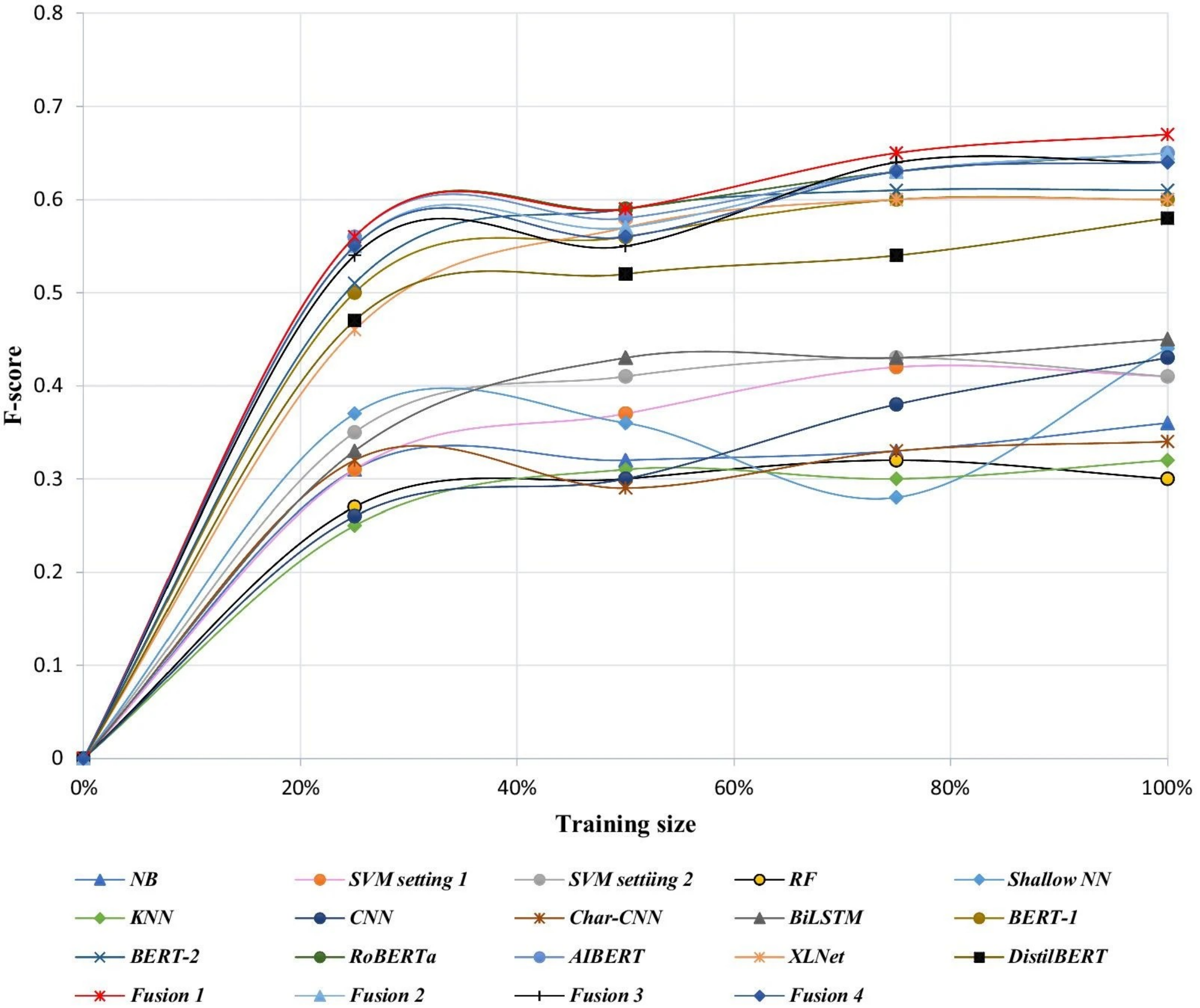


Figure 9: Learning Curve at different amount of training data - Al-Garadi et al

3.9 Twitter Tweet Classification Using BERT – Xiang Yutang

The author attempts to classify tweets for disaster response purposes to make tweets more useful to first-responders, particularly during disasters or violent events. The model is attempting to classifgy tweets to seven categories:

- not informative
- caution and advice
- affected individuals
- infrastructure and utilities damage
- donations and volunteering
- sympathy and support
- other useful information

They use BERT embedding and compare the same method with BiLSTM and GloVE embeddings to compare performance metrics.

The BERT model outperformed the other two models in all classification categories:

Table 1: BERT vs LSTM vs BERT LSTM

Model	Accuracy	Matthews Correlation Coefficient
Baseline	0.6323	0.5674
Base BERT	0.6948	0.6401
LSTM BERT	0.6853	0.6311

3.10 Document Embedding Techniques – Shay Palachy:

The author provides a very useful and easy to read summary of many unsupervised and supervised text embedding methods. The blog discusses:

- Classic methods such as TF-IDF, LDA and LSA
- Unsupervised methods such as: Doc2Vec, Sent2Vec, SBERT, FastSent
- Supervised Embedding methods such as: Deep Semantic Similarity Model (DSSM), GenSen, Universal Sentence Encorder.

While transformer based models may seem to be the models of choice, the author concludes that there is no clear leader for specific tasks. Many trials have to be performed to pick the best model as each problem is very specific.

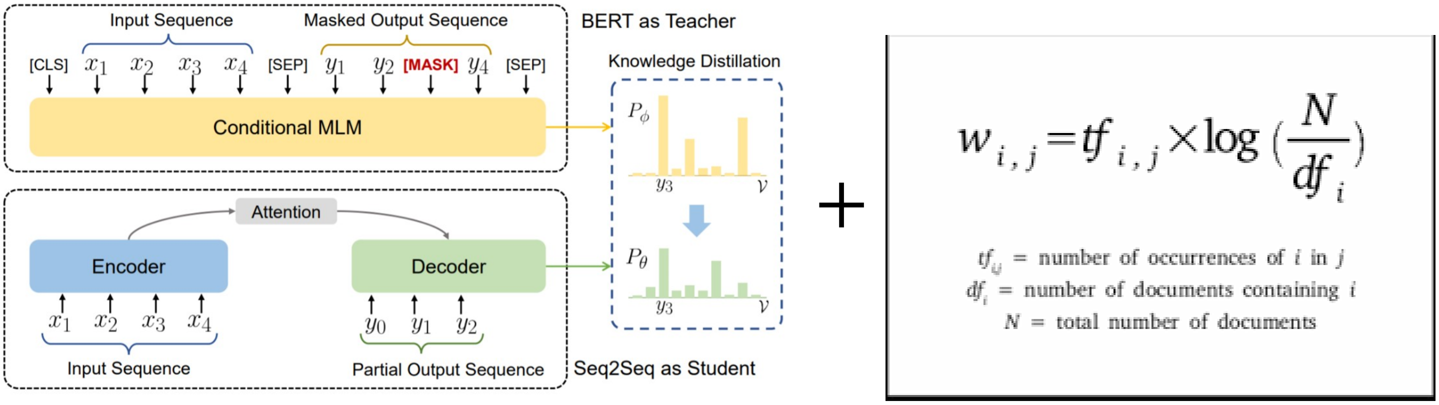
4. Technologies used:

4.1: Packages

- NLTK
- Keras
- Hugging face (Destilled-BERT)
- Scikit-learn
- Numpy
- Pandas
- Matplotlib

4.2 Models / Architecture

While there have been many attempts at this problem using tokenization techniques and using Bag of Words and Term Frequency – Inverse Term Frequency (TF-IDF) embedding methods, I used a different approach for embedding the data. A pretrained Distilled BERT model embedding as a first step, and separately use TF-IDF encoding and combine features of both methods. Prior to concatination of both feature vectors, TF-IDF features were converted to dense using the built-in feature in the Sklearn's implementation of TF-IDF '.todense()' for efficiency reasons. Afterwards, the resulting data will be passed into a Densly Connected Deep Neural Network for classification as described below:



Drawing 1: Merged BERT-TFIDF embedding

5. **Preprocessing**

The input tweets were preprocessed with the following methods:

- Removal of 1 charachter words
- Removal of digits
- Lemmatization (Proved to be more efficient than stemming in preserving contextual value)
- Removal of stop words
- Removal of special charachters such as @, #, ! , etc.

Note that despite many sources pointing the deep learning architecture doesn’t require pre-processing of text data, I found the contrary. Despite that DNNs are inherently designed to extract features, text pre-processing still enhances the model’s performance. This could be related to BERT embeddings being more efficient when the input text is cleaned up of words and characters that have little or no contextual value.

6. Architecture Design, fine-tuning, and discussion

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
======
dense (Dense)                  (None, 128)               4214272
_________________________________________________________________
dropout (Dropout)              (None, 128)               0
_________________________________________________________________
dense_1 (Dense)               (None, 128)               16512
_________________________________________________________________
dropout_1 (Dropout)            (None, 128)               0
_________________________________________________________________
dense_2 (Dense)               (None, 128)               16512
_________________________________________________________________
dropout_2 (Dropout)            (None, 128)               0
_________________________________________________________________
dense_3 (Dense)               (None, 1)                 129
=================================================================
======
Total params: 4,247,425
Trainable params: 4,247,425
Non-trainable params: 0
```

The model structure is rather simple (cf drawing 3): It contains two hidden layers with RELU activation functions. Dropout of 40% are added after each layer with an L2 regularization of 1% at each of the hidden layers. The final layer is sigmoid activated for our binary classification problem.

I have experimented with L2 and drop out ratios and found these to provide the best accuracy and f1-score. Moreover, another unconventional fine-tuning method was to expirement with testing the sigmoid activtication threshold vis-a-vis prediction with 1 or 0. This was done thru a simple for loop from 1 to 100 that tested the f-1 score at each run and the optimal ‘threshold’ was actually found at 0.33 instead of the standard 0.50 cut-off. This increased F1-Score by about 2.5% (c.f. Fig.10)

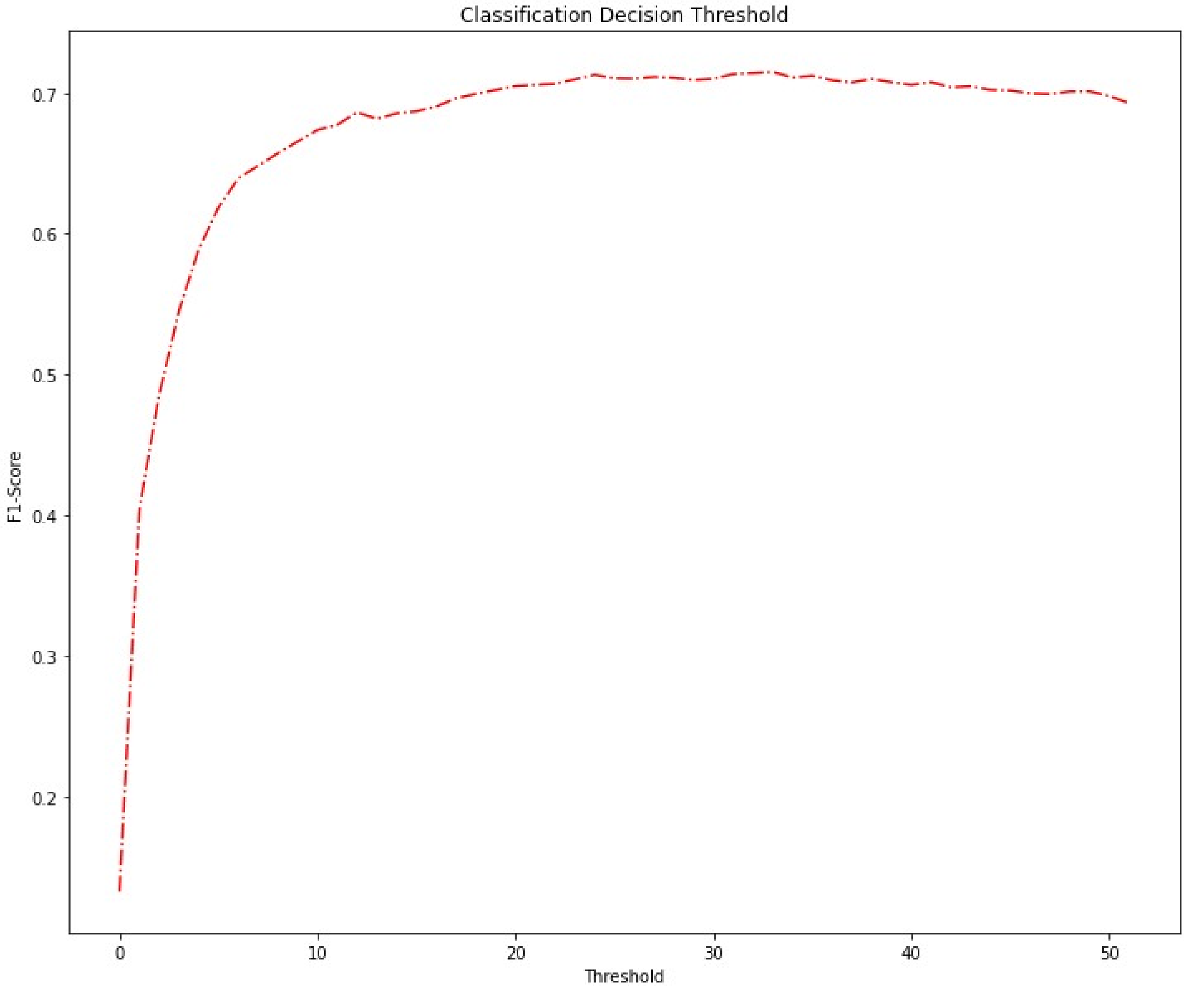


Figure 10: Classification Decision Threshold

Fig.10 is discussed in the previous page. It was an efficient way to boost the performance of our model. Although this approach may not be a good fit for other classification problems.

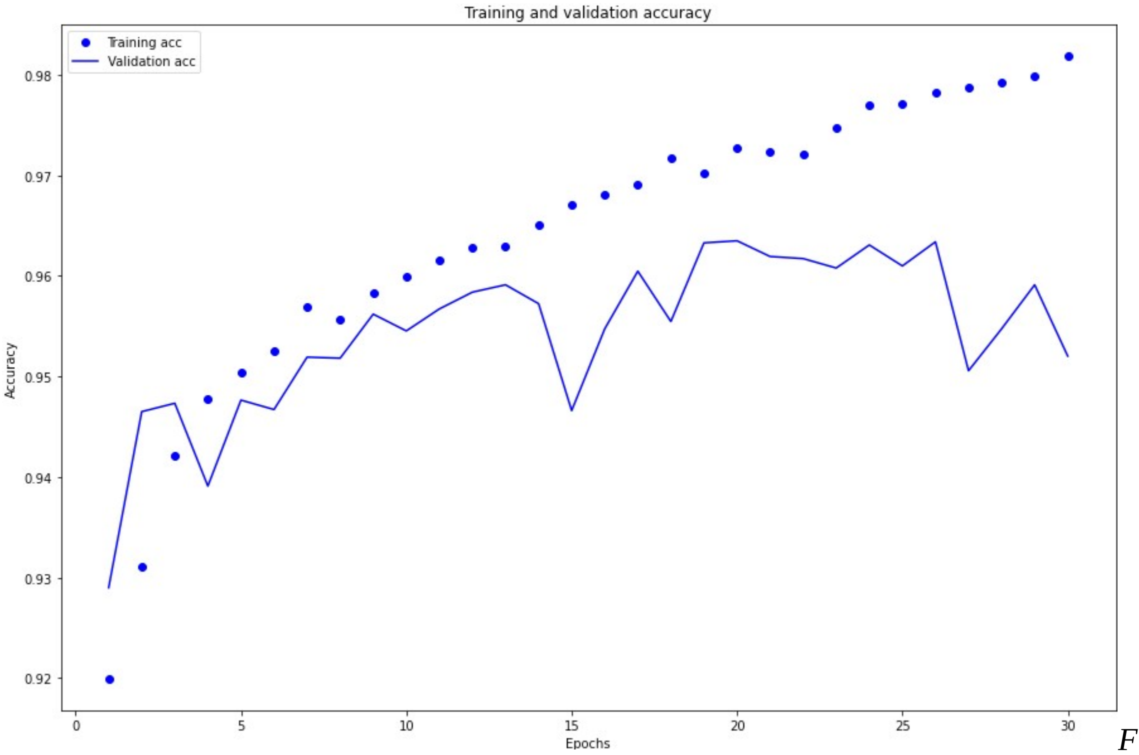


Figure 11: Training and Validation Accuracy

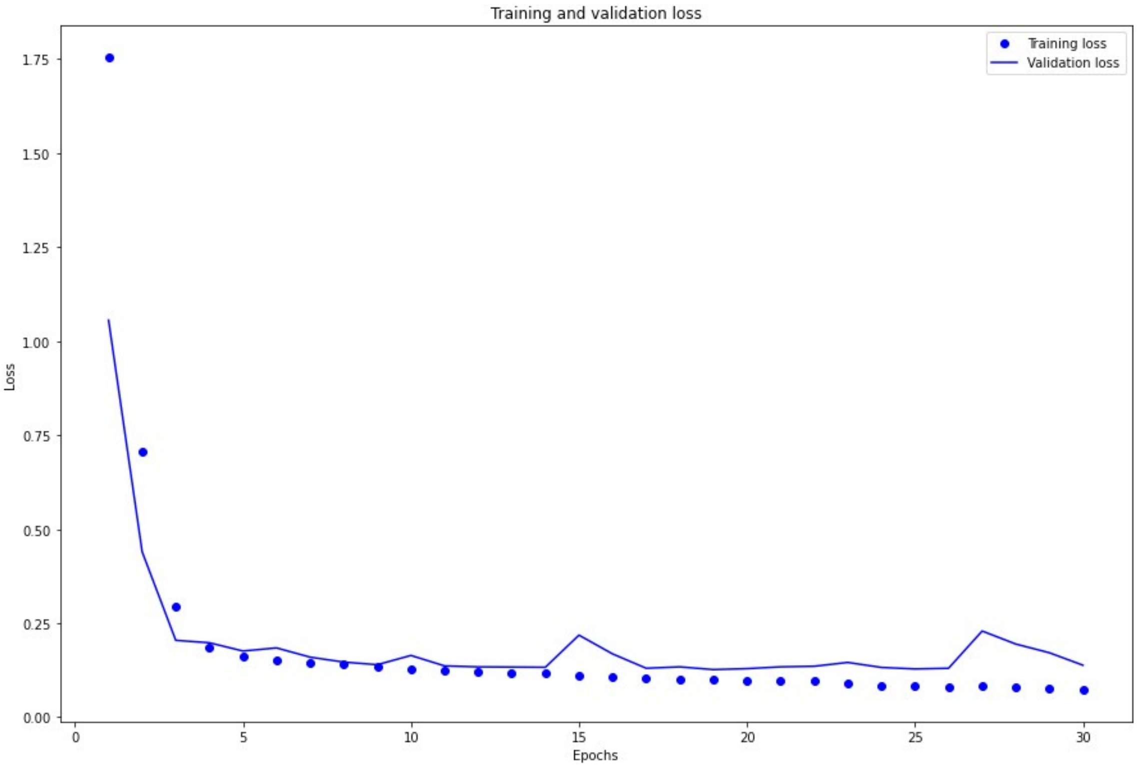


Figure 12: Training and Validation Loss

The plots in Fig 11 and 12 show a 'steep' learning curve for the model where it had reached optimal accuracy after ~20 epochs. An early stop function was also used to stop training at the best cut-off.

The loss plot also indicated that there was little over-fitting happening during the training, this obviously was thanks to the different regularization methods (L2 & drop-out) employed in the model architecture.

Accuracy vs F1-Score

It is important to note that accuracy score could be quite deceiving particularly in unbalanced datasets like the one currently in use. At my first attempt at this problem, I got an accuracy score of 95% while upon inspecting the F-1 Score, the result was 62% - quite a departure from the first impression!

Result comparison and discussion

This project will compare the proposed embedding with standalone BERT and TF-IDF embeddings. Furthermore, it will test the same three combinations on Random Forests and Logistic Regression in a bid to justify the use of Deep Learning Architecture for this binary classification problem.

Table 2: Result Comparision between Embeddings and ML Model combinations

Embedding	Model	F1-Score
Distilled BERT	DNN	**65%**
	LR	61%
	RF	40%
TFIDF	DNN	**65%**
	LR	42%
	RF	64%
BERT-TFIDF	DNN	**74%**
	LR	62.4%
	RF	39%

By observing Table 2, the results are quite clear. First of all Deep Learning Models provide a good advantage using all three combination methods. The proposed 'hybrid' embedding provided ~14% improvement in F1-Score compared to a standard BERT embedding.

However, it is important to note that Random Forest performed almost similar to our Deep learning architecture when using TF-IDF embeddings. Similarly, Logistic Regression performed almost as good as the Deep Learning model when using standard BERT embeddings.

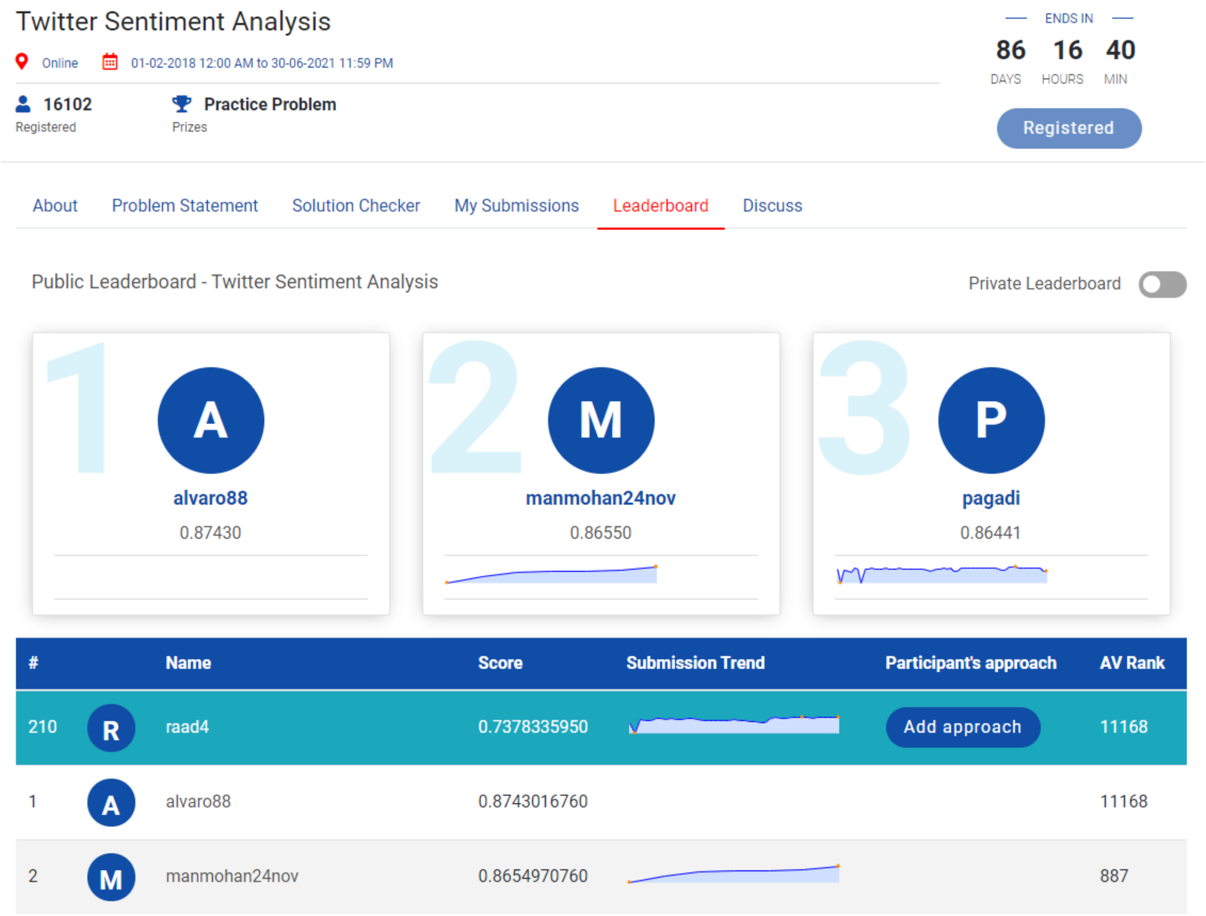
Upon reflecting on the reasons behind this, a possible explanation is that TF-IDF embeddings are generated by probabilistic methods that allign well with the decision trees classifiers that are working under the hood in the Random Forest Classifier Model.

On the other hand, more complex embeddings such as BERT were handled better by Logistic Regression as it seems there is no probabilistic underlying embeddings that a Decision Tree based classifiers could easily unpack.

Eventually, the results achieved, 74% vs 65% F1 scores may very well justify the use of a deep learning architecture for such classification task.

Hackathon results

The proposed architecture was used to participate in the Analytics Vidhya Hackathon, which had more than 16’000 people registered. I was proud to be ranked #210 out of 16102 with my results.



Twitter Sentiment Analysis

Online   01-02-2018 12:00 AM to 30-06-2021 11:59 PM

16102 Registered   Practice Problem Prizes

ENDS IN
86 DAYS 16 HOURS 40 MIN

Registered

About   Problem Statement   Solution Checker   My Submissions   Leaderboard   Discuss

Public Leaderboard - Twitter Sentiment Analysis   Private Leaderboard

1 A alvaro88 0.87430

2 M manmohan24nov 0.86550

3 P pagadi 0.86441

#		Name	Score	Submission Trend	Participant's approach	AV Rank
210	R	raad4	0.7378335950		Add approach	11168
1	A	alvaro88	0.8743016760			11168
2	M	manmohan24nov	0.8654970760			887

The top score was achieved by a fine-tuned ROBERTA embedding using a much larger DNN. Unfortunately training a ROBERTA model on the dataset would take a very large amount of time – compared to the transfer learning approach we used with pre-trained distilled BERT.

7. **Lessons learned**

- Simple ML models can still achieve impressive results in classification problems – so always try them!
- Fine-tuning has infinite possibilities particularly in areas hardly looked at such as the activation threshold.
- Transfer learning can achieve impressive results even if the underlying model did not see the data its being trained on.

8. References

1. https://datahack.analyticsvidhya.com/contest/practice-problem-twitter-sentiment-analysis/
2. https://huggingface.co/transformers/pretrained_models.html
3. https://keras.io/api/layers/core_layers/dense/
4.https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
5. https://towardsdatascience.com/visualizing-keras-models-4d0063c8805e