# Sentiment Analysis in Tweet Classification

Yi Lin

April 7, 2021

## 1 Introduction

This project aims to build up a text classification model in Python. Under the uncertainty of the pandemic period, the motivation behind is to build a system that measures how the perception of the epidemiological situation change in different regions. The system helps various deferral agencies to plan their next move.

Using Natural Language Processing (NLP) to extract the significant pattern of information from Tweet, we use sentiment analysis to analyze the local COVID-19 situation underlying the emotions of tweet description and comments. Sentiments can classify the words into 5 categories: (0) Extremely Negative, (1) Negative, (2) Neutral, (3) Positive, (4) Extremely Positive.

This project is in both lexicon-based machine learning-based classification. We train the model using a traditional ML algorithm with labeled data and then apply it to predict a class for a new text [1]. Moreover, I develop a deep learning model to classify Tweets comments in Python with Keras automatically. Specifically, I apply word embedding while training Convolutional Neural Network (CNN) on the tweet classification. Also, I use the Bag-of-Words methods (Term Frequency-Inverse Document Frequency (TF-IDF) and Word Count Vectors) while training Multinomial Naïve Bayes (MNB) on tweet classification.

## 2 Data Preparing

There are three data documents in CSV format. The *ps5_tweets_text.csv* contains labeled Tweet messages. The *ps5_tweets_labels.csv* contains labels in text format, while the *ps5_tweets_labels_as_numbers.csv* contains labels in numeric format. I first use Pandas and NumPy to read CSV format. I then combined three datasets as one dataset via pd.concat. Here is Figure 1.

Figure 1: Uncleaned Dataset

| | Tweet | Sentiment | Label | id |
|---|---|---|---|---|
| 0 | https://t.co/UpjxfOgQs8\r\n\r\nGaisss! Ple... | Extremely Positive | 4 | 1 |
| 1 | @mygovindia Today just after a week of lockdow... | Negative | 1 | 1 |
| 2 | Tuskys partners with Amref to provide on groun... | Neutral | 2 | 1 |
| 3 | @chrissyteigen are u doing ur own grocery shop... | Negative | 1 | 1 |
| 4 | UK Critical Care Nurse Cries at Empty SuperMar... | Extremely Negative | 0 | 1 |
| 5 | @ymxr6 Makes my heart ache its the elderly tha... | Extremely Negative | 0 | 1 |
| 6 | COVID-19 wrecks aluminium prices and input cos... | Neutral | 2 | 1 |
| 7 | February Home Prices Increased by 4.1 Percent ... | Positive | 3 | 1 |
| 8 | Want advice on avoiding scams related to #COVI... | Extremely Negative | 0 | 1 |
| 9 | @dailyecho @BBCWatchdog @BBCNews @dailymail an... | Negative | 1 | 1 |

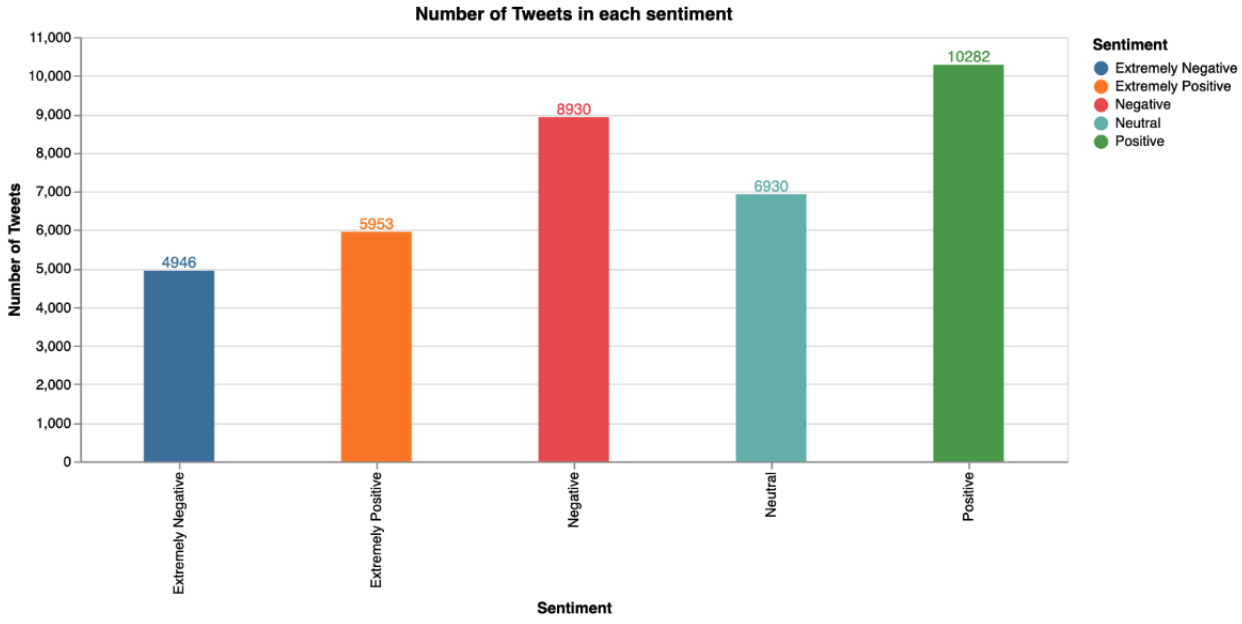
# 3 Exploratory Data Analysis

Since the nature of text-based features, we only do some shallow analysis to get a simple insight into the dataset. I draw two histogram plots based on *seaborn* and *altair.altair* packages from Python [2].

## 3.1 Number of Tweet in each sentiment

Based on Figure 2, we can visualize that the dataset has 37041 tweet messages in total, consisting 4946 Extremely Negative(0), 8930 Negative(1), 6930 Neutral(2), 10282 Positive(3), 5953 Extremely Positive(4).

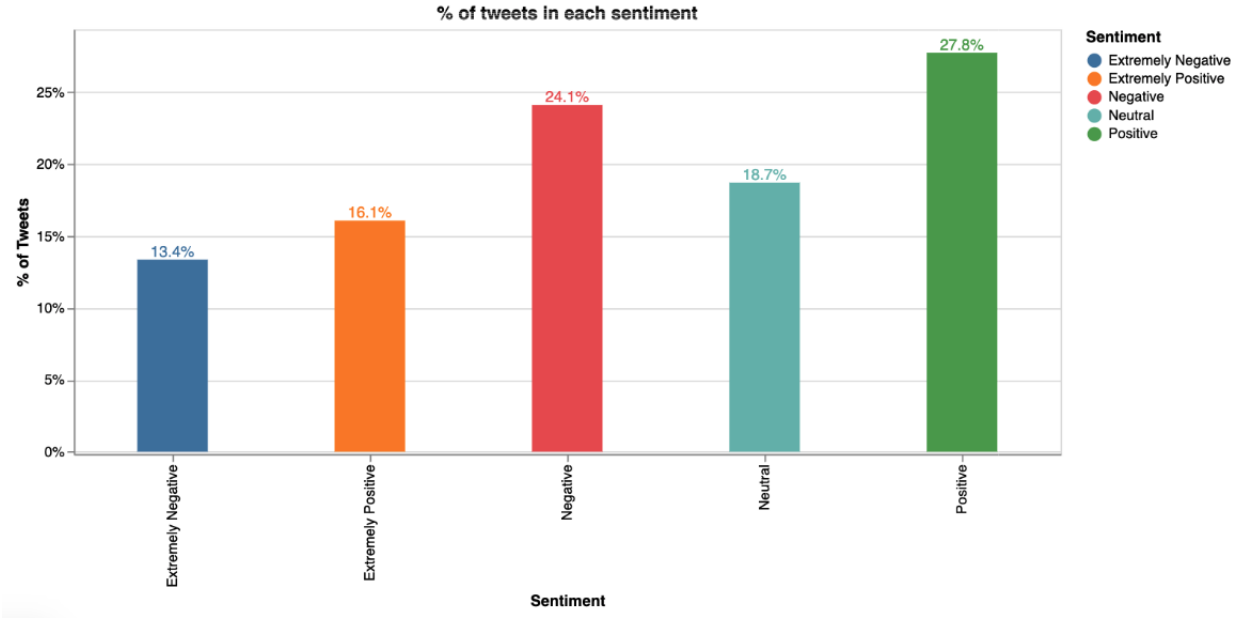
Figure 2: Number of Tweet in each sentiment



### 3.2 Percentage of Tweet in each sentiment

Based on Figure 3, we can get the percentage of tweet belonging to each sentiment. The dataset contains 13.4% Extremely Negative(0), 24.1% Negative(1), 18.7% Neutral(2), 27.8% Positive(3), 16.1% Extremely Positive(4).

Figure 3: Percentage of Tweet in each sentiment




# 4 Data Preprocessing

To clean up the dataset, I use Natural Language Toolkit (NLTK) to tokenize all words [3]. Here are several ways to clean the dataset:

1. Remove special character, URL, retweet (ex. '#', '@', 'http://' )
2. Lowercase (ex. downcase from 'Apple' to 'apple')
3. Punctuation Signs (ex. '?', '!', ';', '{}')
4. Possessive Pronouns (ex. 'government's' to 'government')
5. Stopwords (ex. 'i', 'me', 'our', 'ourselves', 'you', 'your', 'yours')
6. Lemmatize via WordNetLemmatizer (ex. lemmatrize 'rocks' to 'rock', 'better' to 'good')

After data preprocessing, I combine both cleaned Tweet messages and labels as a new data frame in Figure 4.

Figure 4: Cleaned Dataset

| | Tweet_Clean | Label |
|---|---|---|
| 0 | gaisss please read this and please limit yours... | 4 |
| 1 | today just after a week of lockdown lot of con... | 1 |
| 2 | tuskys partner with amref to provide on ground... | 2 |
| 3 | be u do ur own grocery shopping now like a reg... | 1 |
| 4 | uk critical care nurse cry at empty supermarke... | 0 |
| ... | ... | ... |
| 37036 | minnesota classifies grocery store worker a em... | 1 |
| 37037 | u senator have ask for information about a the... | 1 |
| 37038 | just comment on poll be you do more online sho... | 0 |
| 37039 | my wife get lay off yesterday because the smal... | 2 |
| 37040 | humanity be doom | 0 |

37041 rows × 2 columns

# 5 Model Building: Word Embedding & CNN

## 5.1 Dataset Split

I chose to random split with 80% of the tweet messages composing the training set and 20% of the tweet messages composing the testing set. For the deep learning models in terms of CNN, I applied the validation split (=0.2) when fitting the model so that CNN is 60-20-20 in train-validate-test sets.

## 5.2 Word Embedding Layers with Keras

To define a word, I use word embedding to develop a term as a Counter. The Counter as a dictionary mapping of words allows us to easily query or update their counts [4]. Word embedding is the method of showing text where each vocabulary is indicated by a real-valued vector in a high-dimensional space[4]. Here, I use the Keras Embedding layer, which asks integer inputs where each integer maps a single token. The single token has its specific real-valued vector showing within the embedding.

I first encode the training set to convert string words into numbers (text to sequences of integers) using Tokenizer in Keras API. So far, I develop a consistent mapping from words to unique integers and ensure all documents with identical length. To truncate the tweet comments from the smallest size to the longest or maximum length, I use *max()* function to find out the longest length comments. The maximum length of comment has 66 words. Then I use *pad_ sequences()* Keras function to pad the sequence to the longest length by adding '0' values at the end. In this case, I also encode and pad the testing set. After training the CNN, I evaluate the model with a testing set.

## 5.3 Convolutional Neural Network

The first hidden layer is Embedding layer, which requires the vocabulary size (24859 different vocabularies) and the size of real-valued vector space (a 100 element vector), and the longest length of comments (length of 66 words). Notice that the vocabulary size is the total number of words in vocabulary with one additional unknown words (24858 + 1 = 24859).

The second layer is the Cov1D layer. It includes 32 filters (32 parallel fields for processing words), and the kernel size of 8, and the activation function of 'relu'. This is followed by a Max pooling layer with size of 2 to reduce half of the output of the convolutional layer.

The following two layers are one flatten layer and then a dense layer of 10 units with activation function of 'relu'. Based on the Brownlee said, these two layers are trying to "flatten to one long 2D vector to represent the 'features' extracted by the CNN" [4].

Since we have 5 categorical sentiments, the last output layer applies the softmax activation function to generate the value among 0 and 4 for the 5 categorical sentiments in the tweet review. There is Figure 5.

Figure 5: Convolutional Neural Network

```
#Reference: https://machinelearningmastery.com/develop-word-embeddi
# define model
model = Sequential([])
model.add(Embedding(vocab_count, 100, input_length=max_length))
model.add(Conv1D(filters=32, kernel_size=8, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(5, activation='softmax'))
```

```
print(model.summary())
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, 66, 100)           2485900
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 59, 32)            25632
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 29, 32)            0
_________________________________________________________________
flatten_7 (Flatten)          (None, 928)               0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                9290
_________________________________________________________________
dense_15 (Dense)             (None, 5)                 55
=================================================================
Total params: 2,520,877
Trainable params: 2,520,877
Non-trainable params: 0
_________________________________________________________________
None
```

### 5.4 CNN Compile, Fit and Evaluation

To compile the CNN, I use the *sparse_ categorical_ crossentropy* loss function since there are 5 different classes instead of 2 classes, so I don't use the *binary_ crossentropy* loss function in this case. Then I use *Adam* optimizer to keep track of accuracy and loss during the training. Then the model is trained for batch-size of 32, validation split of 0.2, 5 epochs through the training set after padding. *model.fit(pad_ Xtrain, ytrain, epochs=5, validation_ split=0.2, batch_ size=32).*

Once the model is done with the fit, I use *model.evaluate()* function to evaluate the testing set and calculate the accuracy of the model. The accuracy of the model is 71.9%.

## 6 Model Building: Bag-of-Words & Multinominal NB

### 6.1 Data Split

I chose to random split with 80% of the tweet messages composing the training set and 20% of the tweet messages composing the testing set. For the traditional ML model like Multinomial NB, I perform the hyperparameter tunning process with Grid Search Cross-Validation ($GridSearchCV = 3$) in the training set. Then I fit the final model and evaluate the model by metrics[3].

### 6.2 Bag-of-Words (Word Counts & TF-IDF)

The Bag-of-Words for feature extraction do not consider the order of the terms in the comments, which represents text data when modeling text with ML techniques[5]. For both word counts and TF-IDF, I use the sklearn package.

For the Word Counts vector, each column is the term from the corpus, and each cell shows each term's frequency count in each comment [4]. The TF-IDF vector shows the importance of each term in the comment and the entire corpus. The equation of TF-IDF is Term Frequency (TF) * Inverse Document Frequency (IDF). TF measures the frequency of a term in a document, while IDF measures a term's informativeness. Figure 6 shows the details of the calculation work.

Figure 6: TF-IDF Equation

$$TFIDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right)$$

Being:

- $t$: term (i.e. a word in a document)
- $d$: document
- $TF(t)$: term frequency (i.e. how many times the term $t$ appears in the document $d$)
- $N$: number of documents in the corpus
- $DF(t)$: number of documents in the corpus containing the term $t$

### 6.3 Multinominal Naïve Bayes

I first create a Pipeline as an estimator containing "CountVectorizer," "TfidfTransformer," and classifier "MultinominalNB." Later, I use CountVectorizer() to fit and transformed the X-training set, and then use TfidfTransformer() to fit and transformed the results from the last part. Finally, I use MultinomialNB() to fit by using the transformed results and the labels y-training set using the default hyperparameter alpha value of 1.0 [6].

### 6.4 GridSearchCV with Multi-NB

I use 'GridSearchCV' with Pipeline to tune the hyperparameter 'alpha' in Multi-NB as its estimator. I first use GridSearchCV() as '*grid*' with 10-folds cross-validation to fit the training set. Then the '*grid*.score' way of the pipeline is notified where the 10-folds cross-validation is transformed (not fit) by using both CountVectorizer() and TfidfTransformer() accordingly. Finally, the '*grid*.score' way of MultinominalNB() is notified to compute the model's accuracy on the transformed cross-validation testing set. *score = grid.score(Xtest, ytest)*. Finally, the accuracy of the Multinomial-NB model on the testing set is around 45.7%. Notice that when alpha = 0.06, the accuracy of Multi-NB on the testing is around 73.6%, which improves the accuracy of the model roughly 38.7% when alpha = 1.0

## 7 Conclusion

In this project, I use word embedding while training the Convolutional Neural Network and use bag-of-words while training the Multinominal-NB. Based on the observation, the deep learning model with CNN has better performance than the traditional ML algorithms. The CNN accuracy is up to 73.6%, whereas the Multinomial-NB accuracy is 45.7%. Word embedding is the method that words with similar meaning have similar representation in the vector space; on the other hand, the traditional way of bag-of-words would ignore the relationship between words and tokens.

# References


[1] P. Harjule, A. Gurjar, H. Seth and P. Thakur, "Text Classification on Twitter Data," *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, Jaipur, India, 2020, pp. 160-164, doi: 10.1109/ICETCE48199.2020.9091774.

[2] Zafra, M. (2020, May 25). Text classification in python. Retrieved April 08, 2021, from https://towardsdatascience.com/text-classification-in-python-dd95d264c802

[3] Brownlee, J. (2019, August 07). How to clean text for machine learning with python. Retrieved April 08, 2021, from https://machinelearningmastery.com/clean-text-machine-learning-python/

[4] Brownlee, J. (2020, September 02). Deep convolutional neural network for sentiment ANALYSIS (text classification). Retrieved April 08, 2021, from https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/

[5] Brownlee, J. (2019, August 07). A gentle introduction to the bag-of-words model. Retrieved April 08, 2021, from https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[6] Tonypeng1. (2019, October 26). Tf-Idf with multinomial nb and cross validation. Retrieved April 08, 2021, from https://www.kaggle.com/tonypeng1/tf-idf-with-multinomial-nb-and-cross-validation/comments