A Case study: Sentiment evolution throughout a social movement by sentiment classification on Twitter data

NLP Course - Q1 2020/2021 - Anna Mae van de Peut & Arwan Credoz

1. Introduction

The inequality of people of color originates from August 1619 [1]. This is the date that symbolises the slavery's roots. Although slavery has been abolished and people of color have been given their freedom, the so called inequality remained. The Black Lives Matter movement on social media came into existence in 2013 because of a hashtag used on twitter when George Zimmerman was acquitted of murdering an unarmed Black teenager, Trayvon Martin, in the United States [2]. The public opinion of American voters about the Black Lives Matter movement was net negative before 2018, but after that it gained positive net support [3]. There were several major protests over the years, but Black Lives Matter gained further international attention in 2020 following the death of George Floyd [4]. Nate Cohn & Kevin Quealy [3] analysed the American Voters' opinion, however this does not give the total overview of the public opinion.

In recent years, microblogging has become a very popular communication tool. Millions of people use this to share their opinions on different aspects of daily life. Therefore, it is a rich source for sentiment analysis. A. Pak & P. Paroubek [5] created a sentiment classifier using a corpus created from the microblogging platform Twitter and Alec Go et. al [6] introduced a novel approach for automatically classifying the sentiment of Twitter messages. Both studies show promising results for conducting sentiment analysis on Twitter posts. Following these studies, this case study analyses the sentiment of English twitter posts regarding events that triggered the Black Lives Matter movement to give the total public opinion. **The main goal of this case study is giving insight into the question do events in regard to Black Lives Matter have an effect on the sentiment of twitter posts on the Black Lives Matter movement. We will be looking at the opinion polarization pre and post may 25th 2020, which will then allow us to make hypothesis concerning the effect of the event that took place on this date on the attitudes of people regarding the Black Lives Matter movement. We would like to see whether the wrongful murder of George Floyd on that day altered the public opinion and/or reinforced the support for the BLM movement compared to the time period preceding this event.** This case study gives insight on whether we can correlate events with the sentiment of public opinion on Black Lives Matter. This is done by using the labelled data from Sentiment140 [6] to classify the retrieved data using the tool Twint [7]. The retrieved data follows the timeline of the social movement. The data is classified using a supervised machine learning algorithm that uses neural networks and binary logistic regression for 2 classes classifications.

2. Related work

There is a lot of previous work on conducting sentiment analysis on tweets. Such as, Go et. al [6] who created their own dataset, for sentiment analysis, using the Twitter API by querying positive and negative tweets. They scraped tweets using the query ":)" that returns positive tweets and ":(" that returns negative tweets. Then they post-processed their data by removing retweets, tweets with both positive and negative emojis, tweets with ":P" and repeated tweets. After post-processing the data, they took the first 800,000 tweets with positive emojis and the first 800,000 tweets with negative emojis, for a total of 1,600,000 tweets for their training set. For their test set they manually collected tweets using the web application of Twitter by searching specific queries. These queries were arbitrary chosen from different domains. They look at the result set of the query. If the tweet contained sentiment they would mark it as positive or negative and add it to the test set. They explored the usage of unigrams, bigrams, a combination of unigrams and bigrams, and parts of speech as features. They tested Naïve Bayes, MaxEnt and SVM. The results respectively were 81.3%, 80.5% and 82.2% for the unigrams. They did not specify the results for the bigrams, since the use of bigrams as feature did not improve the accuracy. The combination of unigrams and bigrams as feature increased the accuracy for Naïve bayes to 82.7% and for MaxEnt 82.7%. On the contrary, the SVM's accuracy declined to 81.6%. The part of speech (POS) tags as feature did not prove to be useful and even decreased the performance.

Wang et. al [8] compared the performance of sentiment analysis based on Chinese texts and cross media. They used another microblogging platform called Sina Weibo to create their dataset. They extracted messages with images from the top ten hot topics from Sina Weibo. The dataset consists of 2900 positive, 1200 negative and 900 neutral messages of different topics. They labelled the data positive or negative based on the emojis and manually labelled messages without emojis as neutral. They created a Two-class (positive and negative) and Three-class (positive, negative and neutral) classifier. The models they used for their text-based classifiers are Naïve Bayes, SVM and Logistic Regression. They did not specify what kind of features they used for their models. Their accuracies for the Two-Class classifier respectively are 72%, 75% and 76%.

Ruz et. al [9] conducted a similar study on different topic. They used two different Twitter datasets in Spanish. The Chilean earthquake and the 2017 Catalan independence referendum. The former dataset was obtained from the study by Cobo et. al [10]. They formed the latter of 60,000 collected tweets from the 2017 Catalan independence referendum. They pre-processed the data by deleting all URLs, hashtags, targets, punctuations, symbols, numbers, common stop words, repeated characters in a word and all non-Spanish tweets. Furthermore, they converted every tweet to lowercase to make the dataset uniform. Next, they labelled each tweet with a positive or negative sentiment. They used a translated list of English positive and negative opinion words or sentiment words. They used the bag-of-words technique to convert the training tweets into a numeric representation. The models they used are Naïve Bayes, Tree augmented Naïve Bayes (TAN), SVM and random forest (RF). The results for dataset 1 respectively are 74.2%, 72.1%, 81.2% and 72.5%. For dataset 2 the results are respectively 78.1%, 80.8%, 82.9% and 85.8%.

Similarly to the previous work that was mentioned, this study collects tweets from twitter and pre-process them using the same methods. However, the method that will be applied in this study will be using logistic regression. Most of previous conducted studies did not include logistic regression models. However, some studies show that their logistic regression models have better accuracy than the other models they tested [8]. Therefore, it would be interesting to further investigate this method.

3. Data

The data used to **train** the classifier is the dataset from Go et. al [6]. This dataset is commonly used for sentiment analysis. It is suitable for this study since it is data from the same microblogging platform.

The **test** data is retrieved using the Twint tool. This is an advanced Twitter scraping tool that allows for scraping Tweets from Twitter profiles without using Twitter's API. This is advantageous because the Twitter API has a fetch limitation to last 3200 tweets and rate limitations. Additionally, this tool can be used without signing up for Twitter.

The tweets can be scraped in three ways. That is searching for tweets, scraping tweets from the user's timeline or scraping the favourite tweets of a user. This study used the search tool to scrape tweets that have a specific tag. These tags are words or compounds that are commonly used when people are tweeting about the Black Lives Matter movement and rarely used in other situations. The tags that are used for the scraping queries are "Black Lives Matter", "George Floyd", "Breonna Taylor" and "BLM".

N. Cohn & K. Quealy [2] showed the net support of the Black Lives Matter movement from 2017 until 2020 and that there is a significant peak after the 25 May 2020. Initially, the intention was to follow the 2017-2020 timeline for the sentiment analysis. However, this would lead to a large and sparse dataset. Therefore, for this study the sentiment analysis is focused on the peak that follows after the 25th of May 2020. The time of the retrieved data spans from 29-12-2019 until 20-10-2020. Within this timespan the 25th of May is exactly in the middle.

The parameter for the limit of tweets scraped is set to 3500 to control the sparsity of the dataset. This will limit the number of tweets that can be scrapped. Therefore, the tool will continue to scrape until either all the tweets of the query have been scraped or until the limit of 3500 tweets. Twint scrapes chronologically, so when scraping with a limit of 3500 tweets per day, the tweets scraped are distributed over the whole day. Unfortunately, due to a bug in the Twint tool, the python script failed to generate a csv file for the "BLM" tag and therefore results using this tag won't be shown.

Finally, the csv output file was customized such that only the relevant information is saved. The data saved in the csv file is the date, timestamp, time zone, tweet and the hashtags. In the following table you can see a few example tweets that have been scraped using the Twint tool. The output features of the Twint tool have been customised, such that it only outputs the date, time, time zone, tweet and the hashtags. This is to reduce the sparsity of the dataset.

Date	Time	Timezone	Tweet	Hashtags
6/18/2020	23:58:52	200	I hope all my family in buffalo everyone has been to city hall atleast once since the death of George Floyd. We need y'all there at these rallies because we need UNITY!!	[]
6/18/2020	23:58:47	200	Hypocrisy is when we the Africans march because of George Floyd, and get back at home we teach our children to do worse. We need to change OURSELVES FIRST to HEAL THE WORLD ðŸŒ #BlackLivesMatter #AllLivesMatter #Love #Peace https://t.co/go5u6RFKG5	[‘blacklivesmatter’ , ‘alllivesmatter’ , ‘love’, ‘peace’]

4. Method

INTRO

We have been taught multiple techniques for sentiment classification. For example, we had already used a **naive Bayes** algorithm for a similar task as a homework. This bayesian classifier makes a decision based conditional probability, and we had reached an accuracy of 76% relatively easily. Although this technique is sometimes applied in classification tasks, we decided to tackle this project using another technique, involving supervised machine learning, basing our knowledge from chapters 5, 6 and 7 from the book by D. Jurafsky and J.H. Martin.

For our task, we used **logistic regression**. Being limited by the labeled data we could find, we will only do **binary logistic regression** (for two classes, positive and negative sentiments). Like naive Bayes, logistic regression is a **probabilistic classifier**. One of the difference here is that logistic regression is a **discriminative model**: it will model the posterior probability of P(y|x) by learning the input to output mapping by minimizing the error.

We need labeled data to be able to train a model: a so called '**neuron**' will receive some features as input, multiply the input with some weights, pass the value through an activation function and output the final value. This value will be compared to the real label (after being passed through a loss function) and will be used to tweak the weights so that after multiple iterations of this forward and backward propagation, the output finally matches the expected value. We will use the open-source **Keras** library as a support for TensorFlow.

However, why use only a single neuron, when we could stack them and have multiple layers of them for more elaborate feature analysis and better accuracy ? We will hence attempt to build a **neural network** and test different model architecture and hyper-parameters values to achieve high accuracy.

PREPROCESSING STEP

The first step of this method is to **prepare the data** needed for the neuron. First of all, we load the data into *pandas* data-frames and make sure it is correctly stored, labeled etc…

	Label	Index	Date	Tag	User	Text
0	1	1972310480	Sat May 30 08:59:15 PDT 2009	NO_QUERY	killakami	Show was sick last night Out with mom, work l...
1	1	1981902104	Sun May 31 10:07:26 PDT 2009	NO_QUERY	KelcieJForrer	W my eeshy poo
2	1	1968724351	Fri May 29 22:04:02 PDT 2009	NO_QUERY	BackstreetBB21	@cpark9 He was wonderful. Got to see him twice...
3	1	1881868843	Fri May 22 05:44:45 PDT 2009	NO_QUERY	jimmychung	@basantam Good morning! I will be more active ...
4	0	2211211299	Wed Jun 17 12:34:32 PDT 2009	NO_QUERY	Hipchick999	I've just been accused of Image Theft

We separate the data we have into a **training list** and a **testing list**. We will leave 10 000 tweets for the testing and leave the rest for the training.

The data is, as we can see in the 'Text' column, in the form of raw sentences. What we mean by raw is that the sentences could literally be made of any chain of characters since twitter lets the user write anything they want. This means for example that there could be **non-ASCII characters** such as ".网络" which is the equivalent of ".com" in china. Talking about **URLs**, they are also something we need to be removing. Bellow is a list of all the operations we do for the **normalization** of the data (note that their order is relevant for a correct cleaning).

	Action	Input	Output
1	**Remove or replace non-ASCII characters**	- “网络” - “éphémère”	- “” - “ephemere”
2	**Remove URLs**	- “check this link: http://www.example.com/index.html”	- “check this link:”
3	**Replace contractions**	- “isn’t” - “I’m”	- “is not” - “I am”
4	**Remove punctuation**	- “@barbie - love you ! :D”	- “barbie love you”
5	**Replace numbers by text**	- “10 gallons of milk please!”	- “ten gallons of milk please!”
6	**Remove single characters**	- “I want a banana”	- “want banana”
7	**Remove stop-words**	- “He attacked Godzilla with an exploding kitten”	“attacked Godzilla exploding kitten”
8	**Stemming**	- “Quickly ! The exploding kittens !”	“Quick ! The explode kitten !”
9	**Lowercasing**	“FoGive mE fAtheR”	“forgive me father”

We weren’t satisfied by the stemming of LancasterStemmer from NLTK so we decided not to include this step in our normalization. Here is an example of cleaning on what could be a legitimate tweet:

“@brian http://bbc.in/1BO3eWQ Check it quickly héhé ! 网络 This isn't a test. I got 10 good answers! :) #LOL”

becomes:

“brian check quickly hehe this test got ten good answers lol”

We are satisfied with this result.

EMBEDDING MATRIX

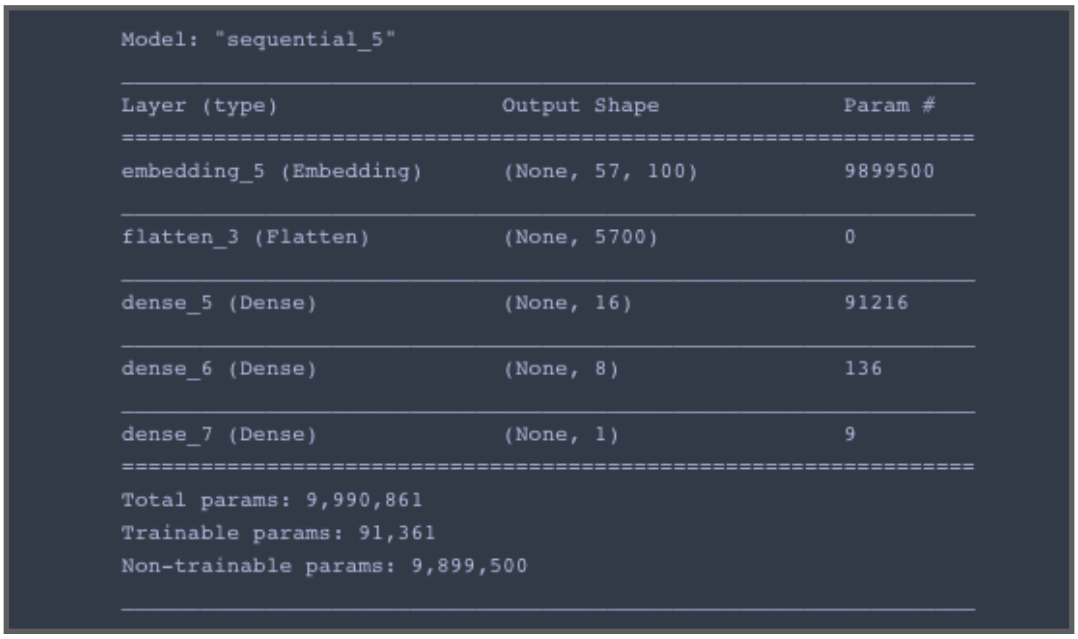
Using the tokenizer’s *fit_on_text()* function we can create a **vocabulary index** based on word frequency. In other word, every different token will be assigned a value (“was” = 1, “sick” = 2, etc…). So we mix together the training and testing sentences in a temporary variable and create this large dictionary based on it. This is the first step in order to create an **embedding matrix**, which will be used in an **embedding layer**. Embeddings are words represented as vectors. By mapping words to vectors of real numbers, we can represent words in a multidimensional space and for example look at their location to differentiate the meaning of sentences by cosine similarity.

Next step is *0-padding*: every sentence has a different size, so we will measure and locate the largest one and use it as reference. After that, we can add ‘0’s to the other list of tokens so that they all measure the same length. Once this is done, we can use texts_to_sequences() from Keras to create a list of integers from the list of tokens (similar to the *fit_on_text()* function).

Next step is loading GloVe’s word embeddings of Twitter tweets. These are pre-trained **word vectors** from 2 billion tweets (27 billion tokens). We then create a matrix where each row will correspond to the index of a word in our corpus. The matrix will have 100 columns, which represents the dimension of the vectors, where each column will contain the GloVe word embeddings for the words in our corpus.

NEURAL NETWORK

Now that the data is ready we can build our network. We will use Keras' ***Sequential*** class to create a network with a plain **stack of layers**. The first layer we add is the **embedding layer** where we reference our **embedding matrix** so that we don't have to train it from scratch. Followed by a **flattening layer** to flatten the output of the embedding layer into a single vector. Then a **dense layer**. This layer is in some way the neurons, since it does the following operation: *output = activation(dot(input, kernel) + bias)* where we chose **sigmoid as activation function** (see appendix) and the kernel are the weights. We define '1' to be the output shape since we only need a probability (closer to 0 will mean negative and closer to 1 positive). Finally, we define the loss function to be the **binary cross-entropy function** (see appendix) and add the 'Adam' optimizer to take care of some hyper parameters such as the learning rate and the momentum. This optimizer is often used because of the good results it yields instead of defining manually a learning rate scheduler or which optimization algorithm to use (SGD, batch gradient descent, etc…). We started testing with this simple architecture first.



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 57, 100)           9899500
_________________________________________________________________
flatten_3 (Flatten)          (None, 5700)              0
_________________________________________________________________
dense_5 (Dense)              (None, 16)                91216
_________________________________________________________________
dense_6 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 9
=================================================================
Total params: 9,990,861
Trainable params: 91,361
Non-trainable params: 9,899,500
_________________________________________________________________
```

The snapshot above is an example of architecture we tested. As you can see, after flattening, the values go through a layer of 16 neurons with '**ReLu**' activation function (see Appendix), then through a layer with 16 neurons, one with 8, and finally our single neuron output all with the sigmoid activation function. The '57' represents the longest sequence of text we have.

The following is an explanation on how to compute the amount of trainable parameters:

```
vocab_size = 98995
matrix_dim = 100
nb_neurons = 16, 8, 1
max_sequence = 57

trainable params:

1st layer  = max_sequence * matrix_dim * nb_neurons + nb_neurons
           = 57 * 100 * 16 + 16
           = 91216

2nd layer  = 8 * 16 + 8
```

```
           = 136

3rd layer  = 1 * 8 + 1
           = 9

Total params     = 91216 + 136 + 9
                 = 91361
```

The difference in total parameters and trainable parameters comes from the fact that we aren't training our embedding layer since we incorporated the weights from GloVe. Thanks to this, we save a precious amount of time every time we do fit a model.

Later on we will try new architectures, combining different type of layers and quantity of neurons. We tried adding *Dropout*, *GRU* which are gated recurrent units or *Conv1D* which is a convolutional layer which could let use specify the kernel size for the convolution, which in our case would actually represent the *n-gram* ([*n,1]* kernel —> *n-gram features*). We ended up finding that the *LSTM* (short term long memory) gave relatively good results. We didn't have enough time to test extensively all the possibilities, but with a bit more testing it is very likely that we could achieve better accuracy. However, the time needed to train the model greatly varies depending on its architecture. For example, a model made of 10 neurons of a dense layer will train much faster than one with a layer of 10 LSTM. Furthermore, we decided to use 10% of the data for the tests in order to be able to try multiple architectures as this is quite time consuming. We assumed that a model working well with 10% of the data could only get better with 100%.

5. Results

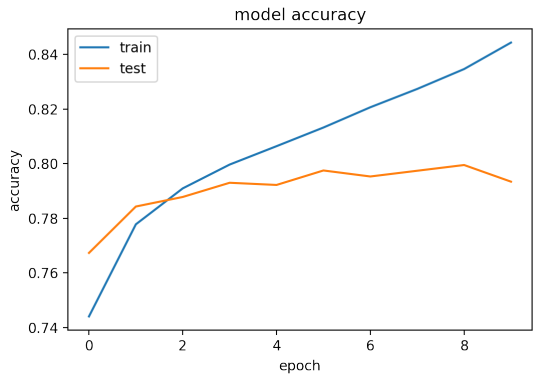
We can share 2 types of results, the ones regarding the training part of the NN, and the ones regarding the project itself (our classification of our retrieved tweets).

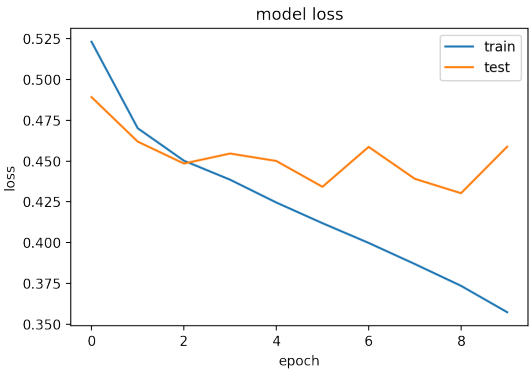
REGARDING THE NN TRAINING

As mentioned before, we have tested different network architectures and have tried to keep track of the one giving the best accuracy. But it is not as simple as sometimes the accuracy on the training data and test data were not always matching. This usually is a sign of overfitting: the model becomes strong at predicting the data it is trained with, but looses track of the general features. This is represented by a training accuracy that keeps improving but a decrease in testing accuracy. Overall, we would keep the weights for which the testing accuracy was the highest. We also had some scenarios where the accuracy was rising but the loss values were decreasing, but we could't understand what caused it.

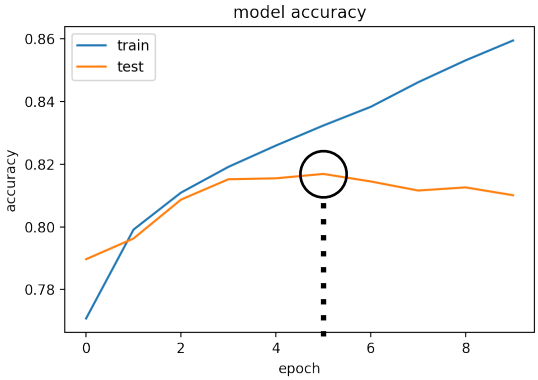
Some of the many optimizations we could add to enhance the results are weights regularization, as well as trying different network architectures.

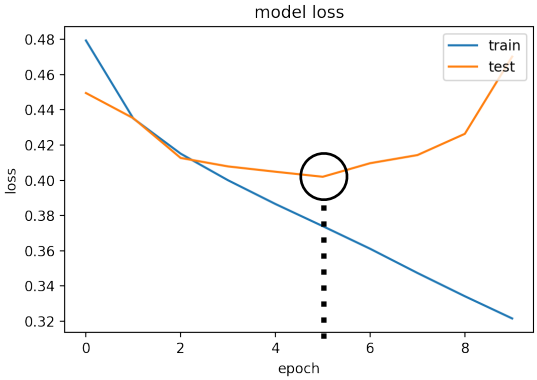
The following results are obtained using only dense layers. As we can see, the training accuracy is greatly increasing and seems like it would continue to do so for many more epochs. However, the testing accuracy is stagnating.





As mentioned before though, we achieve better testing accuracy by using a layer of the recurrent neural network LSTM, while still experiencing a rise in test loss after the 5th epoch. Here is the result, where we ended up keeping the weights from epoch 5 since this is when we reached the highest testing accuracy at approximately 82%:



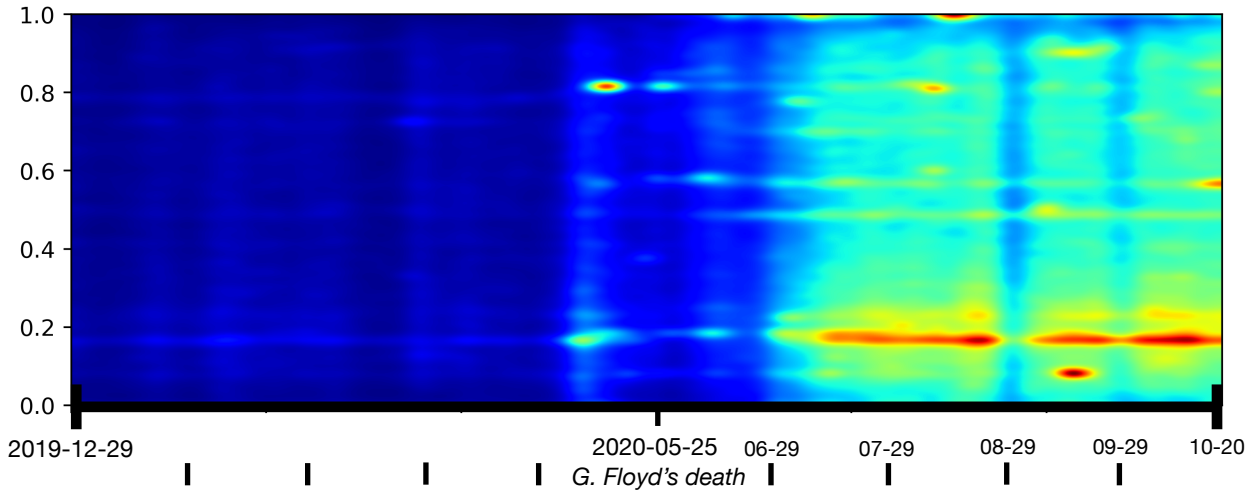


Here are a few examples of made-up sentences and their predictions just to actually see concrete examples and how the classifier reacts:

	Sentence	Prediction	Ground truth
Straightforward sentences	"I am in such a good mood today! Sun is shining and i'm ready for work"	0.894	1
	"I hate Mondays, and on top of that it's raining... Nothing could go worse"	0.239	0
Ambiguous sentences	"Sarah loves her little brother but hates her big sister"	0.030	0
	"Don't let hatred take over. You are not a killer, come back to the right side of the force!"	0.067	No idea
Wrongly classified sentences	"After many days of hard work, the innovation team finally presented their new creation"	0.255	0.5
	"Keep your friends close but your enemies closer"	0.970	0.5
	"Here we go again, four more years with a president unfit to run a country"	0.515	0

REGARDING OUR CLASSIFICATION

Below is a heat-map of the predicted sentiments of approximately 3500 tweets per day regarding BLM, ranging from 29-12-2019 to 2020-10-20, so about 1 million tweets in the span of 10 months.



We chose to illustrate our results in the form of a heat map as it adds a third dimension to the plot: the horizontal axis shows the timeline, the vertical axis shows the sentiment (0:negative, 1:positive) and finally the color map shows the intensity of tweets around a certain period of time and around a certain sentiment.

We can clearly see the beginning of mass tweeting about BLM starting from George Floyd's death. The sentiments from these tweets starts range from positive to negative but we can see a large heated area at around 0.17. This is a proof that most of the reactions were negative as 0.17 is quite a low value. Anything under 0.5 would be considered negative but with some doubt. Since we get sentiments so low, we can assert with certainty that the tweets were negative. This trend continues for the following months. These values coincide with the multiple protests happening around the word to challenge the governments about police brutality against black people, notably the *Strike for Black Lives* event.

6. Discussion

DATA

Regarding the data scraping, we feel that we were limited by multiple factors. First of all, a difficult aspect of our project is that we are looking at a very big timeframe. We reduced it to one year and still had to limit ourselves to 3500 tweets a day. Furthermore, we could have chosen different search tags such as “police brutality” or “cops violence” etc… to get a broader view of the sentiments.

On an additional note, since the BLM movement is a worldwide phenomena, it would have been interesting to gather tweets from multiple languages. This could for example let us observe how different countries reacted.

CLASSIFIER

We were confronted with the time consuming work that is finding the best model architecture for our classifier. The field of neural networks is so vast and varies a lot from case to case. This meant that we had to come up with our own architecture, parameters etc… and the only way to do so is by testing. However doing this many tests takes a lot of time, both for the programmer and for the computer. A solution would be for example lowering the amount of data used to train, but this can only work to some extent because too few data will inevitably result in overfitting. Also, working on a 2012 MacBook laptop isn’t the best idea, so using a computer with high CPU would definitely enhance the experience.

We reached an accuracy of 82%, which we are happy about regarding the time we had on our hands. Yet it might not be high enough to form any conclusion on the data we classified.

RESULTS

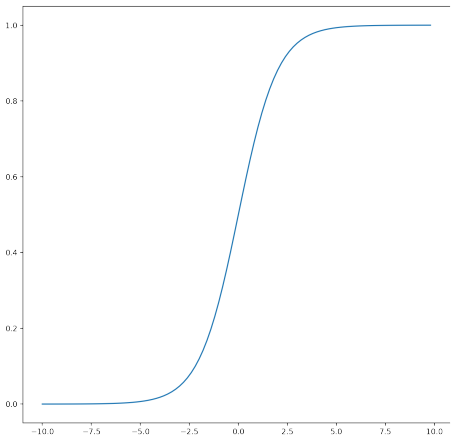
As stated in the introduction, our goal was to look at the opinion polarization over a timeframe to observe the impact on people’s sentiments regarding BLM preceding and following events that took place during the timeframe. An intuitive hypothesis we made was that the tweets would greatly shift in negative sentiments following an event such as George Floyd’s death and as we can see in our heat map of the predictions, the average sentiment seems to tend towards a value of 0.17 so negative sentiment with no hesitation.
The continuous hight concentration of negative tweets starting must be from the multiple protests that emerged during this period and could also be an effect of echo chambers. As stated in Wikipedia, in news media an echo chamber “*is a metaphorical description of a situation in which beliefs are amplified or reinforced by communication and repetition inside a closed system and insulates them from rebuttal.*”.

Overall, the results match with the expectations. We were also looking to see if the sentiments would shift to become more positive with time, or perhaps grow neutral after some point, but apparently it has been staying mostly negative up to today. It would be interesting to reiterate this project in a few years perhaps, taking time to collect more data and look at a bigger timeframe to see if any of these changes appear.

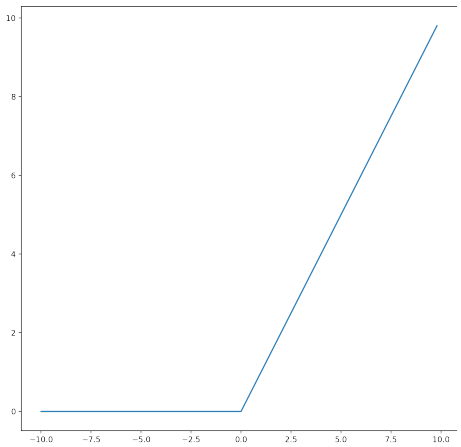
References

[1]	H. Editors, "Black History Milestones: Timeline," History, 10 July 2020. [Online]. Available: https://www.history.com/topics/black-history/black-history-milestones. [Accessed 23 October 2020].
[2]	C. Banks, "Disciplining Black activism: post-racial rhetoric, public memory and decorum in new media framing of the Black Lives Matter movement," *Continuum,* pp. 709-720, 2018.
[3]	N. Cohn and K. Quealy, "How Public Opinion Has Moved on Black Lives Matter," *The New York Times,* 10 June 2020.
[4]	B. P. Dreyer, M. Trent, A. T. Anderson, G. L. Askew, R. Boyd, T. R. Coker, T. Coyne-Beasley, E. Fuentes-Afflick, T. Johnson, F. Mendoza, D. Montoya-Williams, S. O. Oyeku, P. Poitevien, A. A. I. Spinks-Franklin, O. W. Thomas, L. Walker-Harding, E. Willis, J. L. Wright, S. Berman, J. Berkelhamer, R. R. Jenkins, C. Kraft, J. Palfrey, J. M. Perrin and F. Stein, "The Death of George Floyd: Bending the Arc of History Toward Justice for Generations of Children," *Pediatrics,* vol. 146, no. 3, 2020.
[5]	A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," 2010.
[6]	A. Go, R. Bhayani and L. Huang, "Twitter Sentiment Classification using Distant Supervision," *CS224N project report Stanford,* 2009.
[7]	O. team, "TWINT - Twitter Intelligence Tool," OSINT, 17 October 2020. [Online]. Available: https://github.com/twintproject/twint. [Accessed 19 October 2020].
[8]	M. Wang, D. Cao, L. Li, S. Li and R. Ji, "Microblog Sentiment Analysis Based on Cross-media Bag-of-words Model," in *ICIMCS '14: Proceedings of International Conference on Internet Multimedia Computing and Service*, Xiamen, 2014.
[9]	G. A. Ruz, P. A. Henríquez and A. Mascareño, "Sentiment analysis of Twitter data during critical events through Bayesian networks classifiers," *Future Generation Computer Systems,* vol. 106, pp. 92-104, 2020.
[10]	A. a. P. D. a. N. J. Cobo, "Identifying Relevant Messages in a Twitter-Based Citizen Channel for Natural Disaster Situations," in *Proceedings of the 24th International Conference on World Wide Web*, Florence, 2015.

Appendix



Sigmoid function: handy activation function as it maps any values to one ranging from 0 to 1. Ideal for binary classification



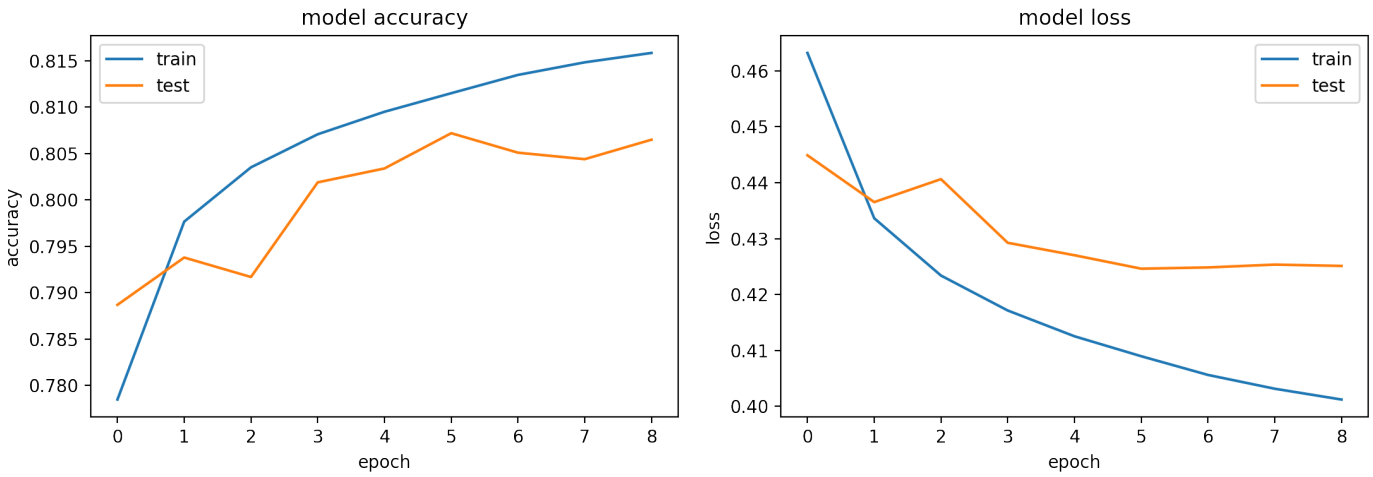
ReLu function: returns 0 if input < 0, returns the input otherwise. This function is helpful to “turn off” some neurons that have negative values making the computations more efficient.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 41, 100)           21720100
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               117248
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 21,837,477
Trainable params: 117,377
Non-trainable params: 21,720,100
_________________________________________________________________
```

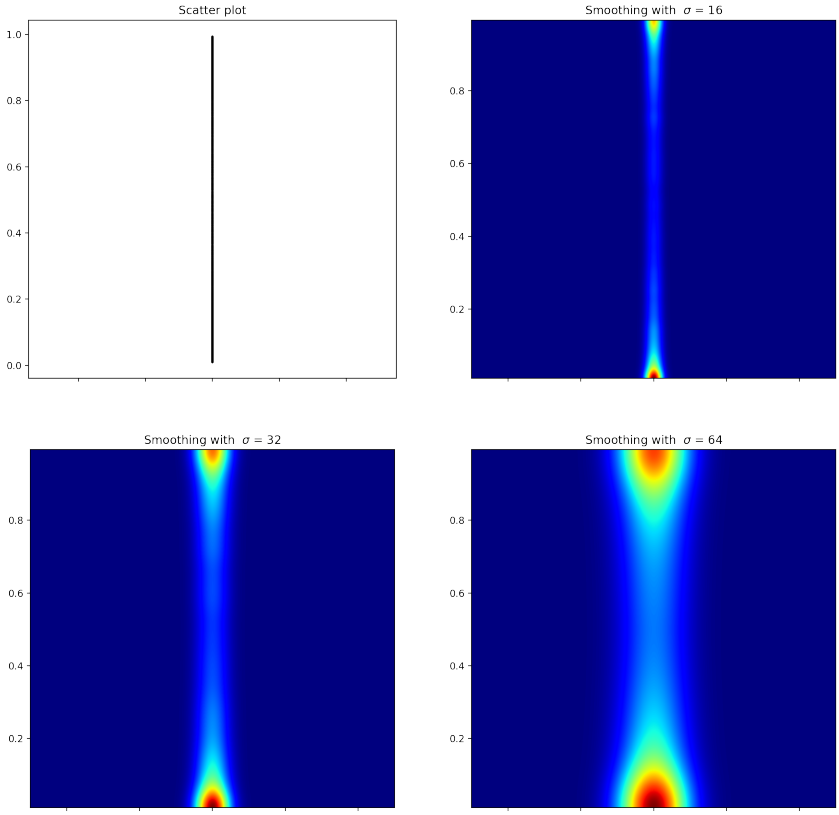
Final NN architecture, using an LSTM layer.

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_18 (Embedding)     (None, 400, 100)          26637000
_________________________________________________________________
conv1d_13 (Conv1D)           (None, 400, 32)           3232
_________________________________________________________________
conv1d_14 (Conv1D)           (None, 399, 32)           2080
_________________________________________________________________
conv1d_15 (Conv1D)           (None, 397, 32)           3104
_________________________________________________________________
max_pooling1d_12 (MaxPooling (None, 198, 32)           0
_________________________________________________________________
flatten_16 (Flatten)         (None, 6336)              0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                63370
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 11
=================================================================
Total params: 26,708,797
Trainable params: 71,797
Non-trainable params: 26,637,000
_________________________________________________________________
```

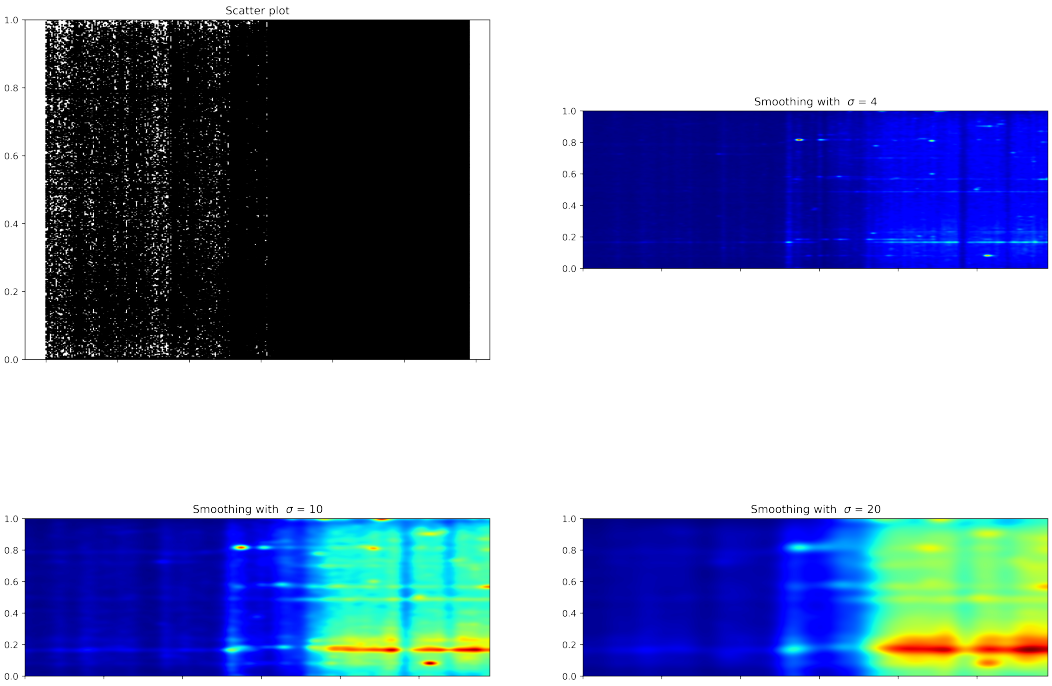
Another model we tested with, it is made of 3 convolutional layers which each have kernels of size 1, 2 and 3. In other words it trains the model with 1-gram, 2-grams and 3-grams. It has achieved better accuracy than doing so separately.



Here are the accuracy and loss value of the above network



Heat-maps with gaussian filter with different smoothing coefficient of 10000 classifications on the testing data (made of positive and negative tweets) flattened in a vertical line. Values ranging from 0 to 1, from negative to positive respectively.



Heat-maps of the result of ~1.5million tweets about BLM classification, distributed from 2019-12-29 to 2020-10-20