Sentiment Analysis Of COVID-19 Tweets - Visualization Dashboard

#IBMHackChallenge2020

Smart Internz

AKASH SAJJAN
Infinity Creations Group
Linked-In
Git-Hub
Bangalore Technological Institute
Bangalore - 560035

Table of contents

1. INTRODUCTION

1.1 Overview

This project of analyzing sentiments of tweets comes under the domain of “Pattern Classification” and “Data Mining”. Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering “useful” patterns in large set of data, either automatically (unsupervised) or semi automatically (supervised). The project would heavily rely on techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and on “Machine Learning” techniques for accurately classifying individual un-labelled data samples (tweets) according to whichever pattern model best describes them. The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word Project Thesis Report 11 lengthening [13], question marks, presence of url in tweets, exclamation marks, internet emoticons and internet shorthand/slangs. Classification techniques can also be divided into a two categories: Supervised vs.unsupervised and non-adaptive vs. adaptive/reinforcement techniques.

1.2 Purpose

The objectives of the study are first, to study the sentiment analysis in microblogging which in view to analyze feedback from a customer of an organization's product; and second, is to develop a program for customers' review on a product which allows an organization or individual to sentiment and analyzes a vast amount of tweets into a useful format.

Humans are fairly intuitive when it comes to interpreting the tone of a piece of writing.

Consider the following sentence: "My flight's been delayed. Brilliant!" Most humans would be able to quickly interpret that the person was being sarcastic. We know that for most people having a delayed flight is not a good experience (unless there's a free bar as recompense involved). By applying this contextual understanding to the sentence, we can easily identify the sentiment as negative. Without contextual understanding, a machine looking at the sentence above might see the word "brilliant" and categorise it as positive.

The aim of this project is to present a model that can perform sentiment analysis of real data collected from Twitter. Data in Twitter is highly unstructured which makes it difficult to analyze. However, our proposed model is different from prior work in this field because it combined the use of supervised and unsupervised machine learning algorithms. The process of performing sentiment analysis as follows: Tweet extracted directly from Twitter API, then cleaning and discovery of data performed. After that the data were fed into several models for the purpose of training. Each tweet extracted classified based on its sentiment whether it is a positive, negative or neutral. Data were collected on two subjects McDonalds and KFC to show which restaurant has more popularity.

The result from these models were tested using various testing metrics like cross validation and f-score. Moreover, our model demonstrates strong performance on mining texts extracted directly from Twitter.

2. LITERATURE SURVEY

2.1 Existing problem

The existing system which have been design have a simple structure and are only used for a particular type of feed. Most of the system which had been developed doesn't have a proper prototype and implementation of the systems which is already done is difficult.

Despite the availability of software to extract data regarding a person's sentiment on a specific product or service,organizations and other data workers still face issues regarding the data extraction.

• Sentiment Analysis of Web Based Applications Focus on Single Tweet Only. With the rapid growth of the World Wide Web, people are using social media such as Twitter which generates big volumes of opinion texts in the form of tweets which is available for the sentiment analysis. This translates to a huge volume of information from a human viewpoint which make it difficult to extract a sentences, read them, analyze tweet by tweet, summarize them and organize them into an understandable format in a timely manner.

• Difficulty of Sentiment Analysis with inappropriate English Informal language refers to the use of colloquialisms and slang in communication, employing the conventions of spoken language such as 'would not' and 'wouldn't'. Not all systems are able to detect sentiment from use of informal language and this could hanker the analysis and decisionmaking process. Emoticons, are a pictorial representation of human facial expressions , which in the absence of body language and prosody serve to draw a receiver's attention to the tenor or temper of a sender's nominal verbal communication, improving and changing its interpretation. For example, ☺ indicates a happy state of mind. Systems currently in place do not have sufficient data to allow them to draw feelings out of the emoticons. As humans often turn to emoticons to properly express what they cannot put into words . Not being able to analyze this puts the organization at a loss. Short-form is widely used even with short message service (SMS). The usage of short-form will be used more frequently on Twitter so as to help to minimize the characters used. This is because Twitter has put a limit on its characters to 140.

2.2 Proposed Solution

In the previous projects, most of the sentiment analysis have been done in binary classification or 3-way classification. Multiclass classification of sentiment analysis using basic human emotions are being researched now-a-days. Now here we will discuss about our framework on sentiment analysis using basic human emotions.

The proposed architecture of four modules: user interface, log pre-processing, Feature Clustering using Modified K-means, Naïve Bayes Classification, Training and testing using KNN for more accurate categorization of opinion. This system can solve irrelevant data and more accuracy by associating Modified K means with Naïve Bayes Classification algorithm.

A dataset is created using twitter posts of electronic products. Tweets are short messages with full of slang words and misspellings. So we perform a sentence level sentiment analysis. This is done in three phases. In first phase preprocessing is done. Then a feature vector is created using relevant features. Finally using different classifiers, tweets are classified into positive and negative classes. Based on the number of tweets in each class, the final sentiment is derived.

1. Creation of a Dataset

Since standard twitter dataset is not available for electronic products domain, we created a new dataset by collecting tweets over a period of time ranging from April 2013 to May 2013. Tweets are collected automatically using Twitter API and they are manually annotated as positive or negative. A dataset is created by taking 600 positive tweets and 600 negative tweets. Table 1 shows how dataset is split into training set and test set.

2. Preprocessing of Tweets

Keyword extraction is difficult in twitter due to misspellings and slang words. So to avoid this, a preprocessing step is performed before feature extraction. Preprocessing steps include removing url, avoiding misspellings and slang words. Misspellings are avoided by replacing repeated characters with 2 occurrences. Slang words contribute much to the emotion of a tweet. So they can’t be simply removed. Therefore a slang word dictionary is maintained to replace slang words occurring in tweets with their associated meanings. Domain information contributes much to the formation of slang word dictionary.

3. Creation of Feature Vector

Feature extraction is done in two steps. In the first step, twitter specific features are extracted. Hashtags and emoticons are the relevant twitter specific features. Emoticons can be positive or negative. So they are given different weights. Positive emoticons are given a weight of ’1’ and negative emoticons are given a weight of ’-1’. There may be positive and negative hashtags. Therefore the count of positive hashtags and negative hashtags are added as two separate features in the feature vector. Twitter specific features may not be present in all tweets. So a further feature extraction is to be done to obtain other features. After extracting twitter specific features, they are removed from the tweets. Tweets can be then considered as simple text. Then using unigram approach, tweets are represented as a collection of words. In unigrams, a tweet is represented by its keywords. So their presence is also added as a relevant feature. All keywords cannot be treated equally in the presence of multiple positive and negative keywords.

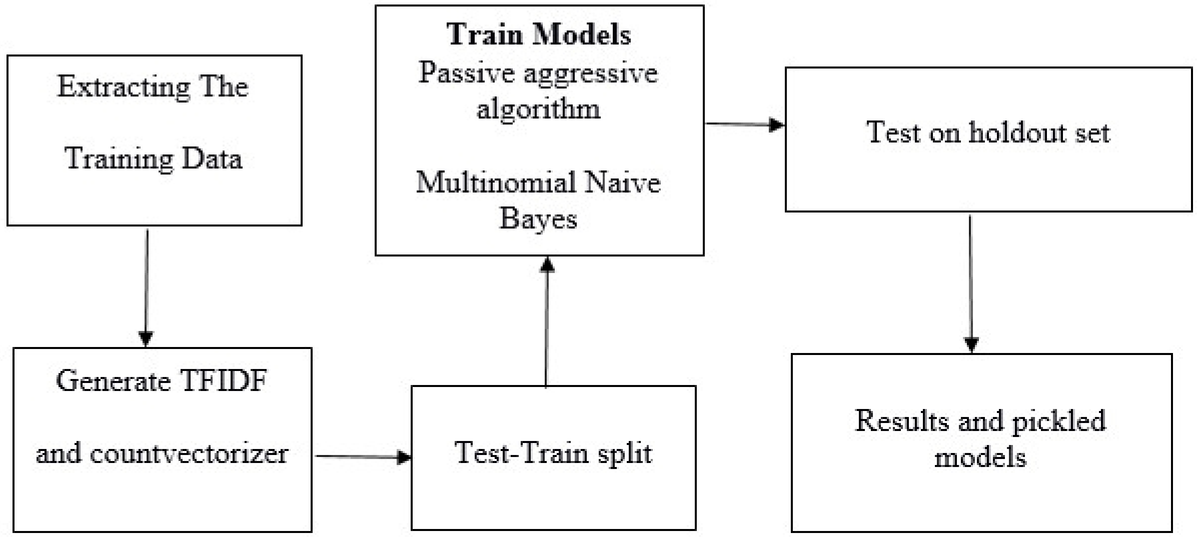
Therefore a special keyword is selected from all the tweets. In the case of tweets having only positive keywords or only negative keywords, a search is done to identify a keyword having relevant part of speech. A relevant part of speech is adjective, adverb or verb. Such a relevant part of speech is defined based on their relevance in determining sentiment. Keywords that are adjective, adverb or verb shows more emotion than others. If a relevant part of speech can be determined for a keyword, then that is taken as special keyword. Otherwise a keyword is selected randomly from the available keywords as special keyword. If both positive and negative keywords are present in a tweet, we select any keyword having relevant part of speech. If relevant part of speech is present for both positive and negative keywords, none of them is chosen. Special keyword feature is given a weight of ’1’ if it is positive and ’-1’ if it is negative and ’0’ in its absence. Part of speech feature is given a value of ’1’ if it is relevant and ’0’ otherwise. Thus feature vector is composed of 8 relevant features. The 8 features used are part of speech (pos) tag, special keyword, presence of negation, emoticon, number of positive keywords, number of negative keywords, number of positive hash tags and number of negative hash tags.

4. Sentiment Classification

After creating a feature vector, classification is done using Naive Bayes, Support Vector Machine, Maximum Entropy and Ensemble classifiers and their performances are compared.

3 THEORITICAL ANALYSIS

3.1 Block Diagram


Extracting The Training Data
Train Models
Passive aggressive algorithm
Multinomial Naive Bayes
Test on holdout set
Generate TFIDF and countvectorizer
Test-Train split
Results and pickled models

3.2 Hardware / Software designing

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

Architectural Design:

It emphasizes the design of the system architecture that describes the structure, behaviour and many more views of that system and analysis. The high-level design breaks the system's architectural design into a less-abstracted view of sub-systems and modules and depicts their interaction with each other.

Logical Design:

The logical design of a system pertains to an abstract representation of data flows, input and outputs of the system. Logical design includes modelling of entity-relationship diagram.

Physical Design:

The physical design relates to the actual input and output processes of the system. Detailed design involves the implementation of what is visible as a system and its sub-systems in a high-level design. This activity is more detailed towards modules and their implementations. It defines a logical structure of each module and their interfaces to communicate with other modules. It involves of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed. In physical design, there are several sub-tasks.

User Interface Design:

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them.

Data Design:

Data Design is concerned with how the data is represented and stored within the system.

Process Design:

Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system.

4. EXPERIMENTAL ANALYSIS

1. Neethu M S, Rajasree R, IEEE, July 2013. Sentiment Analysis in Twitter Using Machine Learning Techniques. In this paper, we try to analyze the twitter posts about electronic products like mobiles, laptops etc using Machine Learning approach. By doing sentiment analysis in a specific domain, it is possible to identify the effect of domain information in sentiment classification. We present a new feature vector for classifying the tweets as positive, negative and extract peoples' opinion about products. There are certain issues while dealing with identifying emotional keyword from tweets having multiple keywords. It is also difficult to handle misspellings and slang words .

2. Aliza Sarlan, Chayanit Nadam, Shuib Basri, ICIMU, November 2014. Twitter Sentiment Analysis. This paper reports on the design of a sentiment analysis, extracting a vast amount of tweets. Prototyping is used in this development. Results classify customers' perspective via tweets into positive and negative, which is represented in a pie chart and html page. However, the program has planned to develop on a web application system, but due to limitation of Django which can be worked on a Linux server or LAMP, for further this approach need to be done.

3. Onam Bharti and Mrs. Monika Malhotra, IJCSMC, June 2016.Sentimental Analysis on Twitter Data. In this paper, The goal of this report is to give an introduction to this fascinating problem and to present a framework which will perform sentiment analysis on online mobile phone reviews by associating modified K means algorithm with Naïve bayes classification and KNN. It is almost 91% accurate on implementation on some sample data and aims to gain 100% accuracy rate on any samples of data.

4. Ankur Goel, JyotiGautam, Satish Kumar, IEEE October 2016, Real Time Sentiment Analysis Using Naïve Bayes. This paper contains implementation of Naive Bayes using sentiment140 training data using twitter database and propose a method to improve classification. The above explained sentiment analysis model has a flaw that it takes a lot of time in fetching data from twitter and in data management.

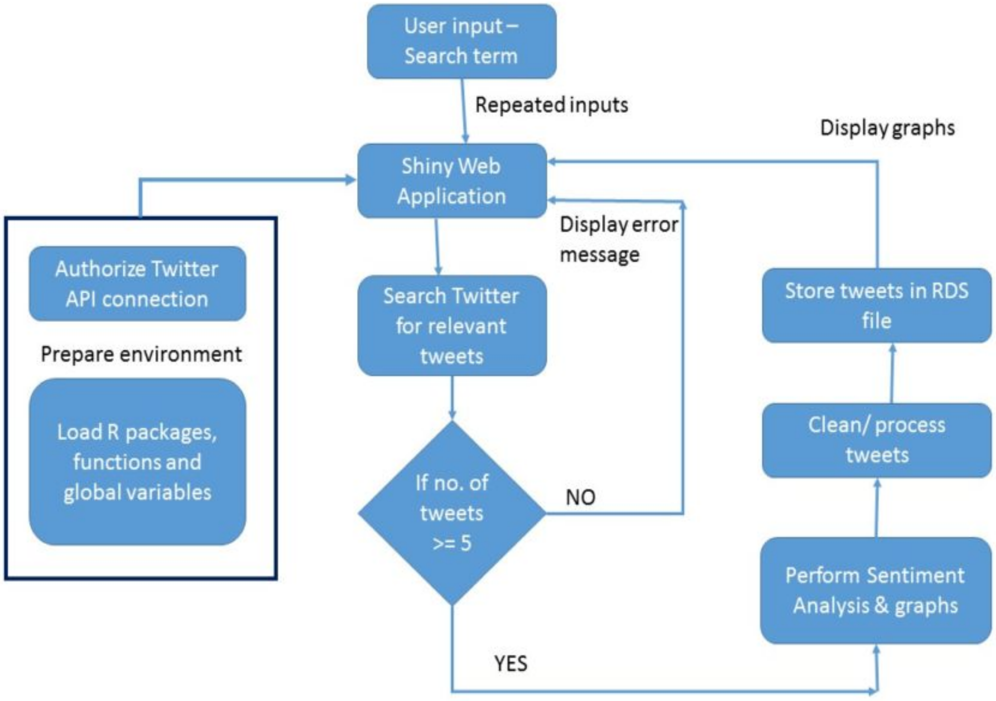
5. Abu Zonayed Riyadh, Nasif Alvi, Kamrul Hasan Talukder, ICCIT, December 2017, Exploring Human Emotion via Twitter. In this paper, focus is on emotion classification of tweets as multi-class classification. We have chosen basic human emotions

(happiness, sadness, surprise, disgust) and neutral as our emotion classes .Comparing with the related projects on social blogging sites by researchers, there is not much difference.

6. Rasika Wagh, Payal Punde, IEEE2018, Survey on Sentiment Analysis Using Twitter Dataset. This paper shows sentiment analysis types and techniques used to perform extraction of sentiment from twets and sees it as an area of text data mining and NLP. The study of literature shows that, the accuracy is improved when semantic analysis WordNet is followed up by the machine learning techniques, like SVM, Naïve-Bayes and maximum entropy.

7. Sahar A.El_Rahman,Feddah Alhumaidi AlOtaibi,Wejdan Abdullah AlShehri, IEEE, 2019. Sentiment Analysis of Twitter Data. The aim of this paper is to present a model that can perform sentiment analysis of real data collected from Twitter. Data in Twitter is highly unstructured which makes it difficult to analyze and data need to be classified using supervised models.

5. FLOWCHART



7. ADVANTAGES AND DISADVANTAGES OF SENTIMENT ANALYSIS:

Advantages:

● The use of this information can be applied to make wiser decisions related to the use of resources, to make improvements in organizations.

● Tracking people’s feelings on products, services and events, which allow enterprise managers to have knowledge and parameters to decision-making.

Disadvantages:

● For they are usually coupled with hashtags, emoticons and links, creating difficulties in determining the expressed sentiment.

8.APPLICATION :

● Social media monitoring

● People analytics and voice of employees

● Voice of customer & Customer Experience Management

● Regulatory Compliance

9. CONCLUSION

Sentiment analysis is a field of study for analyzing opinions expressed in text in several social media sites. Our proposed model used several algorithms to enhance the accuracy of classifying tweets as positive, negative and neutral. Our presented methodology combined the use of unsupervised machine learning algorithm where previously labeled data were not exist at first using lexicon-based algorithm. After that data were fed into several supervised model. For testing various metrics used, and it is shown that based on cross validation, maximum entropy has the highest accuracy. As a result, McDonalds is more popular than KFC in terms of both
negative and positive reviews. Same methodology can be used in various fields, detecting rumors on Twitter regarding the spread of diseases. For future work, an algorithm that can automatically classify tweets would be an interesting area of research

11. BIBILOGRAPHY

[1] Neethu M S, Rajasree R, “Sentiment Analysis in Twitter Using Machine Learning Techniques.” Institute of Electrical and Electronics Engineers (IEEE), July 2013.

[2] Aliza Sarlan, Chayanit Nadam, Shuib Basri, “Twitter Sentiment Analysis.” ICIMU, November 2014.
[3] Onam Bharti and Mrs. Monika Malhotra, “Sentimental Analysis on Twitter Data.” IJCSMC, June 2016.
[4] Ankur Goel, JyotiGautam, Satish Kumar, “Real Time Sentiment Analysis of Tweets Using Naïve Bayes” IEEE October 2016.
[5] Abu Zonayed Riyadh, Nasif Alvi, Kamrul Hasan Talukder, “Exploring Human Emotion via Twitter.” ICCIT, December 2017.
[6] Rasika Wagh, Payal Punde, “Survey on Sentiment Analysis Using Twitter Dataset”. IEEE 2018.
[7] Sahar A.El_Rahman,Feddah Alhumaidi AlOtaibi,Wejdan Abdullah AlShehri, “Sentiment Analysis of Twitter Data”. IEEE, 2019.

Source Code

```
#!/usr/bin/env python
# coding: utf-8

# # Twitter Sentiment Analysis

# In[1]:


get_ipython().system('pip install gensim --upgrade')
get_ipython().system('pip install keras --upgrade')
get_ipython().system('pip install pandas --upgrade')


# In[11]:


```

```python
get_ipython().system('pip install tensorflow --upgrade')


# In[1]:


# DataFrame
import pandas as pd

# Matplot
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

# Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.manifold import TSNE
from sklearn.feature_extraction.text import TfidfVectorizer

# Keras
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout, Embedding, Flatten, Conv1D, MaxPooling1D, LSTM
from keras import utils
from keras.callbacks import ReduceLROnPlateau, EarlyStopping

# nltk
import nltk
from nltk.corpus import stopwords
from  nltk.stem import SnowballStemmer

# Word2vec
import gensim

# Utility
import re
import numpy as np
import os
from collections import Counter
import logging
import time
import pickle
import itertools

# Set log
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)


# In[2]:


import tensorflow as tf
tf.__version__


# In[3]:


nltk.download('stopwords')


# ### Settings
```

```python

# In[23]:


# DATASET
DATASET_COLUMNS = ["target", "ids", "date", "flag", "user", "text"]
DATASET_ENCODING = "ISO-8859-1"
TRAIN_SIZE = 0.8

# TEXT CLENAING
TEXT_CLEANING_RE = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"

# WORD2VEC
W2V_SIZE = 300
W2V_WINDOW = 7
W2V_EPOCH = 32
W2V_MIN_COUNT = 10

# KERAS
SEQUENCE_LENGTH = 100
EPOCHS = 8
BATCH_SIZE = 1024

# SENTIMENT
POSITIVE = "POSITIVE"
NEGATIVE = "NEGATIVE"
NEUTRAL = "NEUTRAL"
SENTIMENT_THRESHOLDS = (0.4, 0.7)

# EXPORT
KERAS_MODEL = "model.h5"
WORD2VEC_MODEL = "model.w2v"
TOKENIZER_MODEL = "tokenizer.pkl"
```

```
ENCODER_MODEL = "encoder.pkl"


# ### Read Dataset

# ### Dataset details
# * **target**: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
# * **ids**: The id of the tweet ( 2087)
# * **date**: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
# * **flag**: The query (lyx). If there is no query, then this value is NO_QUERY.
# * **user**: the user that tweeted (robotickilldozr)
# * **text**: the text of the tweet (Lyx is cool)

# In[5]:


#dataset_filename = os.listdir("../input")[0]
#dataset_path = os.path.join("..","input",dataset_filename)
#print("Open file:", dataset_path)
df = pd.read_csv('trained.csv', encoding =DATASET_ENCODING, names=DATASET_COLUMNS)


# In[6]:


print("Dataset size:", len(df))


# In[7]:
```

```


df.head(5)


# ### Map target label to String
# * **0** -> **NEGATIVE**
# * **2** -> **NEUTRAL**
# * **4** -> **POSITIVE**

# In[8]:


decode_map = {0: "NEGATIVE", 2: "NEUTRAL", 4: "POSITIVE"}
def decode_sentiment(label):
    return decode_map[int(label)]


# In[9]:


get_ipython().run_cell_magic('time', '', 'df.target = df.target.apply(lambda x: decode_sentiment(x))')


# In[10]:


target_cnt = Counter(df.target)

plt.figure(figsize=(16,8))
plt.bar(target_cnt.keys(), target_cnt.values())
plt.title("Dataset labels distribuition")

```

```python

# ### Pre-Process dataset

# In[11]:


stop_words = stopwords.words("english")
stemmer = SnowballStemmer("english")


# In[12]:


def preprocess(text, stem=False):
    # Remove link,user and special characters
    text = re.sub(TEXT_CLEANING_RE, ' ', str(text).lower()).strip()
    tokens = []
    for token in text.split():
        if token not in stop_words:
            if stem:
                tokens.append(stemmer.stem(token))
            else:
                tokens.append(token)
    return " ".join(tokens)


# In[13]:


get_ipython().run_cell_magic('time', '', 'df.text = df.text.apply(lambda x: preprocess(x))')


```

```
# ### Split train and test

# In[14]:


df_train, df_test = train_test_split(df, test_size=1-TRAIN_SIZE, random_state=42)
print("TRAIN size:", len(df_train))
print("TEST size:", len(df_test))


# ### Word2Vec

# In[15]:


get_ipython().run_cell_magic('time', '', 'documents = [_text.split() for _text in df_train.text] ')


# In[16]:


w2v_model = gensim.models.word2vec.Word2Vec(size=W2V_SIZE,
                                            window=W2V_WINDOW,
                                            min_count=W2V_MIN_COUNT,
                                            workers=8)


# In[17]:

```

```

w2v_model.build_vocab(documents)


# In[18]:


words = w2v_model.wv.vocab.keys()
vocab_size = len(words)
print("Vocab size", vocab_size)


# In[19]:


get_ipython().run_cell_magic('time', '', 'w2v_model.train(documents, total_examples=len(documents), epochs=W2V_EPOCH)')


# In[20]:


w2v_model.most_similar("love")


# ### Tokenize Text

# In[21]:


get_ipython().run_cell_magic('time', '', 'tokenizer = Tokenizer()\ntokenizer.fit_on_texts(df_train.text)\n\nvocab_size = len(tokenizer.word_index) + 1\nprint("Total words",
vocab_size)')


# In[24]:


get_ipython().run_cell_magic('time', '', 'x_train = pad_sequences(tokenizer.texts_to_sequences(df_train.text), maxlen=SEQUENCE_LENGTH)\nx_test = pad_sequences(tokenizer.texts_to_sequences(df_test.text), maxlen=SEQUENCE_LENGTH)')


# ### Label Encoder

# In[25]:


labels = df_train.target.unique().tolist()
labels.append(NEUTRAL)
labels


# In[26]:


encoder = LabelEncoder()
encoder.fit(df_train.target.tolist())

y_train = encoder.transform(df_train.target.tolist())
y_test = encoder.transform(df_test.target.tolist())

y_train = y_train.reshape(-1,1)
y_test = y_test.reshape(-1,1)

```

```
print("y_train",y_train.shape)
print("y_test",y_test.shape)


# In[27]:


print("x_train", x_train.shape)
print("y_train", y_train.shape)
print()
print("x_test", x_test.shape)
print("y_test", y_test.shape)


# In[28]:


y_train[:10]


# ### Embedding layer

# In[29]:


embedding_matrix = np.zeros((vocab_size, W2V_SIZE))
for word, i in tokenizer.word_index.items():
  if word in w2v_model.wv:
    embedding_matrix[i] = w2v_model.wv[word]
print(embedding_matrix.shape)


# In[30]:

```

```

embedding_layer = Embedding(vocab_size, W2V_SIZE, weights=[embedding_matrix], input_length=SEQUENCE_LENGTH, trainable=False)


# ### Build Model

# In[31]:


model = Sequential()
model.add(embedding_layer)
model.add(Dropout(0.5))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.summary()


# ### Compile model

# In[32]:


model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])


# ### Callbacks

# In[33]:

```

```

callbacks = [ ReduceLROnPlateau(monitor='val_loss', patience=5, cooldown=0),
              EarlyStopping(monitor='val_acc', min_delta=1e-4, patience=5)]


# ### Train

# In[34]:


get_ipython().run_cell_magic('time', '', 'history = model.fit(x_train, y_train,\n batch_size=BATCH_SIZE,\n epochs=EPOCHS,\n validation_split=0.1,\n verbose=1,\n callbacks=callbacks)')


# ### Evaluate

# In[35]:


get_ipython().run_cell_magic('time', '', 'score = model.evaluate(x_test, y_test, batch_size=BATCH_SIZE)\nprint()\nprint("ACCURACY:",score[1])\nprint("LOSS:",score[0])')


# In[39]:


acc = history.history['accuracy']
```

```python
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))
plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()


# ### Predict

# In[40]:


def decode_sentiment(score, include_neutral=True):
    if include_neutral:
        label = NEUTRAL
        if score <= SENTIMENT_THRESHOLDS[0]:
            label = NEGATIVE
        elif score >= SENTIMENT_THRESHOLDS[1]:
            label = POSITIVE

        return label
    else:
        return NEGATIVE if score < 0.5 else POSITIVE


```

```
# In[41]:


def predict(text, include_neutral=True):
    start_at = time.time()
    # Tokenize text
    x_test = pad_sequences(tokenizer.texts_to_sequences([text]), maxlen=SEQUENCE_LENGTH)
    # Predict
    score = model.predict([x_test])[0]
    # Decode sentiment
    label = decode_sentiment(score, include_neutral=include_neutral)

    return {"label": label, "score": float(score),
       "elapsed_time": time.time()-start_at}


# In[42]:


predict("I love the music")


# In[43]:


predict("I hate the rain")


# In[44]:


```

```
predict("i don't know what i'm doing")


# ### Confusion Matrix

# In[45]:


get_ipython().run_cell_magic('time', '', 'y_pred_1d = []\ny_test_1d = list(df_test.target)\nscores = model.predict(x_test, verbose=1, batch_size=8000)\ny_pred_1d = [decode_sentiment(score, include_neutral=False) for score in scores]')


# In[46]:


def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
```

```python
    plt.xticks(tick_marks, classes, rotation=90, fontsize=22)
    plt.yticks(tick_marks, classes, fontsize=22)

    fmt = '.2f'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label', fontsize=25)
    plt.xlabel('Predicted label', fontsize=25)


# In[47]:


get_ipython().run_cell_magic('time', '', '\ncnf_matrix = confusion_matrix(y_test_1d, y_pred_1d)\nplt.figure(figsize=(12,12))\nplot_confusion_matrix(cnf_matrix, classes=df_train.target.unique(), title="Confusion matrix")\nplt.show()')


# ### Classification Report

# In[48]:


print(classification_report(y_test_1d, y_pred_1d))

```

```

# ### Accuracy Score

# In[49]:


accuracy_score(y_test_1d, y_pred_1d)


# ### Save model

# In[50]:


model.save(KERAS_MODEL)
w2v_model.save(WORD2VEC_MODEL)
pickle.dump(tokenizer, open(TOKENIZER_MODEL, "wb"), protocol=0)
pickle.dump(encoder, open(ENCODER_MODEL, "wb"), protocol=0)


# In[ ]:

```