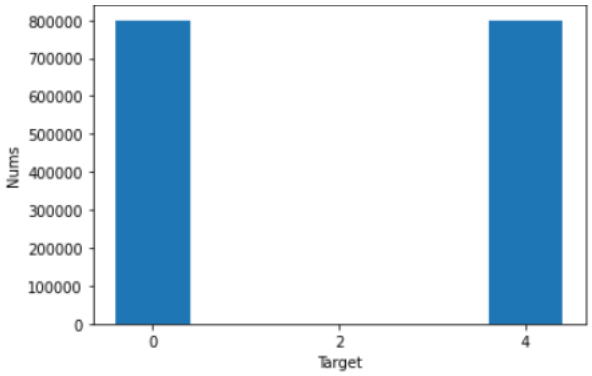
Task 1

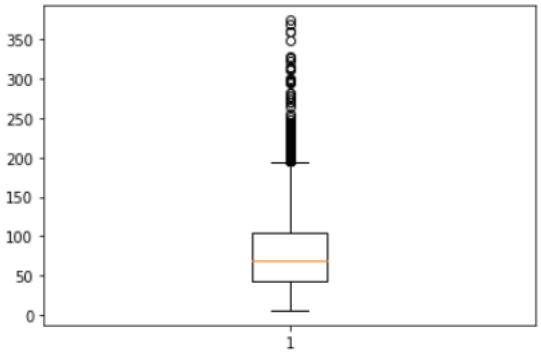


As the figure shows, there is zero Neutral text. The distribution of negative and positive texts is balanced.

Task2

Columns in ['id','date','flag','user']. Since the emotion of text sentences is definitely irrelavent with such features. I only keep the text itself and label.

Task3



The figure shows the box plot of the length of each text in the corpus. It seems that many tweets have number of characters larger than 140. But most of tweets have length within 150. The reason is the data is not clean. For example, some marks from HTML.

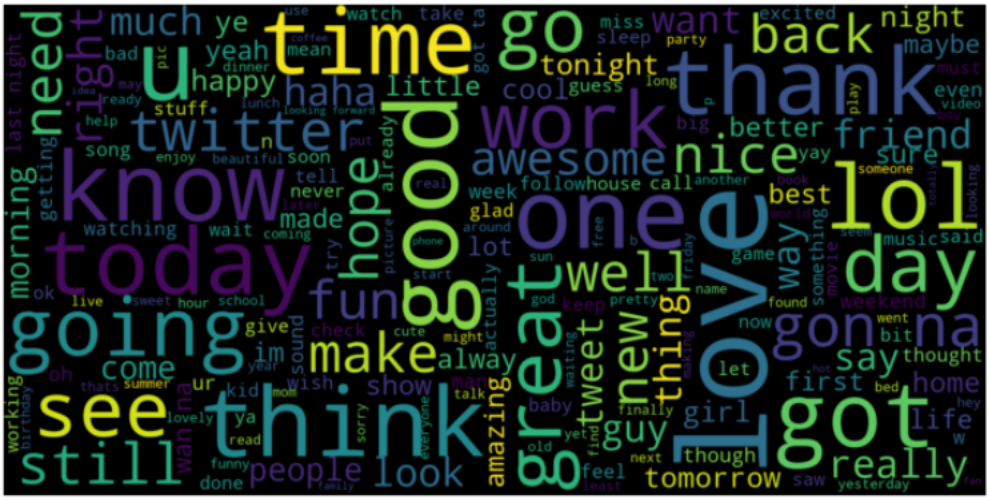
Task4 + Task5+Task6+Task7

Algorithm 1 Data Processing Pipeline
Input : Data Frame d
Output: Clean Data Frame d' used to train
Drop columns "id", "date", "flag", "user"
Use Library BeautifulSoup to remove HTML notations of texts
Remove mention, URL Link, #, punctuations mark, and numbers using Regrex
Remove All stop words in each text, based on NLTK stop word corpus
Transform all characters to lowercase
USE NLTK to lemmatize each word in each sentence
USE TF-IDF vectorize (max features = 512) to transform sentences to vectors.
Return Data Frame d'

Algorithm 1 demonstrates the process of data processing.

Task 8

The following figure demonstrates the word cloud of positive text. It is obvious that some words with certain size are positive, such as good, great, lol, awesome, happy, .etc. It is hard to see words are negative.



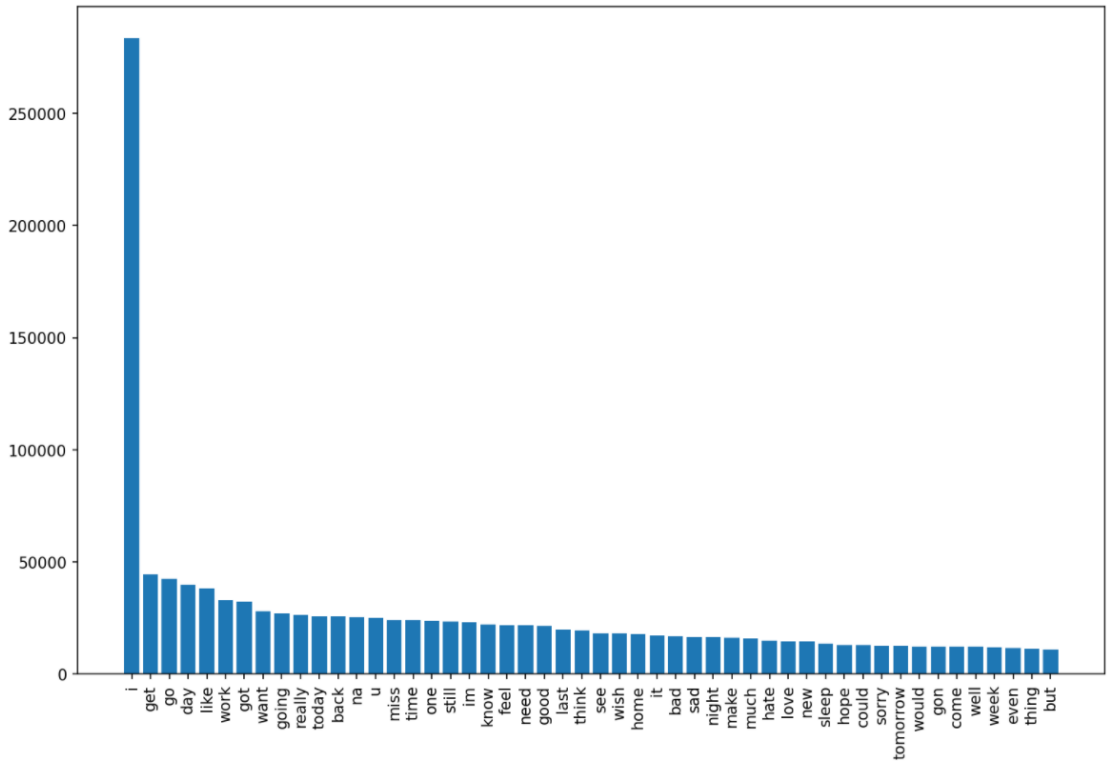
The following figure demonstrates the word cloud of negative text. It is obvious that some words with certain size are negative, such as bad, sad, tired. But you can still see some words are positive, likes love, happy.

Task 9

		negative	positive	total
1	i	283393	204299	487692
2	day	39820	41569	81389
3	get	44580	33934	78514
4	like	38051	33221	71272
5	go	42329	26020	68349
6	good	21417	41454	62871
7	u	25011	32873	57884
8	got	32130	25495	57625
9	love	14428	38063	52491
10	work	32913	16566	49479
11	time	23917	25450	49367
12	going	26886	21532	48418
13	today	25802	20441	46243
14	one	23579	22324	45903
15	know	22069	21954	44023
16	back	25623	18034	43657
17	really	26220	15916	42136
18	na	25278	16016	41294
19	see	18147	22171	40318
20	want	27962	12054	40016

Task 10

The following Figure shows the bar chart of top-50 negative words. The word with most frequency is not really negative, such as I, get, go, and day. They are common words in conservation or paragraphs. They will also on the list of top-50 positive words, either. In this case, Plot the bar chart of top-50 negative words cannot really give us a straightforward understand of it. The things really matter are we should find words that are rare but highly correlated to negative label.



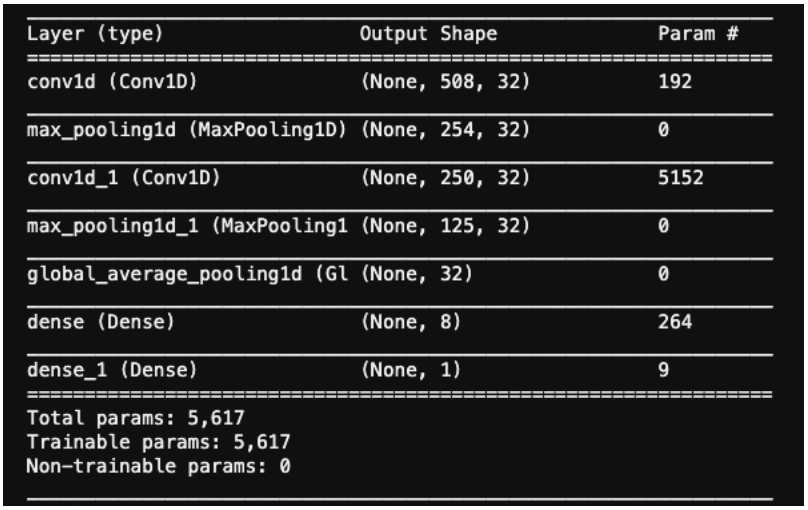
Task 11

Logistic Regression, KNN, CNN, and GRU are used as models.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression. (Or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. In the model, default parameters of sklearn package are used: L2 penalty; intercept = True; Maximum itrations = 100.

KNN, or K-nearest-neighbors, is a classification algorithm used to find top-k nearest neighbors of given test data and return the most of labels. Here, I use 11-NN as the machine learning models.

A convolutional model is used as the first Deep learning model. The structure is shown as followed figure. There are two convolutional 2D layers with kernel size = 5*5, each followed by a Maxpooling1D layer. Then, a globalAverage1D layer is used to flatten the feature map. Then a 8-unit dense layer and 1-unit dense layers are used to predict the probability. Since the data is extremely large and computation resources & time is limited, a simple model is used.



Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 508, 32)	192
max_pooling1d (MaxPooling1D)	(None, 254, 32)	0
conv1d_1 (Conv1D)	(None, 250, 32)	5152
max_pooling1d_1 (MaxPooling1	(None, 125, 32)	0
global_average_pooling1d (Gl	(None, 32)	0
dense (Dense)	(None, 8)	264
dense_1 (Dense)	(None, 1)	9

Total params: 5,617
Trainable params: 5,617
Non-trainable params: 0

The gated recurrent unit is used as another deep learning model. Since the time is limited and data is large, a model with few parameters is used. There is 4 GRU units, followed by 4 dense layers and 1 dense layer. However, the time cost of each epoch is still approximate to half hour.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_4 (GRU)                  (None, 4)                 84
_________________________________________________________________
dense_16 (Dense)             (None, 4)                 20
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 5
=================================================================
Total params: 109
Trainable params: 109
Non-trainable params: 0
```

All of models are trained on FULL training set, Test on test set. There is no hyper-parameters for machine learning model. For deep learning model, following setting is used:

Epoch = 10

Batch size = 128

Optimizer = Adam, with initial learning rate 0.003, decays to its 95% every 2 epochs.

Early Stopping of monitoring val_loss for 5 epochs is used.

Task 12

Model	Accuracy
Logistic Regression	51.80%
KNN	5160%
CNN	38.55%
GRU	40.01%

Analysis:

The overall accuracy of models is not high. Both the deep learning models fail to beat a random guess. But from other perspective, if we predict the label opposite to the prediction, it may work.

There are two reasons that can explain why the accuracy is too low. First, during the vectorization process, the word vector is transformed to tf-idf vector. However, if the dimension of tf-idf is not restricted, the dimension will be approximate to 250,000, which will lead to an extremely sparse matrix, it is impossible to train. So, I restricted the maximum dimension to 512. But compared to 250,000, the number is really small (but still cost many time and memory to train), many useful information is loss. Furthermore, the model (especially for DNN models), the parameters are so few so that model does not have ability to fit the data. The result is more like a random guess on positive or negative.