TWITTER SENTIMENT ANALYSIS USING MACHINE LEATNING ALGORITHMNS IN PYTHON

A PROJECT REPORT

Submitted by

HASHIM HAYATH BASHA -- 210420243024
MAURUS MARIA RUBENSON A -- 210420243036
KISHORE VS -- 210420243032

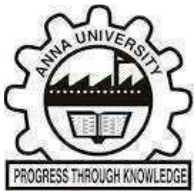
in partial fulfillment for the award of the degree of

B. TECH

IN

ARTIFICIAL INTILLIGENCE
&
DATA SCIENCE

CHENNAI INSTITUTE OF TECHNOLOGY
CHENNAI 600 069



ANNA UNIVERSITY:: CHENNAI 600 025

November & 2022

BONAFIDE CERTIFICATE

Certified that this project report "**Twitter Sentiment Analysis using Machine Learning Algorithms in python**" is the Bonafide work of "**HASHIM HAYATH BASHA-(210420243024),MAURUS MARIA RUBENSON-(210420243036), KISHORE VS- (210420243032)**" who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

(SIGNATURE)

Dr. .T. PREETHIYA,Ph.D

HEAD OF THE DEPARTMENT

Associate Professor

Dept. of Computer Science and Engineering

Chennai Institute of Technology

Kundrathur-600069

(SIGNATURE)

Dr. S.K MUTHUSUNDAR, Ph.D

SUPERVISOR

Associate Professor

Dept. of Computer Science and Engineering

Chennai Institute of Technology

Kundrathur -600069

Submitted for the viva-voice held on _____ at Chennai Institute of Technology, Kundrathur.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMNET

I thank the almighty, for the blessings that have been showered upon me to bring forth the success of the project would like to express my sincere gratitude to our **chairman Shri.P.SRIRAM**, and all trust members of Chennai Institute of Technology for providing the facility and opportunity to do this project as a part of our undergraduate course.

We thank our Principal **Dr.A.RAMESH M.E.. Ph.D,** for his valuable suggestion and guidance for the development and completion of this project.

We sincerely thank our Head pf the Department **Dr. T.PREETHIYA, Ph.D,** Head of the Department of Chennai institute of technology, for having provided us valuable guidance, resources and timely suggestions through our work.

We sincerely thank our project guide **Dr. S.K MUTHUSUNDAR, Ph.D,** our beloved project guide for having provided us valuable guidance, resources and timely suggestions through our work.
.

I express our deep sense of thanks to all faculty members in my department for their cooperation and interest shown at every stage of our endeavor in making a project work success.

Last but not least we extend our deep gratitude to our beloved family members for their moral coordination, encouragement and financial support to carry out this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

S. NO	ABBREVIATIONS	EXPANSION
1	ML	MACHINE LEARNING
2	CNN	CONVOLUTIONAL NEURAL NETWORK
3	WORD2VEC	WORD TO VECTOR
4	CBOW	CONTINUOS BAG OF WORDS
5	SVM	SUPPORT VECTOR MACHINE
6	LSTM	LONG SHORT-TERM MEMORY
7	RNN	RECURRENT NEURAL NETWORK

LIST OF TABLES

ABSTRACT

Twitter is a platform used by people to express their opinions and display sentiments on different occasions. Twitter is one of the non-traditional data sources with unlimited potential. It contains a large reserve of data sets which are easier to access and collect when compared to others. One of the available tools is twitter API which let us to collect data and various other information about the tweets. The tweets are collected through Tweepy and the tweets are being labelled through Text-Blob. To increase the processing of the tweets are done. The accuracy we will create an vocab using W2R (word to vector) model containing all the related words. The words are vectorized with the relationship among them. This makes the prediction easier. Then through implementation of CNN (conventional neural network) we insert the embedding layer which is formed with vectorized words and a LSTM (long-short-term-memory) layer. The accuracy value of the model is created and we can analyze the tweets of the particular domain.

INTRODUCTION

Twitter is a free social networking site where users broadcast short posts known **as** tweets. Twitter is a popular microblogging service where users create status messages These tweets sometimes express opinions about different topics. These tweets can contain text, videos, photos or links. Twitter is a service for friends, family, and coworkers to communicate and stay connected through the exchange of quick, frequent messages. People post Tweets, which may contain photos, videos, links, and text.

In order to extract sentiment from tweets, sentiment analysis is used. The results from this can be used in many areas like analyzing and monitoring changes of sentiment with an event, sentiments regarding a particular brand or release of a particular product, analyzing public view of government policies etc. Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. Also known as opinion mining, MEANS deriving the opinion.

It is a process of deriving sentiment of a particular statement or sentence. Sentiment analysis approaches can be broadly categorized in two classes:
lexicon based and machine learning based. Lexicon based approach is unsupervised as it proposes to perform analysis using lexicons and a scoring method to evaluate opinions.

Whereas machine learning approach involves use of feature extraction and training the model using feature set and some dataset Aim of twitter sentiment analysis is that it allows you to keep track of what's being said about your product or service on social media, and can help you detect angry customers or negative mentions before they they escalate. Performing sentiment analysis is challenging on Twitter data, as we mentioned earlier.

Here we define the reasons:

- Limited tweet size: only 140 characters in hand, compact statements are generated, which results sparse set of features.

- Use of slang: these are different from English words and it can make an approach outdated because of the evolutionary use of slangs.

- Twitter features: it allows the use of hashtags, user reference and URLs.

- User variety: the users express their opinions in a variety of ways, some using different language in between, while others using repeated words or symbols to convey an emotion

LITERATURE SURVEY

In CNN for situations understanding based on sentiment analysis of twitter data by Shi yang ,Liao a, Junbo Wangb, Ruiyun Yua, Koichi Satob, Zixue Cheng they proved CNN's ability to extract a set of features from a global dataset is the primary justification for using it in image analysis and classification. information, and it can take into account how these qualities relate to one another. The aforementioned method can increase the accuracy in classification and evaluation. Text data features can also be piecemeal collected for natural language processing in order to Take into account how these elements interact, but without taking into account context or the entire sentence, the sentiment be misunderstood.

We create a straightforward convolutional neural network model and evaluate it using the Results indicate that it performs more accurately in classifying Twitter sentiment than certain standard methods, including similar to the SVM and Naive Bayes techniques.

In A sentiment analysis study for twitter using the various model of convolutional neural network. Machine learning produces poor accuracy that is used in a larger range of applications. Consequently, the deep learning approach is being developed to increase sentiment analysis' accuracy. In order to increase accuracy performance, this study discusses several configuration options based on deep learning utilizing the Convolutional Neural Network (CNN) algorithm. To evaluate the effectiveness of the CNN models, a variety of parameters are offered, including the number of convolutional layers, the quantity of filters, and the size of the filters. The Word2Vec model for Indonesian has been utilized with the Indonesian-Sentiment-Analysis-Dataset, which consists of 10.806 tweets, as a word vector representation. The remaining 20% of the dataset is used for testing once the CNN models have been trained on 80% of it. Results from the proposed CNN models are compared and shown to be superior.

In Twitter sentiment analysis using distributed word and sentence representation by Dwarampudi Mahidhar Reddy, Dr. N V Subba Reddy and Dr. Prema K V they introduced usage of Long Short-Term Memory (LSTM) Networks and Convolutional Neural Networks (CNNs)for Distributed Representation of words. Which lets us to capture

Capturing local co-occurrence statistics and gives good performance with small (100-300) dimension vectors that are important for downstream tasks. It makes the process faster as only non-zero counts matter. Instead of utilising conventional techniques or preparing text data, this research uses distributed representations of words and phrases. While the latter is used for the distributed representation of sentences, the first two are utilized for the distributed representation of words. This document has an accuracy rate of up to 81%. Out of the various techniques, it also offers the best and most effective approaches to generate distributed.

In Using Word2Vec to Process Big Text Data by Long Ma and Yanqing Zhang they proposed a method to decrease the dimension of the feature vector which will be used to make our embedding layer for the CNN model which we prepare. Processing large data sets takes time since they can have a variety of distinct data types and sophisticated structures in addition to their large volume of data. If the learning algorithm can choose useful features or reduce the feature dimension, it will be more effective when taking the data dimension into account. Continuous Bag of Words (CBOW) and Skip-gram are the two learning models that make up Word2Vec. Word2Vec creates word vectors from text data that can be represented as a substantial passage of text or perhaps the complete article. In our work, we trained the data using a Word2Vec model and then assessed the degree of word similarity.

In addition, we clustering the similar words together and use the generated clusters to fit into a new data dimension so that the data dimension is decreased which will decrease time consumption and increase performance.

EXISTING SYSTEM

The present work on sentiment analysis can be categorized from a variety of angles, including the method employed, the perspective on the text, the amount of text analysis detail, the rating level, etc. We identified machine learning, lexicon-based, statistical, and rule-based techniques from a technological perspective.

By training on a known dataset, the machine learning method employs different learning algorithms to ascertain the sentiment. The lexicon-based method involves determining a review's emotion polarity based on the semantic orientation of its words or sentences. A text's subjectivity and viewpoint are measured by its "semantic orientation."

The rule-based technique scans a document for opinion words before classifying it according to the proportion of positive and negative words. It takes into account a variety of classification criteria, including dictionary polarity, negation and boosting words, idioms, emoticons, and mixed viewpoints, among others.

Each review is represented by statistical models as a combination of latent features and ratings. In order to cluster head words into aspects and sentiments into ratings,
it is believed that aspects and their ratings can be represented by multinomial distributions. Another categorization is focused primarily on the organization of the text classification.

At the document, phrase, or word level. Sentence- or word-level classification can express a sentiment polarity for each sentence in a review or even for each word, as
opposed to document-level classification, which seeks to identify a sentiment polarity for the entire review.

According to our research, the majority of approaches concentrate on the document-level. Additionally, we may distinguish between techniques that aim to score a review globally versus techniques that gauge the strength of sentiment for various parts of a product.
Most approaches to global review classification that depend on machine learning simply take into account the polarity of the review (positive/negative) into account. More linguistic variables, such as intensification, negation, modality, and discourse structure, are used in solutions that aim for a more precise classification of reviews
(Such as three- or five-star ratings).

A thorough classification of existing approaches is shown in Figure 1. This grouping is not exhaustive. One solution can be used in multiple categories.

1. Rule or Lexion based approach - (how it works):
   It counts number of positive or negative values in denote their polarity and sentiment to calculate the score.

- (Disadvantages):
  - It does not care about combination of words
  - quick but need constant maintenance

2. Automated or Machine Learning approach
   - Traditional Models:: gathering of a dataset with examples for positive, negative, and neutral classes, processing data finally training the algorithm

Deep Learning Models::
- Naive Bayes sentiment analysis
- Deep Learning

Sentiment analysis using NLP deep learning are able to learn patterns through multiple layers from unstructured and unlabeled data to perform sentiment analysis.

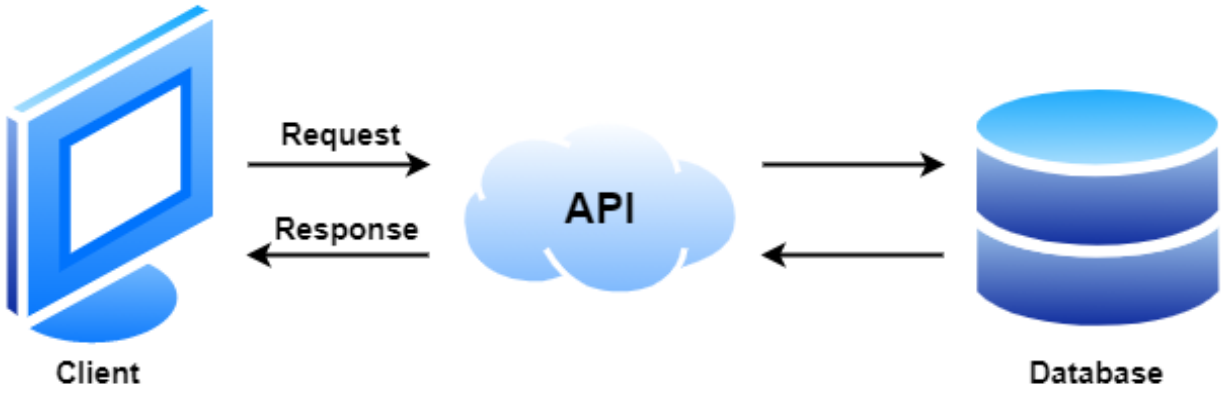
3. Hybrid approach

4. Named Entity based Sentiment Analyzer

Sentiment Analysis based on top named entities Targeted by finding sentences containing the named entities and performing sentiment analysis only on those sentences one by one.

PROPOSED SYSTEM

Here we will implement our system from the data collection process itself. We will be using twitter API for data collection for sentiment analysis of twitter data. Major obstacle in twitter analysis is domain dependence and world knowledge. To overcome these, we will collect data directly from the topics where the tweets are tested.



We will be collecting 1.6 M data from the twitter API. These datasets are divided into training datasets and testing datasets. Using the Online Text Blob library, we will append the texts along with their respective polarity of the texts. Before we Use the Text blob we will convert the emojis in the text and convert them into the appropriate meanings. This is one of the reasons why twitter is considered one with the most potential to develop the Natural Language processing where we can include other information present in the tweets.

The Preprocessing of data does not end here as they are various steps involved in making the data adaptable to our model that is being created. The first step always practiced in NLP is Removal of Stop words which are commonly used words to make sure phrase structure does not get affected. Then Tokenization is done where the text is splitted to array of words. Then Stemming is done to reduce each word in the array to reduce it to their respective root words. After the preprocess is done the dataset is divided into testing dataset and training dataset.

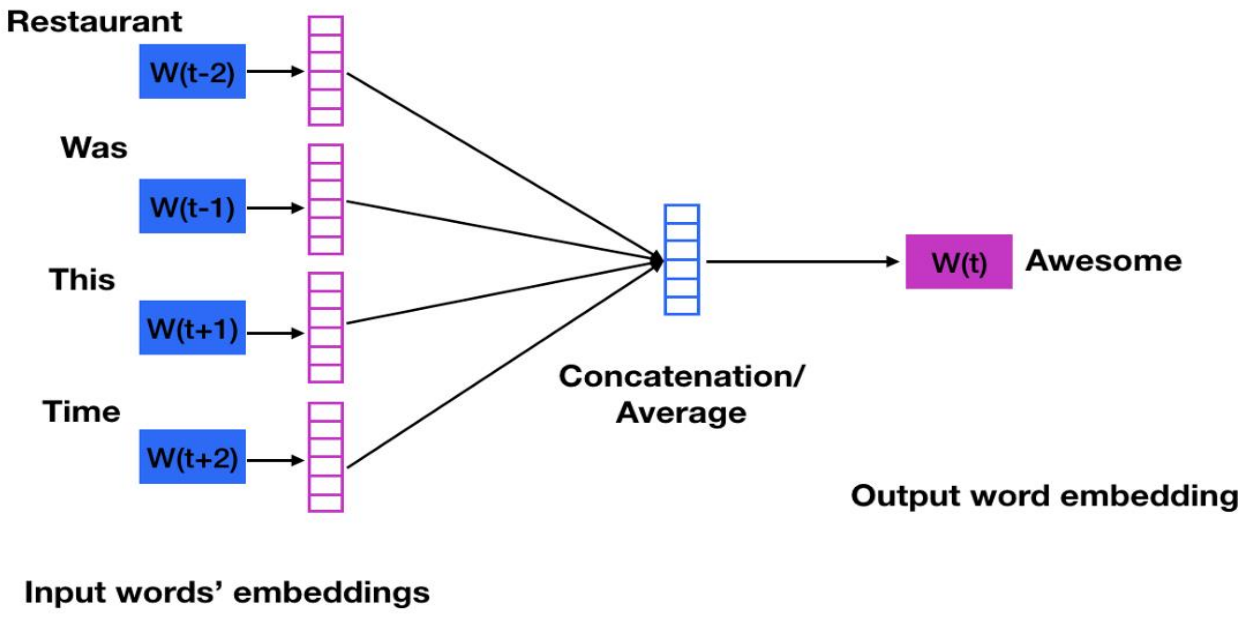
Word2Vec Model

Word Embeddings

Word embeddings are words that have been translated into real number vectors in a way that preserves their semantic meaning. The techniques used in my earlier entries of BOW and TFIDF failed to capture the meaning between the words; instead, they treated each word separately as a feature. This model considers the words around a given word inside a specific window size when training it. The term "embedding vectors" can be derived in a variety of ways. One such approach is Word2vec, which makes use of a neural embeddings model to learn that.

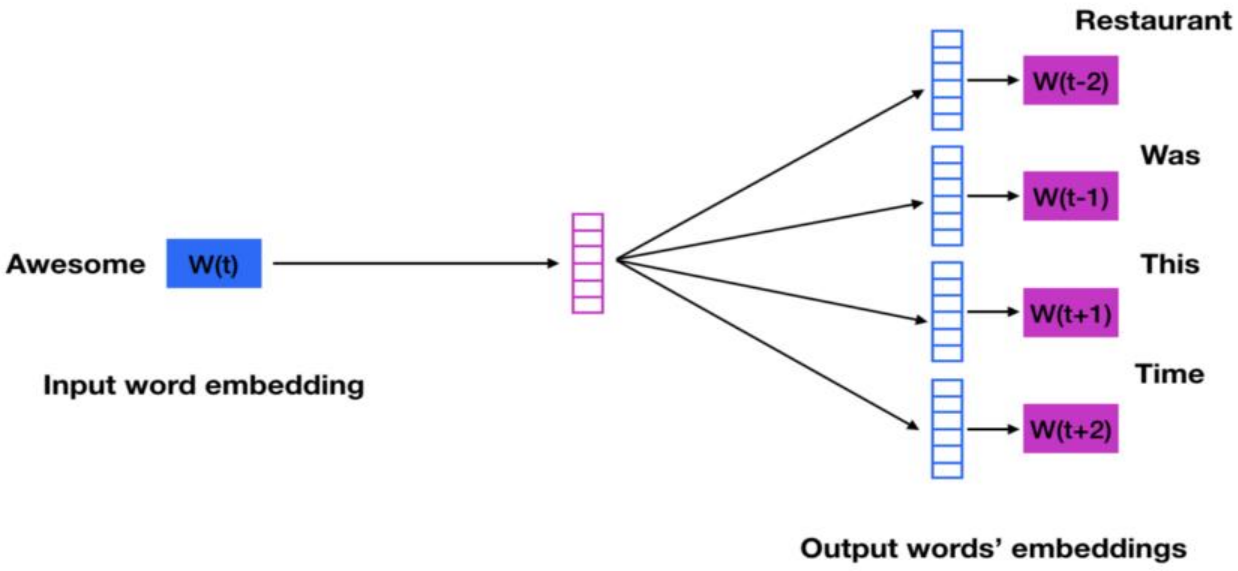
CBOW

Here the model predicts the word under consideration given context words given within a specific window. The hidden layer has the number of dimensions in which the current word needs to be represented at the output layer.



Skip Gram

Skip gram is opposite of CBOW where it predicts embeddings for the surrounding context words in the specific window given a current word. The input layer contains the current word and the output layer contains the context words.



Word2Vec Vectors

Word2Vec vectors are generated for each iteration by traversing through the training dataset. By simply using the model on each word in the training dataset we will be able to obtain the word embedding for those words in tweets. We will implement the average over the vectors of words in the sentence that will represent the sentence.

Convolutional Neural Network

Here after creating the embedding layer using the word2vec vectors, it is added as one of the layers in the CNN then a Dropout layer will be added with a value of 0.5. The dropout layer is added which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 – rate) such that the sum over all inputs is unchanged. Finally, an LSTM layer is being introduced along with a dense layer. The dense layer used to reduce the dimensionality of the output. In the end of the CNN modeling along with the Word2Vec vectors word embeddings we get the polarity or the sentiment of the tweets.

Model: "sequential"

Layer (type)	Output Shape	Param #

embedding (Embedding)	(None, 300, 300)	12659700
dropout (Dropout)	(None, 300, 300)	0
lstm (LSTM)	(None, 100)	160400
dense (Dense)	(None, 1)	101

Total params:12,820,201
Trainable params: 160,501
Non-trainable params: 12,659,700

This classification and label as Proposed System. This classification and labeling of tweets make it more polarized towards the specific domain and the respective world knowledge through our suggestion and the way of preparation of our data.

APPLICATIONS

- The application of sentiment analysis ranges from business to marketing, politics, health to public. Sentiment analysis is also not just limited to one application only, but it also provides a vast application in various different areas to assist in the decision making.

- Twitter Sentiment analysis also allows raising awareness of data security and the danger of security breaches. It also acts as a guideline for the organization/companies to respond to security alerts in shaping public views.

- Sentiment analysis also creates advantage for business owners to identify their popularity among customer and how they think about their product or service and assessing the effectiveness and capability of business brand communication and social media to evaluate their business flow of stock price through the use of social media .

- Sentiment analysis on social media allows the organization to evaluate the success level of a program. As a recent study shows where a high positive sentiment is obtained from a tweet of a community development program activity.

- Other include :

    - Social Media Monitoring
    - Customer Service
    - Market Research
    - Brand Monitoring

SYSTEM SPECIFICATION

SOFTWARE REQUIREMENT

S.NO	SOFTWARE	VERSION	URL
1	ANACONDA	3.7	https://www.anaconda.com/
2	NUMPY	1.11.3	https://numpy.org/
3	KERAS	2.3.0	https://keras.io/
4	TENSOR FLOW	2.0	https://www.tensorflow.org/
5	MATPLOTLIB	3.5	https://matplotlib.org/
6	NLTK	3.9	https://www.nltk.org/
7	PANDAS	3.9	https://pandas.pydata.org/
8	TWEEPY	4.12.1	https://pypi.org/project/tweepy/
9	WORLDCLOUD	1.8.2.2	https://pypi.org/project/wordcloud/
10	GENSIM	4.2.0	https://pypi.org/project/gensim/

ANACONDA

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.). that aims to simplify package management and deployment. Package versions are managed by the package management system anaconda. The Anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.

TENSORFLOW

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library. and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

KERAS

Kera's is an open-source neural-network library written in ion li la capable of running on top of TensorFlow, Microsoft Cognitive Toolkit. It focuses on being user-friendly, modular, and extensible. Kera's contains numerous implementations of commonly used neural-network building blocks such layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

MATPLOTLIB

Matplotlib is a comprehensive library for creating static, animated. They are interactive visualizations in python.

PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make the working with "relational" or "labeled" data both easy and intuitive. the name is derived from the term panel data, that is an econometrics term for multidimensional structured data sets.

TWEEPY

Tweepy is an open-source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as: Data encoding and decoding.

WORLDCLOUD

Wordcloud is basically a visualization technique to represent the frequency of words in a text where the size of the word represents its frequency. In order to work with wordclouds in python, we will first have to install a few libraries using pip.

GENSIM

Gensim = “Generate Similar” is a popular open-source natural language processing (NLP) library used for unsupervised topic modeling. It uses top academic models and modern statistical machine learning to perform various complex tasks such as − Building document or word vectors.

HARDWARE REQUIREMENT

PROCESSOR	INTEL CORE
OPERATING SYSTEM	WINDOWS 11(64-BIT)
RAM	8 GB
HARD DISK	40 GB
MONITOR	15 VGA COLOUR
MOUSE	LOGITECH
GRAPHICS	DIRECTX 9 WITH WDDM 1.0

DATASET DESCRIPTION

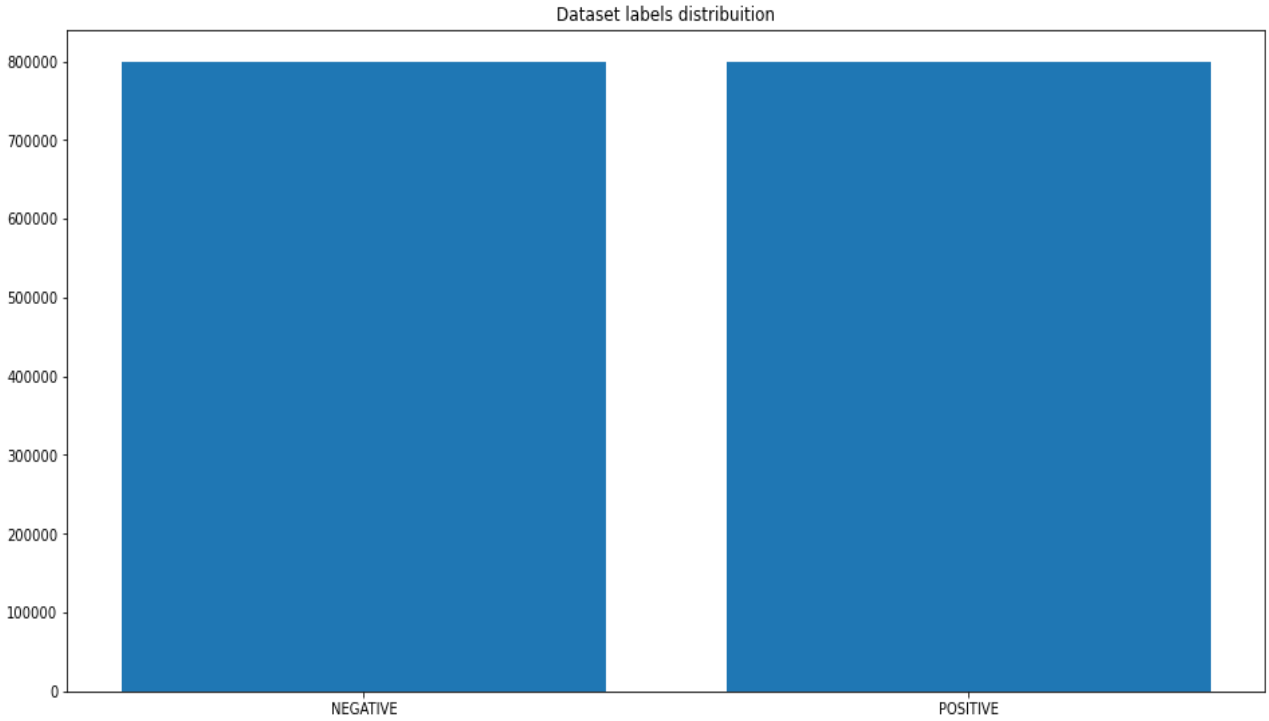
Here we collect the data through twitter api called tweepy. This API has a limitation of how much data can be mined through it for 10 min. Standard tweepy requests only allow 180 tweets per 10 min. This process is both time consuming and resource intensive. We can use a cloud service to run our program for the required hours and get our data through the selection process.

After the end of collection of data, we can use the demote package to convert emoji and emoticons to their respective text version.

The collected json files are used to collect the entities as target which is manually labeled using Textblob, ids of the user, date of the tweets posted, flag, user name, the tweet they posted.

	target	ids	date	flag	user	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 data mining is done through proper selection. In tweepy we can mention various parameters which helps us to select a particular tweet. These tweets make our model and help us to overcome the limitations and challenges in the present in the present model. Our new improvements and increase of accuracy for a particular domain and involvement of world knowledge is done through the data mining process itself.


Dataset labels distribuition
800000
700000
600000
500000
400000
300000
200000
100000
0
NEGATIVE
POSITIVE

OBJECTIVE

To put into practice an algorithm that automatically categorizes material as positive, negative, or neutral.

Sentiment analysis is used to evaluate whether the general public has a good, negative, or neutral view toward the topic at hand.

Native Models does not include world knowledge and domain specific considerations here in this model we use tweets of a specific category so that we can make more accurate predictions of sentiments.

Twitter sentiment analysis allows us to keep a track of what's being said about your product/service on social media, and can help to detect angry customers or negative mentions before they escalate.

USE CASES

Twitter sentiment analysis provides many exciting opportunities. Being able to analyze tweets in real time, and determine the sentiment that underlies each message, adds a new dimension to social media monitoring.

Twitter sentiment analysis can also help us to use the data to optimize the future Twitter campaigns and yield us better results.

CHALLENGES FOR SENTIMENT ANALYSIS

- Implicit Sentiment and Sarcasm
- Domain Dependency
- Thwarted Expectations
- World Knowledge and Pragmatics
- Subjectivity Detection
- Entity Identification
- Negation

FUTURE WORK

In the present work we provided an implementation of sentiment analysis using distributed presentation of words and sentences and fit them into the Convolutional Neural Network through gensim Word2Vec for Sentiment Analysis based on tweets. There are many research done such as considering the Recursive Neural Tensor Network, Attention-Based Neural Networks, Multi-Task Learning Based Models, Unsupervised Sentiment Neuron and Non-Neural Networks Based Models these models are still in research and their strong points and advantage of using these particular models in specific situations are clearly defined.

With use of twitter as a Database we can explore various clusters of data and we can do proper collection of data and remove the other domain texts at the collection process itself. This process can be improved with having more advanced access to data in the twitter developer options. The speed of collection may help us build models of each domain separately and using hierarchical methods we can classify the polarities of tweets of specific domains.

CODE

```
from cgi import test
import tweepy
import configparser
import pandas as pd
import numpy as np
from emot.emo_unicode import UNICODE_EMOJI
import re
from nltk.tokenize import wordpunct_tokenize


#read configs
config = configparser.ConfigParser()
config.read('config.ini')

api_key = config['twitter']['api_key']
api_key_secret = config['twitter']['api_key_secret']

access_token = config['twitter']['access_token']
access_token_secret = config['twitter']['access_token_secret']

#authentication
auth = tweepy.OAuth1UserHandler(api_key,api_key_secret)
auth.set_access_token(access_token,access_token_secret)

api = tweepy.API(auth)

 #q = success also gives us the tweets to be searched

 n=1600000

 tweets=tweepy.Cursor(api.search_tweets,q="peace -filter:links -
 filter:retweets",lang="en",tweet_mode="extended").items(n)
```

```
tweets_json=[]

for tweet in tweets:
  tweets_json.append(tweet._json)

x = len(tweets_json)

# To save the data as collected from twitter

import json

for i in tweets_json:
   with open("basic_data.json", "w") as outfile:
      json.dump(i, outfile)

df = pd.DataFrame(columns = ['target','ids','date','flag','user','text'])

def convert_emojis(text):
   for emot in UNICODE_EMOJI:
      text = text.replace(emot," "+UNICODE_EMOJI[emot].replace(",","").replace(":","")+" ")
      text = text.replace("_"," ")
   return text


artext = []
user_id = []
timing = []
user_name = []

for i in range(x):
   basic = tweets_json[i]["full_text"]
   basic = convert_emojis(basic)
   artext.append(basic)
```

```
    user_id.append(tweets_json[i]["id"])
 user_name.append(tweets_json[i]["user"]["name"])
 timing.append(tweets_json[i]["created_at"][:-
 10]+"PDT"+tweets_json[i]["created_at"][-5:])


 artext = pd.Series(artext)
 timing = pd.Series(timing)
 user_name = pd.Series(user_name)
 user_id = pd.Series(user_id)


 df["text"] = artext
 df["ids"] = user_id
 df["date"] = timing
 df["user"] = user_name
 df["flag"] = "NO_QUERY"


test1=df.copy()

test1=test1.apply(lambda x:x.astype(str).str.lower())

#replace all the non-alphabetic elements with 1 and keep
# [^A-za-z.!+ ] keep all the letter, dot and exclamation mark because a lot of word
 attached to ! or . as
# form of expression
test1["text"]=test1["text"].apply(lambda x:  re.sub("[^A-za-z.!+ ]", '1', x))

test1["text"]=test1["text"].apply(lambda x: " ".join(w or w in
 wordpunct_tokenize(x.strip()) if (w.isalpha())))

x = len(test1["text"])
```

```
#TextBlob api
from textblob import TextBlob

sent = []

for i in range(x):
    blob = TextBlob(test1["text"][i])
    ss = 0
    den = 0
    for sentence in blob.sentences:
        ss += sentence.sentiment.polarity
        den += 1
    if den==0:
        sent.append(2)
    else:
        sco = ss/den
        if sco==0:
            sent.append(2)
        elif sco>0:
            sent.append(4)
        else:
            sent.append(0)

sent = pd.Series(sent,dtype="string")

test1["target"] = sent

print(test1)

test1.to_csv('testntrain.csv')
```

```
!pip install keras
pip install tensorflow
pip install gensim
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

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

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

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

import gensim
import re
import numpy as np
import os
from collections import Counter
import logging
import time
import pickle
import itertools
```

```
df =
    pd.read_csv("testntrain.csv",names=["target","ids","date","flag","user","text"])
print("Dataset size:", len(df))
df.head(5)
decode_map = {0: "NEGATIVE", 2: "NEUTRAL", 4: "POSITIVE"}
def decode_sentiment(label):
   return decode_map[int(label)]
%%time
df.target = df.target.apply(lambda x: decode_sentiment(x))
target_cnt = Counter(df.target)

plt.figure(figsize=(16,8))
plt.bar(target_cnt.keys(), target_cnt.values())
plt.title("Dataset labels distribuition")
nltk.download('stopwords')
stop_words = stopwords.words("english")
stemmer = SnowballStemmer("english")
def create_wordcloud(text):
   words=' '.join([words for words in text])
   wordcloud = WordCloud(max_font_size=50, max_words=200,
    background_color="white").generate(words)
   plt.figure(figsize=(12, 10))
   plt.imshow(wordcloud, interpolation="bilinear")
   plt.axis('off')
   plt.show()
from wordcloud import WordCloud
create_wordcloud(stop_words)

# 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 = 300
EPOCHS = 6
BATCH_SIZE = 100

# 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"
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)
```

```
%%time
df.text = df.text.apply(lambda x: preprocess(x))

df_train, df_test = train_test_split(df, test_size=1-TRAIN_SIZE, random_state=42)

df_train = df_train[0:60000]
df_test = df_test[0:10000]

%%time
documents = [_text.split() for _text in df_train.text]

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


w2v_model.build_vocab(documents)

words = list(w2v_model.wv.index_to_key)
vocab_size = len(words)
print("Vocab size", vocab_size)

%%time
w2v_model.train(documents, total_examples=len(documents),
epochs=W2V_EPOCH)

w2v_model.wv.most_similar("love")


%%time
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df_train.text)
```

```
vocab_size = len(tokenizer.word_index) + 1
print("Total words", vocab_size)

%%time
x_train = pad_sequences(tokenizer.texts_to_sequences(df_train.text),
maxlen=SEQUENCE_LENGTH)
x_test = pad_sequences(tokenizer.texts_to_sequences(df_test.text),
maxlen=SEQUENCE_LENGTH)
print(x_train[0])


print(x_train)
print("----------------------------------------")
print(x_test)


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


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)


print("x_train", x_train.shape)
```

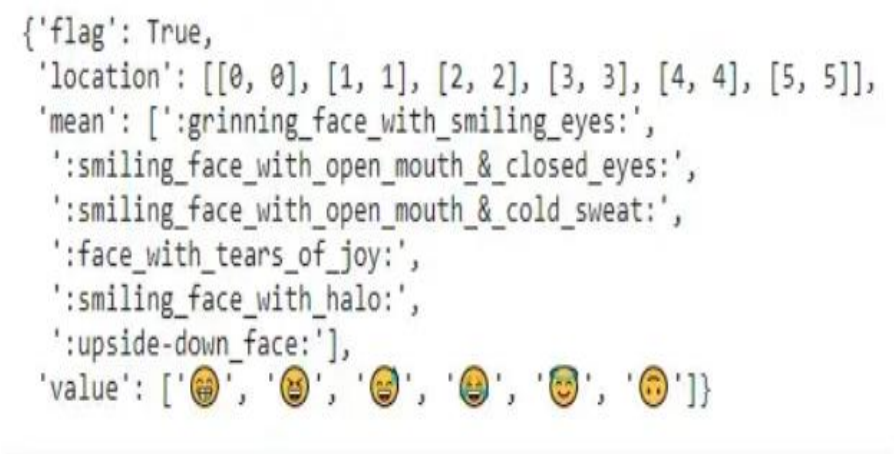
```
print("y_train", y_train.shape)
print()
print("x_test", x_test.shape)
print("y_test", y_test.shape)


y_train[:10]


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)
embedding_matrix[9]

embedding_matrix

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

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()

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

callbacks = [ ReduceLROnPlateau(monitor='val_loss', patience=5, cooldown=0),
```

```
        EarlyStopping(monitor='val_accuracy', min_delta=1e-4, patience=5)]

%time
history = model.fit(x_train, y_train,
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,
                    validation_split=0.1,
                    verbose=1,
                    callbacks=callbacks)


%%time
score = model.evaluate(x_test, y_test, batch_size=BATCH_SIZE)
print()
print("ACCURACY:",score[1])
print("LOSS:",score[0])

acc = history.history['accuracy']
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()


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

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}

predict("I love the music")
```

RESULT

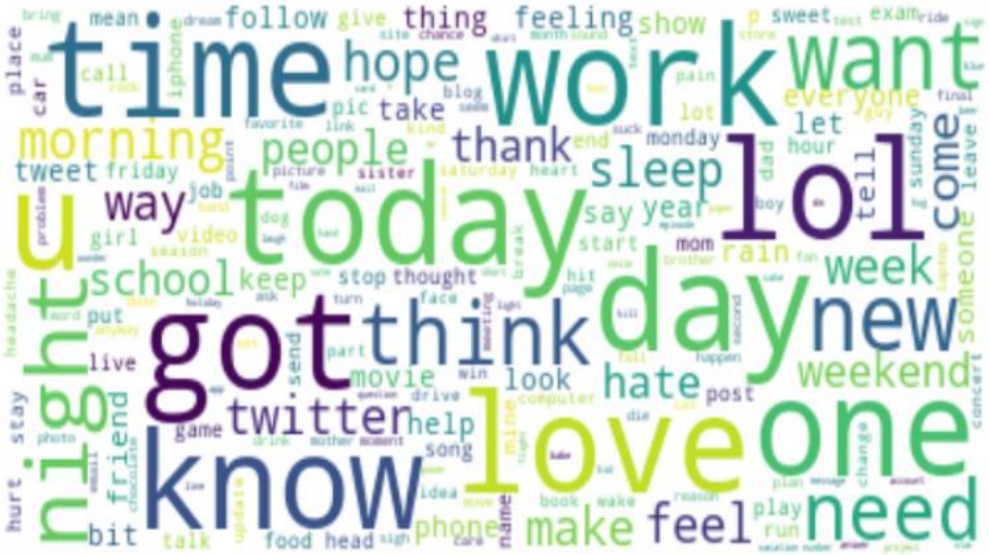
The First step is converting the emojis to their respective texts using the demote package during the data mining process through Twitter API.



The non-alphabetic characters are removed in this transition.

The next step is to remove the stop words from the texts so that the quality of the final word vectors will be high and non-relevant and outliers are removed from the Tweets before creating our embedding layer using the gensim word2vec model.

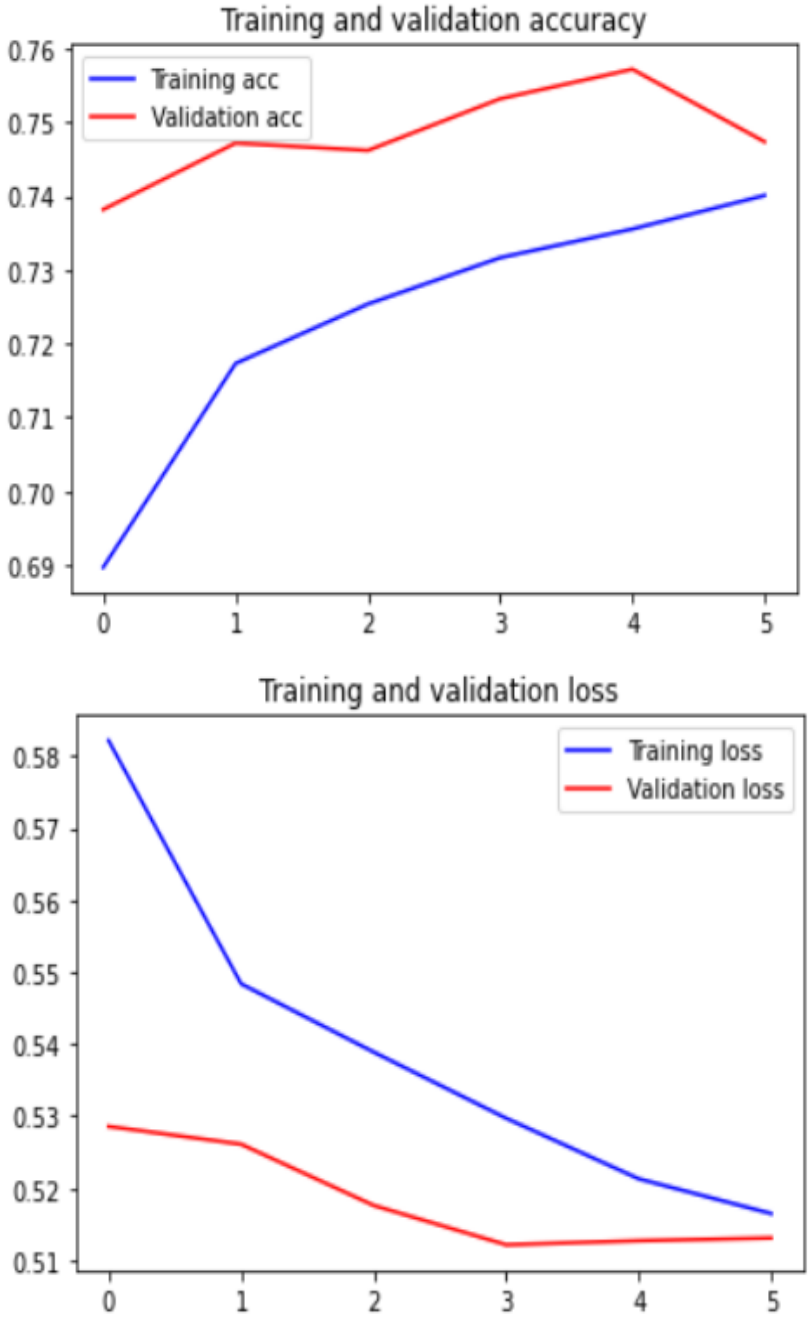
After removing the stop words, we are left with all the important and relevant words which helps to classify the polarity of texts.



Then the word2vec is used to create vectors and words with most similar meanings. This helps to find the root words and help us make it classify the polarity using the embedding matrix created for all these individual data.

```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.35748425, -0.03806522, -0.02751208, ...,  0.09821914,
         0.29414114, -0.61697841],
       [-0.31881461,  0.09820278, -0.37134796, ...,  0.0489414 ,
        -0.45882538, -0.38045585],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

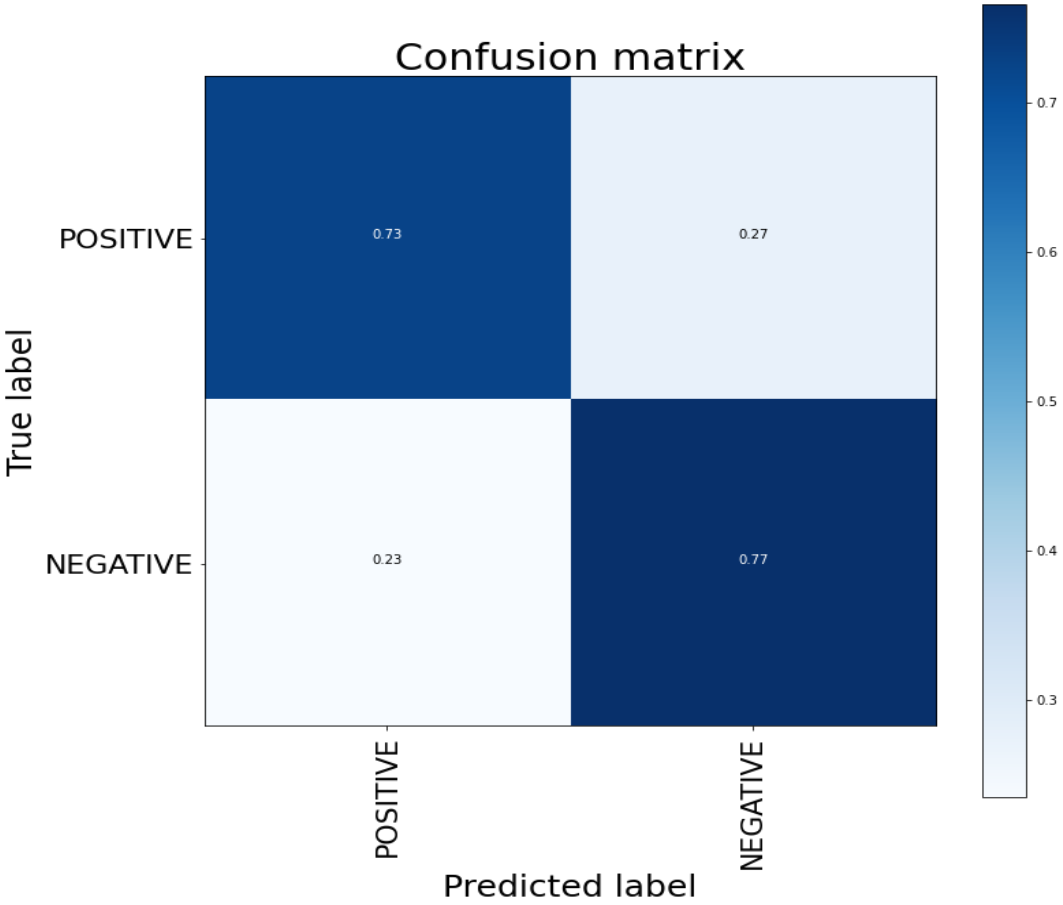
This embedding Layer is fit into the Convolutional Neural Network Sequential Model and the model is trained. The time for each model takes up a huge amount of time as each text is converted to a vector of 300 words of array. This each time the model is trained and overlapped with a loss of 50%. The final output of the accuracy and the confusion matrix is given below.



As the accuracy of our trained model is given above and we can see the clear increase of the accuracy through epoch is clearly seen. This model can be further improved as we only trained it for 6 hours and by utilizing all the data we can get accuracy up to more than 98%. The present model's confusion matrix is present below and we can see the true positive compared with other three factors in it.

```
1 predict("The World is at Peace")
```

```
1/1 [==============================] - 1s 519ms/step

{'label': 'POSITIVE',
 'score': 0.9010736346244812,
 'elapsed_time': 0.6069979667663574}
```



CONCLUSION

This project about the sentiment analysis can be categorized from a variety of angles, that includes the method employed, the perspective on the text, the amount of text analysis in detail, the rating level, and many more.

We identified machine learning, lexicon-based, statistical, and rule-based techniques from a technological perspective.

This model is trained using a known dataset, the machine learning method employs different learning algorithms to ascertain the sentiment. the lexicon-based method involves determining a review's emotion polarity based on the semantic orientation of its words or sentences. a text's subjectivity and viewpoint are measured by its "semantic orientation".

Twitter sentiment is the category of the text and option mining. It focuses on analyzing the sentiment of tweets, feeding the data into a machine learning model, training it, checking its accuracy, and depending on the results, making this model available for future use.

It also includes steps such as data collection, text preprocessing, sentiment detection, sentiment classification, training, and model testing.

Therefore, we can conclude that Twitter Sentiment analysis has a very bright future.

REFERENCES

[1] CNN for situations understanding based on sentiment analysis of twitter data by Shiyang
Liaoa, Junbo Wangb, Ruiyun Yua, Koichi Satob, Zixue Cheng at 8th International Conference on Advances in Information Technology, IAIT 2016, 19-22 (December 2016)

[2] A sentiment analysis study for twitter using the various model of convolutional neural network by Sartini, Subiyanto and M F Alim at Journal of Physics: Conference Series, ICMSE 2020 (2021)

[3] Twitter sentiment analysis using distributed word and sentence representation by Dwarampudi Mahidhar Reddy, Dr. N V Subba Reddy and Dr. Prema K V at arXiv :1904.12580v1 [cs.IR] (1 April 2019)

[4] Using Word2Vec to Process Big Text Data by Long Ma and Yanqing Zhang conference held at Big Data (Big Data), 2015 IEEE International ConferenceAt: San Jose, CA (October 2015)

[5] Sentiment Analysis of Twitter Data: A Survey of Techniques by Vishal A.Kharde and S.S.Sonawane held at International Journal of Computer Applications 139(11): 5-15 (April 2016)

- https://liris.cnrs.fr/Documents/Liris-6508.pdf
- https://towardsdatascience.com/getting-started-with-data-collection-using twitter-api-v2-in-less-than-an-hour-600fbd5b5558
- https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/
- https://www.kaggle.com/code/nitin194/twitter-sentiment-analysis-word2vec-doc2vec

- https://towardsdatascience.com/predicting-tweet-sentiment-with-word2vec-embeddings-67aace9b019d

- https://www.kaggle.com/code/paoloripamonti/twitter-sentiment-analysis