Detecting hate speech on Twitter using Bidirectional Encoder Representations from Transformers (BERT)

Apolline Foucher
a.foucher@mpp.hertie-school.org

Augustine Malija
a.malija@mpp.hertie-school.org

Vlad Surdea-Hernea
v.surdea-hernea@mpp.hertie-school.org

Abstract

The following project[1] *devises a method of detecting hate speech on Twitter, using deep-learning algorithms. In this regards, the project proposes the Bidirectional Encoder Representations from Transformers (BERT), which is the state-of-the-art method for the majority of NLP-associated tasks. We contrast the results of our proposed BERT architecture with the more traditional Support Vector Machines approach, trying to highlight the improved accuracy offered by deep-learning algorithms. Additionally, we try to optimise the SVM by using complex pre-processing methods, which lead to accuracy levels similar to that of the unoptimised BERT algorithm. This allows us to ultimately see whether BERT's improved accuracy remains significant even after the pre-processing.*

1. Introduction

1.1. Background

Twitter is considered by people to be one of the most popular social-media platforms used currently in the English-speaking world, and it has significantly improved the capacity in which normal people communicate and share opinions, ideas, and sources of information. Nevertheless, given the low levels of active moderation, the very high number of users and the democratic nature of speech transmission, Twitter has come to be exploited by malintended individuals. In this sense, the dissemination of offensive, aggressive and even hateful content has become a staple of Twitter, with policymakers trying to combat this wave of digital aggression.

This untamed distribution of hateful content is indeed one of the biggest problems concerning the content present in the digital world. It has the potential of negatively affecting disenfranchised groups, who already suffer due to the stereotypes in society. However, regardless of the policies that need to be adopted to prevent these development, and regardless of the ethical discussions on the limits of free speech, existing hate speech needs to be detected on Twitter, in order for any measure to be functional. Given the scale of the phenomenon on social media, the best solution remains the automatic detection of hate speech.

Once again, this process is not straightforward .The automatic detection of hate speech is a convoluted and challenging task due to disagreements existing over the different hate speech definitions. Therefore, some content published on Twitter might be hateful to some individuals and not to others, based on the definition of hate speech they employ. Several academic papers and other research projects that rely on traditional machine learning approaches (such as bag of words, and word and character n-grams) have been published in the last decade. Recently however, the proffered methodological setups involved deep-learning algorithms, such as LSTM-based RNNs, or BERT.

1.2. Research Scope

In this project we deploy deep-learning methods for detecting hate-speech on Twitter, focusing on the English language. We make use of existing implemen tation of BERT, an architecture based on the Transformers model. To precisely characterise the efficiency and effectiveness of our implementation of BERT, we use numerous automatic metrics that allow us to see both if the model identifies hate-speech, and if it avoid conflating hate-speech with non-hateful speech. In this sense, our project is interested in both false positives and false negatives, which is an approach rarely seen in the field.

In addition to our implementation of BERT, we also develop a more traditional "shallow" approach, in the form of SVMs. In this regard, we are able to actually see the improvement of using the state-of-the-art in the field of NLP.

One novel approach brought by our project is that in the

[1] GitHub account https://github.com/ApollineFo/NLP---Hate-Speech

case of the SVM, we use complex and comprehensive preprocessing methods that improve as much as possible the prediction power of the model. As such, we are able not only to compare SVMs with BERT, but to compare the best possible SVMs with the latter deep-learning architecture.

1.3. Structure

The remainder of this paper is organized as follows:

- Section 2 briefly reviews the literature in the field, emphasizing the diverse portfolio of methods employed by researchers and scientists for the task of automatic hate-speech detection.
- Section 3 presents our methodological setup, including brief descriptions of both the SVMs and BERT.
- Section 4 and 5 describe the experiments we have conducted and analyses the results obtained through them.
- Section 6 concludes this paper.

Whenever possible, we make use of tables and visual representations for supporting our main arguments.

2. Related Work

2.1. Summary of the literature

The results of similar endeavours, using different "shallow" or deep machine-learning algorithms to identify hate speech in online discourse can be summarised in the table below. As a general rule, results where accuracy is close to 80% have mostly used traditional or "shallow" machine learning methods of hate-speech detection, while the best results are achieved through implementing deep-learning algorithms, mostly complex LSTM architectures or BERT.

Paper	Accuracy
Kwok and Wang (2013)	76.0%
Davidson et al. (2017)	91.0%
Badjatiya et al. (2017)	91.3%
Devlin et al. (2019)	93.2%
Ali et al. (2019)	93.0%
Alshanan and Khalifa (2020)	83.0%
Abro et. al(2020)	79.0%

The first thing we can infer is that generally speaking, state-of-the-art methods tend to have an accuracy of above 91%. The baseline models that we have included in the table reach only around 80% in terms of accuracy. As a result, we can already establish that our implementation of BERT should also achieve an accuracy of at least 91% in order for the algorithm to be valuable.

2.2. Selection of papers

Given the rapid and exponential use of social media, researchers have utilized different machine learning methods and deep-learning architectures for social media challenges. These deep-learning architectures have been used to target those challenges: detection of racism, white supremacism, sexism, moral harassment or incitement to hatred etc. Different periods were also studied to understand if it led to an upsurge in racist tweets as we saw with the resurgence of cyber racism during the COVID-19 pandemic. Therefore, our proposed model of applying BERT in racism and non-racist tweets is based in the latest literature in the field, as represented by papers such as the following:

- Kwok and Wang (2013) focused on anti-black racism. They used a supervised machine learning approach (i.e., Naıve Bayes classifier/"bag-of-words) to label tweets as "racist' and '"nonracists" based on unigrams resulting in an average accuracy of classification of about 76%. Nonetheless, many of these unigrams are not necessarily "racists" based on the context they are being used (i.e., the word "gay" for example).
- Davidson et al. (2017) constructed unigrams, bigrams, trigrams, and a sentiment lexicon to assign sentiment scores to each tweet for three categories: offensive, hate, and neither. They used different models used in previous literature (logistic regression, naive Bayes, decision trees, random forests, and linear SVMs) to test the performance. They finally focused on the logistic regression with L2 regularization for the final model, which had a precision of 91% for offensive while hate categorization performed poorly with 44% hate speech being misclassified. Again, tweets with hateful contents were misclassified because of the context, broad definition, and the use of slurs in everyday communication (i.e., coming from rap songs, for example).
- Badjatiya et al. (2017) applied deep learning architecture (i.e., FastText, CNN, and LSTM) to multiple classifiers with different semantic embeddings (i.e., ngrams, Term Frequency-Inverse Document Frequency, Bag of Words). The best method was LSTM (with Random Embedding and GBDT) which reveals a precision of 91.3%.

- Devlin et al. (2019) improved the fine-tuning based approaches by proposing a new model representation model: BERT (Bidirectional Encoder Representations) from Transformers. They show that the model achieved state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures. In order to prove the effectiveness of their methodology, they test the BERT on eleven natural language processing tasks and prove that BERT outperformed in all of the specific tasks.

- Mozafari et al. (2019) use BERT in order to develop an efficient automatic hate speech detection model. The authors use two different datasets of respectively 16k and 25k tweets provided by Davidson et al. (2017) and Waseem et al. (2018) which are annotated as racism, sexism or neither and train their model such that it detects and classifies automatically the tweets in their respective categories. The results reveal that the BERT model with a CNN-based fine tuning strategy provides the best results as F1- score of 88% and 92% for the two datasets. BERT models outperform traditional methods as it analyses the tweet's information which contains both syntactic and contextual features coming from lower layers to higher layers of BERT.

- Isaksen and Gamback (2020) uses four deep learners based on the BERT, with either general and domain-specific language models. These four deep learners were based on two datasets of 25k and 100k provided by Davidson et al. (2017) and Founta et al. (2018). All four models achieved F1-scores close to or above the current state-of-the-art models. Additionally, attention-based models tested in the study confused profoundly hate speech with offensive and normal language which makes it hard to bring the system into practical use. It is also worth noticing that the pre-trained models outperformed generally in terms of accuracy for the hateful instances.

- Saleh et al. (2020) apply the same deep-learning methods to detect hateful tweet on a collected dataset of 1 M tweets from white supremacist accoynts and hashtags which was reduced to 2k tweets after annotation. They first use domain-specific word embedding learning from the corpus and then classify the the tweets using a Bidirectional LSTM-based deep model and then used a pre-trained BERT which is fine-tuned on the white supremacist dataset using a Neural Network dense layer. BERT outperformed all the distributional-based embeddings (Google Word2Vec, GloVe and WSW2V) with the Bidirectional LSTM-based with an accuracy of 86.4%. This implies that the model provides a better meaningful vector of the words due to its training strategy (deeply bidirectional) and the large corpus it was trained on.

3. Proposed Method

3.1. Baseline method

In terms of a baseline method, we chose to use Support Vector Machines (SVMs). Given a training dataset, each observation labeled as to belong to one of two general categories, an SVM algorithm builds a model that assigns all the observations from a testing dataset to one category or another, making it, therefore, a non-probabilistic binary linear classifier.

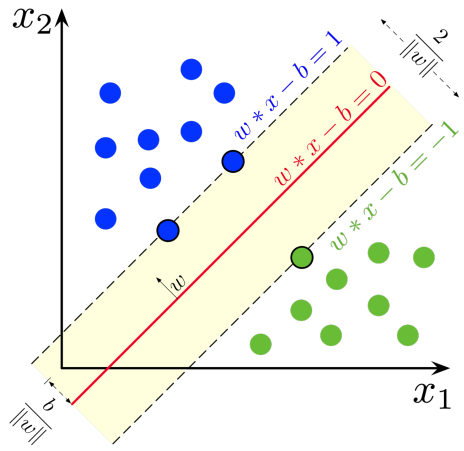

Figure 1. Visual representation of SVMs

The main reason we use this is because of its reputation of having a high prediction power when operating with data from Twitter. The advantages of SVMs include:

- effectiveness in high dimensional spaces, where the number of dimensions is greater than the number of samples;

- memory efficacy, because SVM uses a subset of training points in the decision function.

- versatility, in the sense that different common kernels can be specified in the decision function but also allows for specifying custom kernels

Nevertheless, operating with SVMs has significant disadvantages:

- the risk of over-fitting if the number of features is greater than the number of samples

- they also do not directly provide probability estimates - they have to be calculated using cross validation and probabilities.

3.2. Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a Transformer-based machine learning technique for natural language processing developed by Google in 2019. In turn, Transformers are encoder-decoder architectures developed in 2017 The encoder consists of a set of encoding layers that processes the input iteratively one layer after another and the decoder consists of a set of decoding layers that does the same thing to the output of the encoder. The introduction of Transformers, and specifically BERT, has led to the replacement of previous models such as LSTM-based RNNs.

As opposed to the majority of the architectures that have preceded BERT, this algorithm does not make use of directional models, which read the text input sequentially, either left-to-right or right-to-left. BERTs encoder reads the entire sequence of words at once, making it bi-directional—simultaneously left-to-right and right-to-left.This particular construction feature allows the BERT model to learn the context of a word based on all of its surroundings, regardless of position in relation to the word. The general representation of the Transformer architecture can be seen in the figure below:

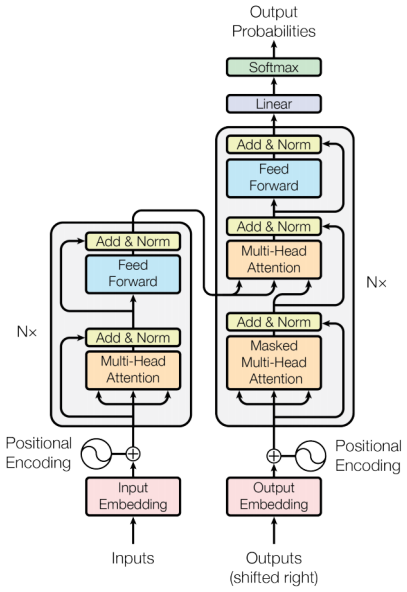

Figure 2. Architecture of Transformers

4. Experiments

Data: Given the scope of our study, we made use of dataset provided by Davidson et al. (2017), which contains tweets that are characterized as being either hateful, offensive or neither of these. As hate speech is a contested term, the authors define it as language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group. This definition avoids a narrow focus on only the most extreme usage of the English language, as accepting the existence of instances of offensive language where people use terms that are highly offensive in a qualitatively different manner.

The dataset provided already annotated data, process which was fulfilled manually by CrowdFlower workers. Three or more people coded each tweet where the intercoder-agreement score provided by CF is 92%. Davidson et al. (2017) decided to assign labels to each tweet based on the majority decision, which given the high intercoder-agreement score was easily achieved. The dataset we used is composed of 24,802 labeled tweets. Given the focus of our study, specifically the automatic detection of hate speech, and not offensive language, we dropped tweets that were labelled as being offensive and not hateful. This action generates a final dataset consisting of 1,430 tweets labeled as hate speech and 4,163 tweets labeled as neither hateful or offensive. A sample of the final dataset can be seen below:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
40	40	3	0	1	2	2	" momma said no pussy cats inside my doghouse "
63	63	3	0	0	3	2	"@Addicted2Guys: -SimplyAddictedToGuys http://...
66	66	3	0	1	2	2	"@AllAboutManFeet: http://t.co/3gzUpfuMev" woo...
67	67	3	0	1	2	2	"@Allyhaaaaa: Lemmie eat a Oreo & do these...

Figure 3. Sample of Davidson et al. (2017) dataset

The large number of unnecessary features, as well as the typical way in which language is used to generate tweets implies a need for a serious and comprehensive process of pre-processing. The pre-processing leads to clean textual data that is easily transformed to numeric arrays. Corollary text pre-processing offers the opportunity for the model's optimality to be tested. Additionally, this will further allow us to improve the accuracy of "shallow" machine-learning algorithms.

In terms of the methodological choice, we have used two different pre-processing techniques without handling stemming/misspelling.This step is crucial to simplify and improve feature extraction. This entailed extracting tweets separately from the whole data frame, ready to be cleaned. To explain the process, we provide an original tweet as follows:

"!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. amp; as a man you should always take the trash out..."

To clean the tweets, we used the following two different pre-processing methods:

First pre-processing method

The first step was removing all unnecessary white spaces in tweet sentences, special characters, substituting multiple spaces with a single systematic space, removing prefixes, removing single characters from the start of tweets, hashtags, punctuations, numbers (e.g.,@, $, %, *).

"rt mayasolovely as woman you shouldn complain about cleaning up your house amp as man you should always take the trash out"

All single characters that would jeopardize the transformation of features to arrays were removed. This also included ones that were found at the beginning of tweets. There was more than one white space between words. They were cleared to remain with a single standard space between words in the tweets. To have our features much clearer, we also removed all prefixes found in the tweets.

Second pre-processing method

In the second pre-processing, we removed all retweets (annotated with an "RT" at the beginning of the tweets), removed all users' name (started with @username), removed all punctuation from the tweet, deleted all numbers, lowered the cases of the tweets, removed all hashtags (i.e., word) and finally deleted all URLs (i.e., https://). After that, all tweets were lowercase, and all stop words were removed.

After all the second pre-processing, we ended up with an updated tweet as follows:

"as a woman you shouldnt complain about cleaning up your house amp as a man you should always take the trash out"

We can now visualize the most common 100 terms based on our pre-processing of the tweets:

Figure 4. The 100 most common terms based on hateful and neutral contents

Software : We made use of both Jupyter Lab and Google Colab for writing the code. In terms of managing the project, we benefited from GitHub. Overleaf was the choice for writing the project in LaTeX. In terms of deep-learning architectures, we used of Hugging Face to develop BERT.

Hardware : In general, training the deep-learning architecture required the usage of significant GPUs, which was mainly done via Google Colab. However, a large number of tests and experiments were also conducted using our personal hardware, listed below:

- **Apolline**: Asus, Intel(R) Core(™) i5-8250U CPU @ 1.60GHz
- **Augustine**: Lenovo T470s, Intel(R) Core(™) i5-7300U CPU @ 2.60GHz 2.71 GHz, RAM 16GB
- **Vlad**: MacBook Pro 2019, 2,3 GHz Quad-Core Intel® Core™ i9 2.30 GHz

Evaluation method: The evaluation of our models, both the SVMs and the BERT, has been achieve through a diverse set of automatic metrics. First and foremost, in both cases we have computed the accuracy of the model, namely the number of tweets in the testing dataset that are correctly labelled by our proposed models. This is the primary metric of evaluation for this type of NLP task, and allowed us to compare our results with the ones already existing in the literature. However, given the additional task we have performed, of designing and optimal "shallow" model in order to see how well they perform contrasted with deep-learning architectures, we also uses additional metrics focused on the SVMs.

Thus, we used two further methods for evaluating our model. These are calculating the ROC AUC score and using the cross-validation evaluation score. These evaluation techniques are helpful for our model since it is a multiclass model. Unlike precision, recall, and F1 scores, they do not form part of the training and testing process. This is because if they were, they would return a perfect evaluation score suggesting that the model, both optimal and sub-optimal, does a good job. In the event of a sub-optimal model, this would lead to model over-fitting. Thus, it was essential to have evaluation techniques that did not form part of the training and testing process.

- With the ROC-AUC score, we are computing the area under the receiver operating characteristics from the prediction scores. This served as our first evaluation technique because it is easy to interpret, with a single score being above 0.5 means that our model is performing suitably well. As it turned out, our model performed well with this technique since the ROC-AUC score was 0.882, meaning that the model is doing a very good job.
- Cross-validation was used to determine the level of model precision. By running cross-validation, we can obtain the score for standard deviation—the lower the standard deviation, the more precise the model. Our model turned out to be exact enough to form a baseline since the standard deviation score, we obtained was 0.004, which is relatively low, suggesting that the model's precision is too high for standards. With such a precise standard deviation, the model can apply to an even larger dataset.

Experimental details for SVMs: We evaluate our model by 5-fold cross validation using our processed tweet. To have our TF-IDF pre-processor working efficiently, we specified the following parameters. We first ignored words with frequencies higher than 90 percent from a cluster of our pre-processed document through a min-df function. In the same respect, we missed records with lower frequencies than 2. Our lower and upper boundaries of the range of n-values for different word grams were unigrams and bigrams. This is because having both of them offered a high enough accuracy, a perfect ROC-AUC score, and a precise standard error from our model's cross-validation test. We also specified the strength of our regularization parameter to be "C" to be 1.

We decided to specify both Gaussian and Linear kernels to take care of irregularities that may have arisen from the dataset as we run our machine learning model. With a Gaussian kernel, we specified a kernel coefficient γ of zero. We set the class weight to "balanced" to automatically adjust weights/probabilities inversely proportional to class frequencies. We did not have to specify further for a linear kernel since it regards the dataset as a regular one with no imbalances.

Since we have more than one class of tweets categorization - hate speech and neutral, we built a pipeline that applies a TF-IDF vectorizer taking into account the parameters we specified. It also uses an OneVsRest classier of SVC that takes into account probabilities/weights. With these in check, we separated our dataset to train and test sets where we have 80 percent of the data as a training set while 20 percent of it as a test set. We then use a Grid Search to do an exhaustive search over parameter values where all processors are used. To be concise, we control the verbosity of messages by only allowing computation time and parameter candidates for each fold to be displayed. The use of Grid search is deemed paramount since it picks the best performing kernel out of the Gaussian and linear ones we specified earlier. The output it produces is one for the best-performing kernel. In this regard, we did not have to worry about being explicit of a single kernel among the two. We then built a classifier by fitting our training sets from which we were able to predict our test set.

Experimental details for BERT To build on SVMs, a further step is taken to develop a deep learning model named Bidirectional Encoder Representations from Transformers (BERT). This model takes care of word contexts and thus avoid generalizations in classifications. Through computing the probability of a word based on a combination of words it has seen previously, BERT learns to predict the probability of a sequence of words. A Hugging face flavor to the model is applied with the use of Pytorch as a language.

In general, a BERT base model comprises a 12 transformer blocks encoder, 12 self-attention heads, and a hidden size of 768. It takes a sequential input of no more than 512 tokens and outputs the sequence representation. One or two segments are attached to the sequence where the first token is comprised with a special classification embedding (CLS) and another one for separating segments (SEP). It is bidirectional in the sense that it considers left and right contexts when learning representations of texts.

In text classification, BERT utilizes the final hidden state *h* of the CLS token to represent the whole sequence as seen in the following equation.

To predict the probability of label *c*, a softmax classifier gets added to the top of BERT. *W* is the matrix for task-specific parameter. All parameters from BERT as well as *W* are jointly fine-tuned by maximizing the log probability of the correct label.

In particular, this served to accelerate the deep learning process. A dataset and a data loader functioning as variables to be used during fine tuning and training phase of the model were then prepared.

This entailed creating a dataset class and a data loader itself. The dataset class defines how text gets pre-processed before being sent to the neural network. The dataset class was defined to accept a data-frame as an input and generate tokenized output from pre-processed tweets that can be used with DistilBERT Model for training. During pre-training, DistilBERT reduces the size of BERT by 40 percent while retaining 99 percent of its capabilities. It also makes inference 60% faster. An $encode_plus$ method of the tokenizer is used to perform tokenization that generates output in the form of ids and attention-ask. It also incorporates a column of the dataset that comprises the three labels assigned to the tweets, namely, in-favor, against and neutral that were coded as 1,2 and 3 respectively. It goes on further to split the dataset into two, one for training comprising 80% and another one for validation that serves for evaluating the model's performance.

Dataloader's definition serves to create training and validation aspects that load data to the neural network in a structured manner. This serves to control the amount of data loaded to the neural network since data cannot be loaded at once from the dataset to the memory.

To enhance the fine-tuning process, a neural network and a loss function optimizer are created. Through a DistillBERTClass, the creation of a neural network is done. It encompasses a DistilBERT Language model with a dropout and a linear layer that will feed the dataset. Final output layers from the feeding will be compared to encoded categories to determine the accuracy of the model's prediction.

A Loss function and an optimizer are further created. Whereas the loss function is used to calculate the difference between the output created by the model and the actual out-

put, an optimizer is used to update weights of the neural network to improve its performance.

An actual fine-tuning happens with the creation of a training function. In this function; a data loader passes data to the model based specified size of batches – a specified batch is 2, model outputs are and actual categories are compared to calculate the losses that are used to optimize the weights of the neural network and get printed out after every 5000 steps.

To test how good the model performs, the validation stage is paramount. At this stage, the unseen data or the testing dataset gets passed to the model to check how well the model performs on unseen data. At this stage, weights of the model are not updated. It's only the final output that is compared to the actual value from which the accuracy of the model is calculated.

Results:

Classifier	Accuracy scores
SVM with no pre-processing	0.664
SVM Pre-processing 1	0.864
SVM Pre-processing 2	0.835
BERT	0.916

The table above shows the results of our multiple models. As such, we describe the overall accuracy of the SVMs (both with and without the pre-processing mechanisms that we have described in the steps above), and the BERT. One general observation is that BERT achieves results that are close to the best results in the literature, while the versions of SVM that have used pre-processing achieve an accuracy above similar "shallow" architectures we have identified.

Comment on your quantitative results First and foremost, we can infer that using deep-learning methods such as BERT outperforms even the best attempts made using "shallow" machine learning algorithms. The differences in accuracy are large, and when translated into policy, would imply significant differences regarding the ease of implementation.

Secondly, the results of the SVMs using complex pre-processing methods are significantly higher than expected. This also implies that the increase in accuracy when contrasting BERT and the pre-processed SVMs is significantly smaller than when comparing BERT and basic SVMs. While the results of the SVMs are nowhere near to the state-of-the-art results of BERT, this is the case for the English language. When trying to automatically predict hateful content in other languages, in the absence of more complex pre-trained Transformer models or in the absence of significant GPU resources, it might be sufficient initially to use basic SVMs, as long as the pre-processing step has been thorough.

Thirdly, the results of the BERT model are rather static and independent of the pre-processing methods used. We have tried multiple experiments using differently pre-processed versions of our original dataset, and the differences in accuracy have not been significant. This is to be expected, but it's important to acknowledge this in practice, as it can prevent people from wasting time on tasks that do not improve the process of automatic detection of hateful speech on social media.

5. Analysis

In order to analyse the efficiency and effectiveness of our approach to automatic hate speech detection on Twitter, we need to properly understand the baseline model and how the results of the baseline model appear based on our expectations. Looking at the figure below, we see that our baseline method correctly predicted about 77 percent of tweets that related to hate speech. On the other hand, 98 percent of tweets that did not relate to hate speech (classified as neutral) were predicted correctly. In this light, about 2 percent of tweets that were related to hate speech were classified as neutral, while about 23 percent of neutral tweets were classified as hate speech. Ultimately, this means that the problem with the SVMs (both with and without pre-processing) was that it had serious trouble correctly identifying and labelling neutral tweets. While this is not absurd, and was to be expected based on the literature review, it's still problematic that the the prediction power of the baseline model lacks balance.

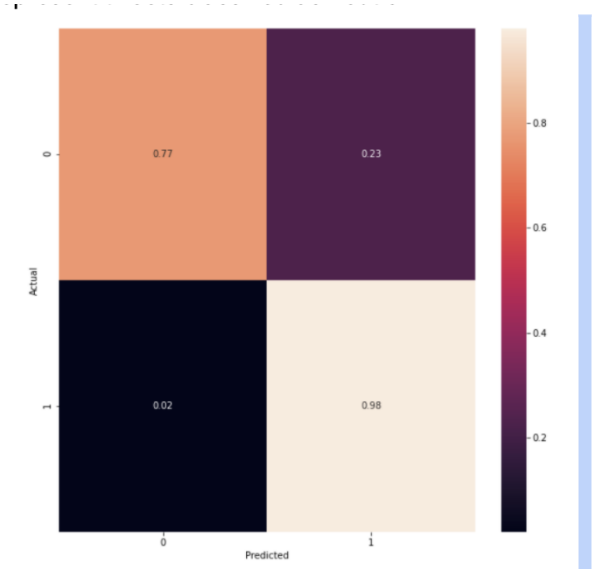

Figure 5. Normalization matrix for the SVM

This is even more of a reason for using BERT. In the case of BERT, having such a high accuracy implies that

we have, regardless of our optimisation strategy, statistically less chances of reaching an unbalanced prediction pattern. Therefore, by using the latest versions of BERT, and building on previous results, we are able to achieve an accuracy of 91.6% distributed uniformly across the categories studied. This was achieved both by balancing the original dataset, and by leaving the dataset unbalanced, proving once again that the BERT architecture is remarkably flexible and not pretentious at all.

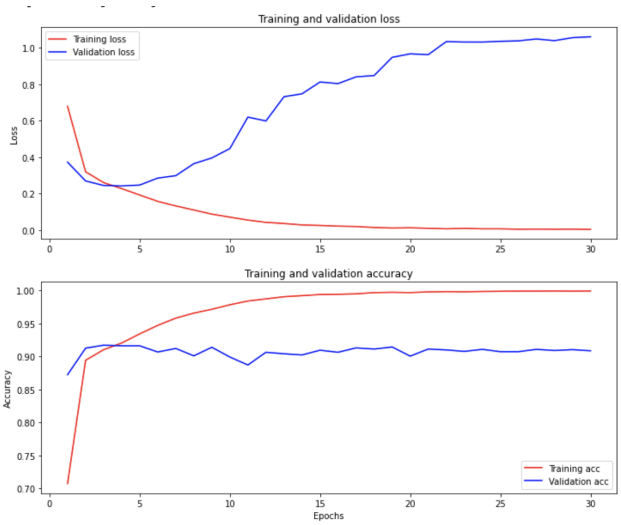

Figure 6. BERT

The figure above show that the accuracy cannot increase constantly, and after a number of epochs it's a matter of very small improvements, if any. This means that we would need, for better performance, a more expansive dataset, or another pre-trained version of the model that is more used with dealing with Twitter. Both extensions are worht discussing in future endeavours.

6. Conclusions

Our project proves that while the automatic detection of hateful content on social media remains a complicated tasks, and current models are imperfect, we can confidently tackle this challenge. By using the most advanced deep-learning methods, we are able to design models that reach accuracies above 90%, in line with the best results present in the literature. Additionally, we show that deep-learning is superior to other approaches that might be considered by researchers, such as SVMs. Nevertheless, pre-processing can increase the accuracy of the SVM algorithm, which could be useful for researchers working in languages other than English.

7. Contributions

All the members of the team have contributed towards the project at all stages. In this sense, we shared responsibility between data analysis tasks and writing tasks as equal as possible.

References

[1] Badjatiya et al. (2017) *D*eep Learning for Hate Speech Detection in Tweets WWW '17 Companion: Proceedings of the 26th International Conference on World Wide Web Companion

[2] Davidson et al. (2017). *A*utomated Hate Speech Detection and the Problem of Offensive Language roceed- ings of ICWSM 2017

[3] Devlin et al. (2019). *P*re-training of Deep Bidi- rectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics.

[4] Isaksen and Gamback (2020). *U*sing Transfer-based Language Models to Detect Hateful and Offensive Language Online Proceedings of the Fourth Workshop on Online Abuse and Harms

[5] Kwok and Wang (2013) . *L*ocate the hate: detecting tweets against blacks. Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

[6] Mozafari et al. (2019). *A* BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. 8th International Conference on Complex Net- works and their Applications

Sahlgren et al. (2018). *L*earning Representations for Detecting Abusive Language Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)

[7] Saleh et al.(2020) White Supremacist Hate Speech using Domain Specific Word Embedding with Deep Learning and BERT arXiv preprint