The Sentiment Analysis of Depression using Tweets – Classification using Word2Vec Embedding with Neural Networks

Ahmad Luqman Al Hakim bin Shamsudin (966034)

ahmadluqmanalhakim.shamsudin@studenti.unimi.it

Master of Science in Data Science and Economics, University of Milan, Italy.

Contents

Section 1: Introduction

When there is a fact, there is an opinion. And an opinion comes from how we feel and think, which in turn shapes our perception of the world and our life satisfaction. With the growing technology of micro-blogging platforms such as Twitter, people share their opinions more freely, which creates an abundance of data ready to be analysed. Coupled with speedy advancements in the field of artificial intelligence (AI), it opens the gate to a plethora of new research fields. In recent years, researchers such as Aramaki et al., 2011 and Hawn, 2009 have been using social media data to study health at individual and population levels. In this project, population mental health using natural language processing (NLP) is studied.

Mental health as a pedestrian topic of conversation has not had the best record in the past. Emanated by misinformation, it was and still is a heavily stigmatised topic (Baumann, 2007). This causes people to not seek help when needed for the fear of being discriminated against. An instance of where this turns into a public health problem is when the fear of discrimination leads to a significant rate of underdiagnosis. Despite the availability of appropriate care, underdiagnosis of mental health conditions leads to unfavourable outcomes such as higher suicide rates, and a higher rate of antisocial behaviour amongst others. Therefore, it is in the public health interest for researchers to take advantage of NLP

technology and data availability from Twitter, which has great potential in improving the detection of mental health conditions. Evidence in this area has been growing rapidly due to contributions such as those of Coppersmith et al., 2014 and De Choudhury et al., 2014.

In this project, I aim to identify machine learning methods that are appropriate to analyse tweets. Furthermore, I use 3 models to classify tweets that belong to the 'depressed' category. This project will serve as a foundational work for my thesis project, where I will analyse my own tweets dataset and extend it into a causal analysis in the microeconomics framework. This is explained further at the end of this paper. The analysis stage of this project is divided into 2 main substages: the benchmarking analysis and the main analysis. In the benchmarking analysis, I run the dataset through an off-the-shelf sentiment analysis tool 'TextBlob'. In the second stage, I perform the sentiment analysis using 2 different models: vanilla Word2Vec with RNN and Word2Vec with RNN and LSTM cell.

In section 2, I delineate the methodology used for this project. Section 2(a) provides an overall impression of the acquired dataset with some summary statistics and data visualisation. Section 2(b) provides the data pre-processing techniques employed to clean the dataset. Section 2(c) provides a brief explanation of how the models are used in this project. Section 3 provides the results of the different models used in this project and they are compared. The discussion of the results is provided in section 4 and thus concludes the project.

Section 2: Methodology

This section explains all the steps performed in this project starting from dataset acquisition, data pre-processing and analysis.

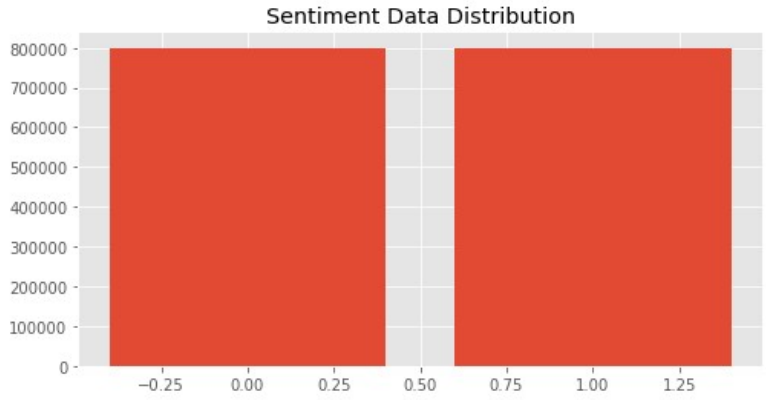
Section 2(a): Obtaining the dataset

The dataset used in this project is called the Sentiment140 dataset. It is constructed by Go, Bhayani and Huang, 2009 who were graduate students in computer science at Stanford University. The dataset contains 1.6 million tweets which are scraped from Twitter using Twitter API access by the authors. In the original file downloaded, it has 6 columns which are sentiment classification, tweet id, tweet date, the query, username, and tweet text.

However, for my project, I only keep sentiment, tweet date and tweet text columns. This section provides a general overview of the dataset. The image below shows a screenshot of the first few rows of the dataset.

	sentiment	id	date	query	user_id	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

The encoding of the sentiment classification is mapped onto 0 and 1, where 0 represents non-depressed tweets and 1 represents depressed tweets. Shown below is the distribution of the dataset with respect to the sentiment. It is a perfectly balanced dataset.



Lastly, I inspect the text column to see some samples, as shown below:

"Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D"

"is upset that he can't update his Facebook by texting it... and might cry as a result. School today also. Blah!"

"I dived many times for the ball. Managed to save 50%. The rest go out of bounds"

Section 2(b): Dataset pre-processing

The data pre-processing procedure is done using the 'nltk' library combined with regular expressions processing, in steps outlined as follows:

1. Lowercasing the texts
2. Remove URL's
3. Replace emojis with text equivalents
4. Remove non-alphabet characters
5. Remove repeating characters
6. Remove stop words
7. Stemming the texts
8. Lemmatising the texts

Performing all these procedures outputs a column of processed texts which are saved in a column of the dataset, as shown below. This pre-processed dataset is also saved as a local file so the processes will not have to be repeated.

	sentiment	date	text	processed_text
0	0	Mon Apr 06 22:19:45 PDT 2009	@switchfoot http://twitpic.com/2y1zl - Awww, t...	aww bummer shoulda got david carr third day EM...
1	0	Mon Apr 06 22:19:49 PDT 2009	is upset that he can't update his Facebook by ...	upset update facebook texting might cry result...
2	0	Mon Apr 06 22:19:53 PDT 2009	@Kenichan I dived many times for the ball. Man...	dived many time ball managed save 50 rest go b...
3	0	Mon Apr 06 22:19:57 PDT 2009	my whole body feels itchy and like its on fire	whole body feel itchy like fire
4	0	Mon Apr 06 22:19:57 PDT 2009	@nationwideclass no, it's not behaving at all....	behaving mad see

Section 2(c): The sentiment analysis (TextBlob)

Up until now, we have the sentiment labels that come with the dataset. However, the methodology used to obtain the original labels are based on emojis. Where tweets containing positive emojis are labelled as non-depressed and vice versa. For comparability with more advanced techniques that are used in the later stage of this project, I perform a baseline sentiment analysis on this dataset using TextBlob. TextBlob predefines polarity and weight carried by words. Parsing the tweets through this tool returns the polarity for every tweet. The polarity ranges from -1 to 1. The negative values are associated with negative sentiment and vice versa. Therefore, tweets with negative polarity are mapped into 1, which means they are most likely to be depressed tweets. While tweets with positive polarity are mapped as 0, which means they are most likely to be non-depressed tweets. The accuracy of this method is computed by comparing it to the original sentiment labels.

Shown below is the resulting dataset:

	sentiment	date	text	processed_text	textblob_polarity	textblob_sentiment
0	1	Mon Apr 06 22:19:45 PDT 2009	@switchfoot http://twitpic.com/2y1zl - Awww, t...	aww bummer shoulda got david carr third day EM...	0.150	0
1	1	Mon Apr 06 22:19:49 PDT 2009	is upset that he can't update his Facebook by ...	upset update facebook texting might cry result...	0.000	0
2	1	Mon Apr 06 22:19:53 PDT 2009	@Kenichan I dived many times for the ball. Man...	dived many time ball managed save 50 rest go b...	0.500	0
3	1	Mon Apr 06 22:19:57 PDT 2009	my whole body feels itchy and like its on fire	whole body feel itchy like fire	0.200	0
4	1	Mon Apr 06 22:19:57 PDT 2009	@nationwideclass no, it's not behaving at all....	behaving mad see	-0.625	1

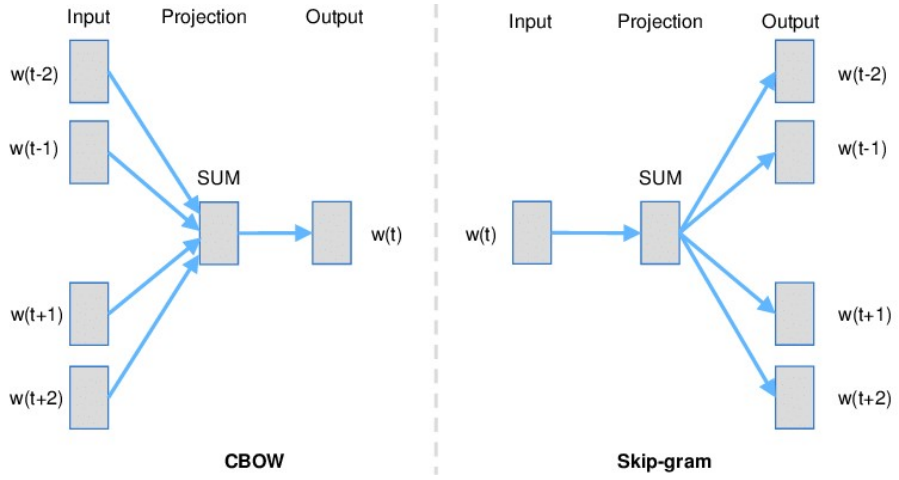
Section 2(d): The sentiment analysis (Word2Vec with bi-LSTM)

Using word embedding techniques has been a popular approach to transforming textual data into numerical data that can be processed by computers. This is due to its relative accuracy as compared to one hot encoding or relative efficiency as compared to the BERT transformer technique. The word embedding technique extracts words from all tweets and trains shallow neural layers to generate sentence-level features by analysing all the segments of sentences. The vector of sentence-level features is then passed through neural network

layers and the output is the prediction. In another way, the word embedding technique allows us to capture the context of a word by its semantic and syntactic similarity, with relations to other words in the same document.

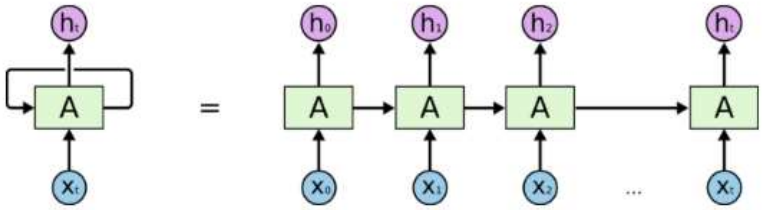
There are several popular word embedding frameworks. For instance, the Word2Vec framework. It is developed by Google in 2013 and is one of the most used word embedding frameworks. It can find similarities among words using cosine similarity. Thus, allowing it to learn associations amongst words. Similar words that share similar contexts in the corpus are positioned near one another in the vector representation. Generally, the larger the vector size of the generated word embedding, the more accurate is the model. However, that comes with extra computational costs. The size of the embedding vector is defined as 100, which generally is the minimum requirement.

Word2Vec operates on neural network architecture, and it has 2 main variants i.e., CBOW and Skip-gram. CBOW uses many context words to predict a target word and is faster and better represents frequent words. On the other hand, Skip-gram uses a word to predict target context words and is slower and better represents rare words. For the sake of efficiency only, I decided to go with CBOW. Below is a graphical representation of the two Word2Vec variants considered.



Now that I have decided to use Word2Vec with the CBOW variant, let's consider the neural language models that I could use to perform the classification task. A type of neural network model, the Recurrent Neural Networks (RNN) model is famously more suitable for textual data compared to other neural network architecture types like Convolutional Neural Networks (CNN). The main reason RNN is more suitable for textual data is that it uses its internal memory to process sequences of inputs. Therefore, unlike other neural networks type, all the inputs are not independent of each other. Shown in the diagram below, is the flow of inputs in a generic RNN. It shows that in RNN, the X0 input is processed and outputs as h0.

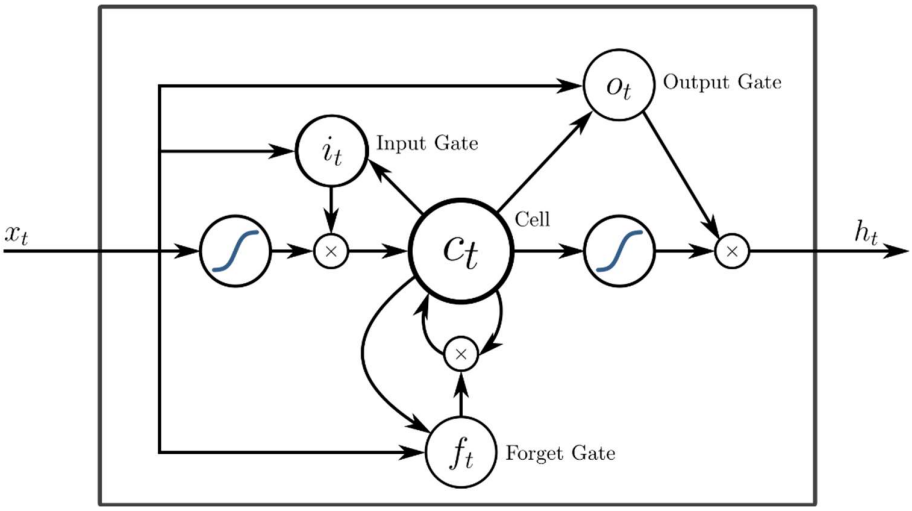
Then, the output is used in the next step making the input X1 + h0. And the process continues in this manner making RNN remember the context while being trained.



To use the RNN architecture, I must consider its parameters such as the current state, the activation function, and the output.

Starting from the aforementioned knowledge, I consider different variants of RNN architecture, which are basic RNN and RNN with Long-Short-Term Memory (LSTM). These considerations are necessary since RNN is known for its short-term memory problem, due to vanishing gradient. What this means practically is that the basic RNN model pays more attention to words that appear at the end of the sentence, rather than giving attention to all parts of the sentence. This problem should be resolved by using RNN with LSTM.

The defining difference between basic RNN and RNN with LSTM is that the latter consists of gate layers in its architecture. The gate layers regulate the flow of information through the model. Shown in the diagram below are the gates in RNN with LSTM.



The input gate layer regulates the input into the model. It does so by the sigmoid function and tanh function such as below. The sigmoid function decides which input to be let

through (0/1) and the tanh function provides importance weightage to the inputs which are let through (-1 to 1).

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

The forget gate layer regulates what the model should forget. This task is performed by the sigmoid function presented below. Depending on the h-1 state and Xt the input content, it calculates an output which ranges from 0 to 1. 0 means forget and 1 means remember.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

And lastly, the output gate regulates the output from the model. And this is achieved by the sigmoid and tanh functions. The mechanisms of these 2 functions are the same as in the input gate, except for the last part where the tanh function is multiplied with the output of the sigmoid function, as presented below.

$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

To set up word2vec in Python, the first step is splitting the dataset into training and test dataset. The test dataset is set to 30% of the whole dataset. Tokenising and padding are also performed to ensure that the texts can be converted into embeddings. Afterwards, the embedding matrix is created, which represents the words on the dataset, and is used as part of the RNN architecture.

The ‘callback’ function is also defined by using ReduceLROnPlateau and EarlyStopping from the ‘TensorFlow’ library. The former reduces the learning rate whenever the gain in the performance metric specified stops improving. The latter stops training when a monitored metric has stopped improving. Finally, both word2vec models are compiled and trained with the vector size argument set to 100. Considering all facts presented in this section, below is the summary of the word2vec models to be used in this project. The results are presented and discussed in the next section.

Model	Parameters
Word2Vec, RNN	Layer (type) Output Shape Param # embedding_1 (Embedding) (None, 60, 100) 6000000 conv1d_1 (Conv1D) (None, 56, 100) 50100 global_max_pooling1d_1 (GlobalMaxPooling1D) (None, 100) 0 dense_2 (Dense) (None, 16) 1616 dense_3 (Dense) (None, 1) 17 Total params: 6,051,733 Trainable params: 51,733 Non-trainable params: 6,000,000
Word2Vec, RNN with bi-LSTM	Layer (type) Output Shape Param # embedding (Embedding) (None, 60, 100) 6000000 bidirectional (Bidirectional) (None, 60, 200) 160800 bidirectional_1 (Bidirectional) (None, 60, 200) 240800 conv1d (Conv1D) (None, 56, 100) 100100 global_max_pooling1d (GlobalMaxPooling1D) (None, 100) 0 dense (Dense) (None, 16) 1616 dense_1 (Dense) (None, 1) 17 Total params: 6,503,333 Trainable params: 503,333 Non-trainable params: 6,000,000

Section 3: Experimental results

In this section, the classification results from the baseline model, and 2 RNN models are presented. Some auxiliary results are also presented such as word similarities and word clouds.

Model	Training Accuracy	Running Time
Baseline – TextBlob	60%	5 mins
Word2Vec, RNN (vanilla)	n/a (gradient vanished)	1.5 hours
Word2Vec, RNN with bi-LSTM	80%	3 hours

Note that the vanilla model terminated while being trained. Observing the evolution of the loss function, this happens due to a vanishing gradient. A model suffers from a vanishing gradient when large errors accumulate and result in very large updates to the model weights during training. This implies that the specified model is unstable and unable to learn from the training data. The training is then stopped by the 'callback' function. This occurrence is explained more in detail in section 4.

The RNN model with LSTM cell however arrives at 80% accuracy after 12 epochs. The results below show the accuracy of the model on the test dataset. The results imply that the model is slightly better at predicting non-depressed tweets than depressed ones. However, the overall accuracy is the same as on the training data.

	precision	recall	f1-score	support
0	0.79	0.81	0.80	240334
1	0.81	0.78	0.79	239666
accuracy			0.80	480000
macro avg	0.80	0.80	0.80	480000
weighted avg	0.80	0.80	0.80	480000

Presented here are word similarities of relevant tokens of interest i.e., depressed, depression. The metrics represent the cosine similarities between a word and the target context words.

```
#word similarities
word2vec_model.wv.most_similar('depressed')
```

✓ 0.4s

```
[('upset', 0.7423167824745178),
 ('homesick', 0.7191253304481506),
 ('stressed', 0.6789378523826599),
 ('annoyed', 0.6764383316040039),
 ('sad', 0.6757582426071167),
 ('sadder', 0.661658525466919),
 ('unhappy', 0.6602359414100647),
 ('angry', 0.6497887969017029),
 ('frustrated', 0.6476497650146484),
 ('anxious', 0.6437581181526184)]
```

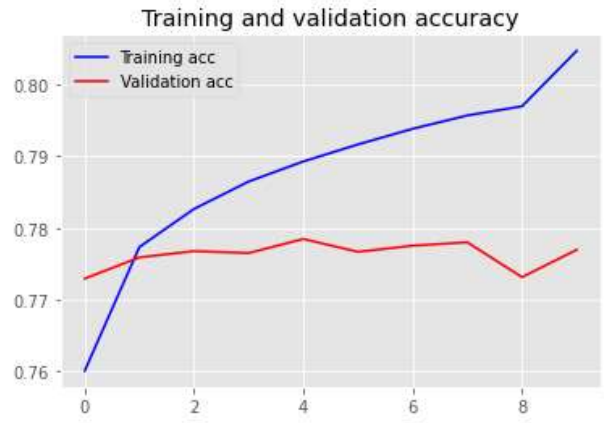
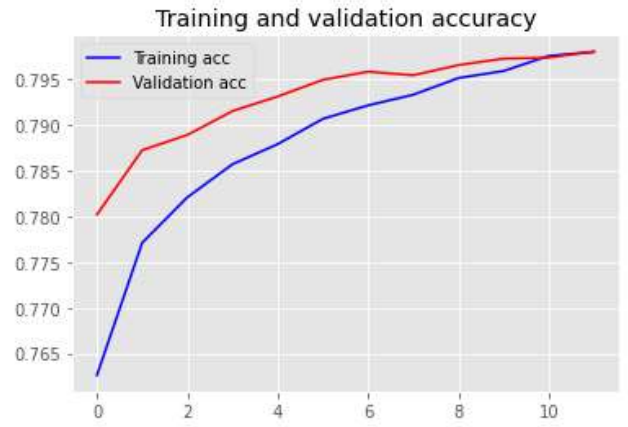
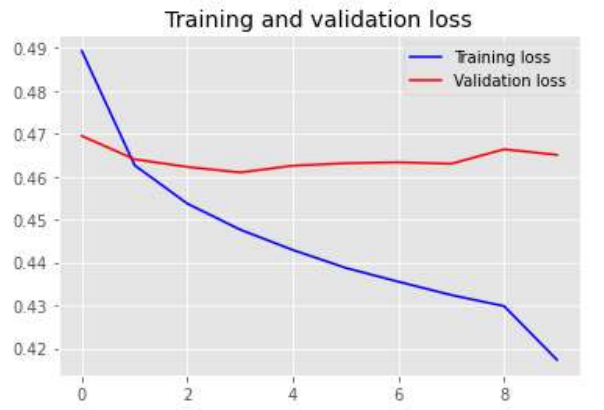
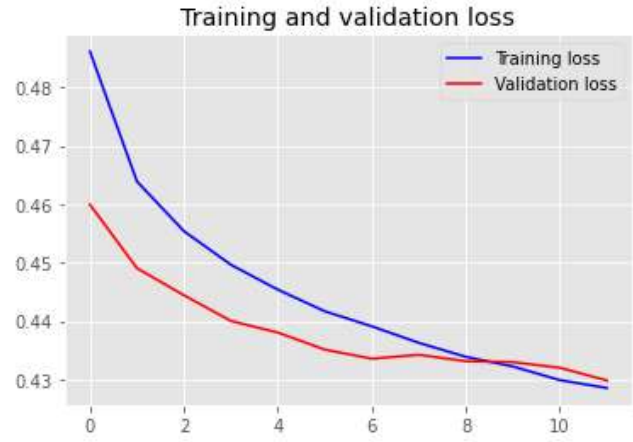
```
#word similarities
word2vec_model.wv.most_similar('depression')
```

✓ 0.6s

```
[('anxiety', 0.6708846688270569),
 ('separation', 0.6347039937973022),
 ('illness', 0.6050086617469788),
 ('detest', 0.5558192133903503),
 ('flare', 0.5542736649513245),
 ('redundancy', 0.5503567457199097),
 ('economic', 0.5468493700027466),
 ('statistic', 0.5434815287590027),
 ('traumatic', 0.5413870811462402),
 ('nausea', 0.5391903519630432)]
```

Section 4: Discussion and Conclusion

In this section, the vanishing gradient observed in the vanilla model is explained. Following that, a discussion on how the models can be improved is also presented. The paper is concluded with a discussion of future works.

Vanilla RNN	RNN with bi-LSTM
	
	

As observed above, the vanilla model terminated before epoch 10, out of 12 epochs scheduled during training and cannot be completed. The evolution and stop of the training accuracy and loss are observed in the first column of the table below. During the training, each step of the layer produces a derivative. If the derivatives are large, then the gradient decreases exponentially until it vanished. What this means for the model is that it is unstable and not effective for learning. Fortunately, using the LSTM cell in the RNN model as in the second model solves the vanishing gradient problem.

With 80% accuracy, the RNN model with bi-LSTM cells performs quite well. However, an acknowledgement must be made here that the model can be further fine-tuned. Some have been experimented with such as CBOW vs skip-gram embedding generation, with or without

negative sampling, and the number of embedding dimensionalities. However, these tweaks are very computationally extensive and take too long to run. Thus, these tweaks will be experimented with further in the future.

Based on this project, I have identified that RNN with bi-LSTM cell performs well to analyse short textual data such as tweets. This finding is crucial for my future research plan which will involve analysing a similar dataset. Of course, further iterations mentioned in the previous paragraph will be performed as well. On top of that, BERT transformers are also considered for future work.

Declaration

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

Appendix

This project is performed entirely in Python programming language. The project code is shared here.

References

1. Aramaki, E., Maskawa, S., & Morita, M. (2011, July). Twitter catches the flu: detecting influenza epidemics using Twitter. In Proceedings of the 2011 Conference on empirical methods in natural language processing (pp. 1568-1576).
2. Baumann, A. E. (2007). Stigmatization, social distance and exclusion because of mental illness: the individual with mental illness as a 'stranger'. International review of psychiatry, 19(2), 131-135.
3. Coppersmith, G., Dredze, M., & Harman, C. (2014, June). Quantifying mental health signals in Twitter. In *Proceedings of the workshop on computational linguistics and clinical psychology: From linguistic signal to clinical reality* (pp. 51-60).
4. De Choudhury, M., & De, S. (2014, May). Mental health discourse on reddit: Self-disclosure, social support, and anonymity. In *Eighth international AAAI conference on weblogs and social media*.
5. Go, A., Bhayani, R. and Huang, L., 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford, 1(2009), p.12*.

6. Hawn, C. (2009). Take two aspirin and tweet me in the morning: how Twitter, Facebook, and other social media are reshaping health care. Health affairs, 28(2), 361-368.
7. Peek, N., Combi, C., Marin, R., & Bellazzi, R. (2015). Thirty years of artificial intelligence in medicine (AIME) conferences: A review of research themes. Artificial intelligence in medicine, 65(1), 61-73.