LIVE TWEET ANALYSIS

PROJECT REPORT
A2 SLOT

Submitted by

Veera Vijayaprasad C, 19BCE2127

Nitin Pramod Ranjan, 18BCE0272

Siddharth Garg, 18BCE0525

Pranav Batta, 18BCE0670

Course Code: CSE3021

Course Title: Social and Information Networks

Under the guidance of

Prof. Illantheral KPSK

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

Live Tweet Analysis

Winter Sem 2020-'21 Project

VIT –Vellore Institute of Technology

ABSTRACT

Sentiment analysis, also referred to as opinion mining, is a sub machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling objects, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to work on. In this project we choose to try to classify tweets from Twitter into "positive" or "negative" sentiment by building a model based on probabilities. Twitter is a microblogging website where people can share their feelings quickly and spontaneously by sending tweets limited by 140 characters. You can directly address a tweet to someone by adding the target sign "@" or participate to a topic by adding an hashtag "#" to your tweet. Because of the usage of Twitter, it is a perfect source of data to determine the current overall opinion about anything.

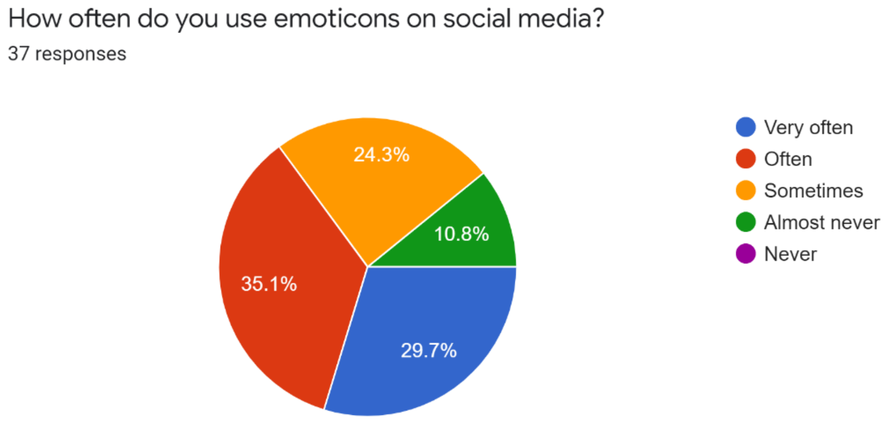
Keywords: Emoticons, Lexicons, Sentiment, Dataset, Positive Review, Negative Review

I. INTRODUCTION

Sentiment Analysis refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry", "sad", and "happy". Precursors to sentimental analysis include the General Inquirer, which provided hints toward quantifying patterns in text and, separately, psychological research that examined a person's psychological state based on analysis of their verbal behaviour. Subsequently, the method described in a patent by Volcani and Fogel, looked specifically at sentiment and identified individual words and phrases in text with respect to different emotional scales. Many other subsequent efforts were less sophisticated, using a mere polar view of sentiment, from positive to negative, such as work by Turney, and Pang who applied different methods for detecting the polarity of product reviews and movie reviews respectively. This work is at the document level. One can also classify a document's polarity on a multi-way scale, which was attempted by Pang and Snyder among others: Pang and Lee expanded the basic task of classifying a movie review as either positive or negative to predict star ratings on either a 3- or a 4-star scale, while Snyder performed an in-depth analysis of restaurant reviews, predicting ratings for various aspects of the given restaurant, such as the food and atmosphere (on a five star scale). First steps to bringing together various approaches—learning, lexical, knowledge-based, etc.—were taken in the 2004 AAAI Spring Symposium where linguists, computer scientists, and other interested researchers first aligned interests and proposed shared tasks.

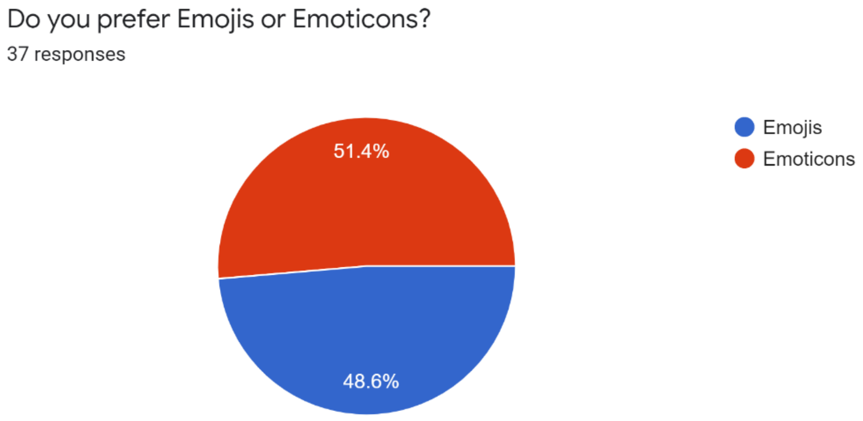
II. SURVEY (GOOGLE FORM)

We circulated a google form containing 7 questions related to usage of emoticons by people on social media platforms. Here are the responses we received.



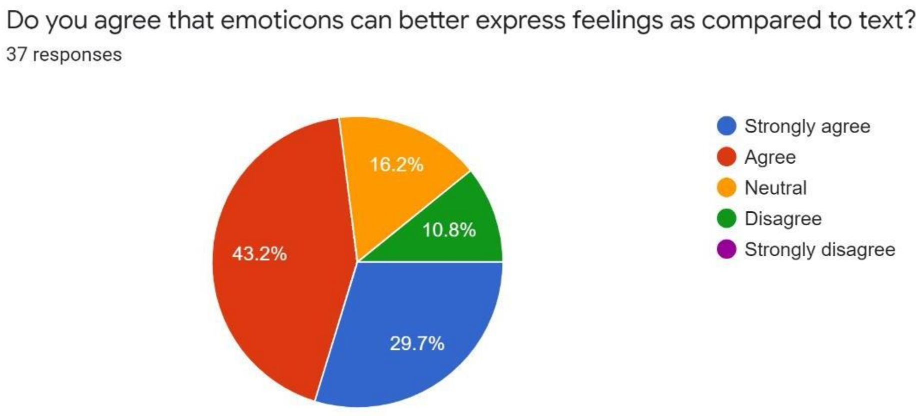
How often do you use emoticons on social media?

37 responses



Do you prefer Emojis or Emoticons?

37 responses

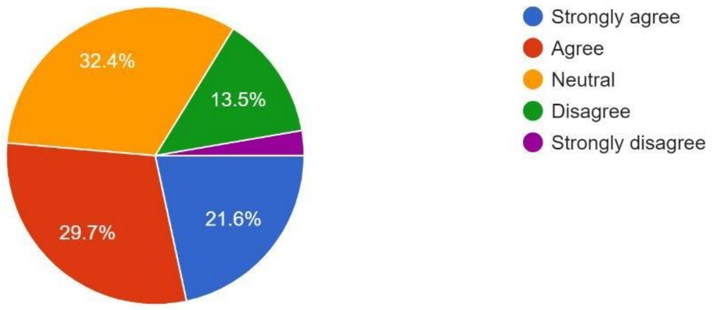


Do you agree that emoticons can better express feelings as compared to text?

37 responses

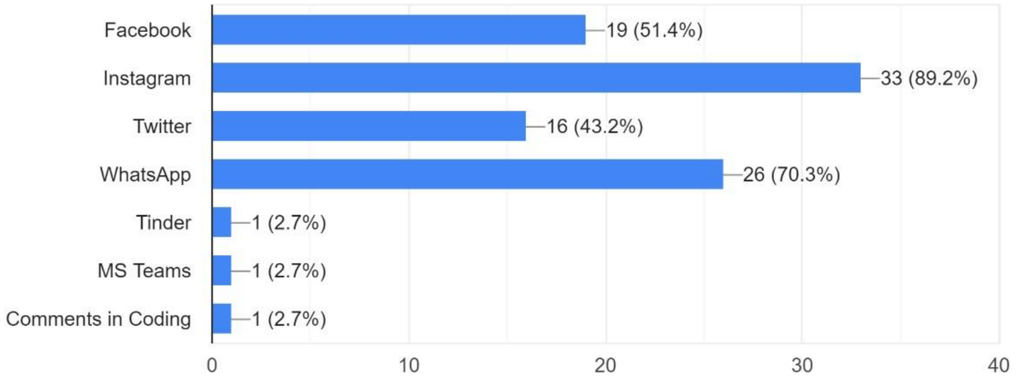
Do you agree that just using emoticons and no text enough to express your feelings?

37 responses



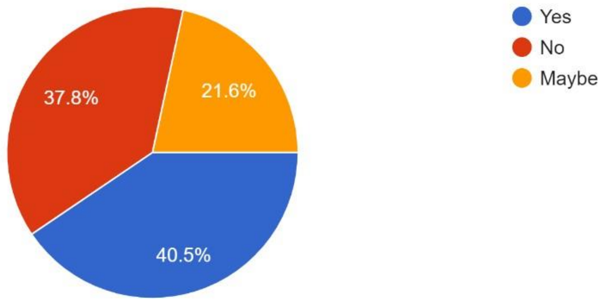
On which social media platforms do you use emoticons?

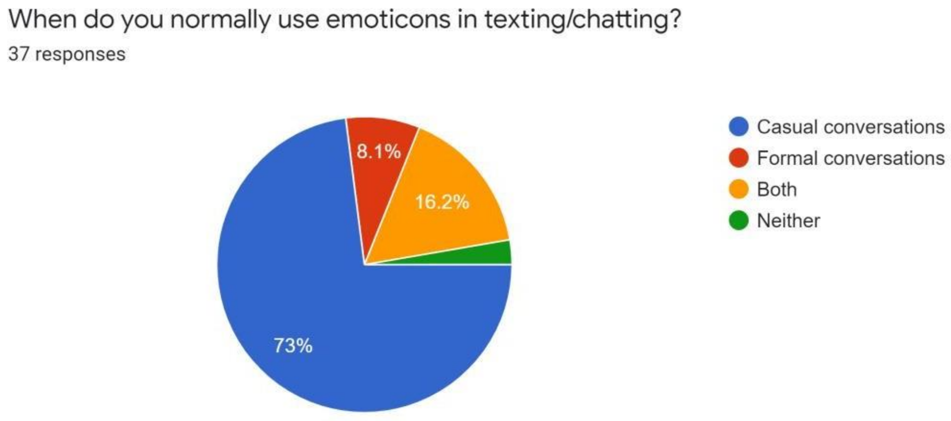
37 responses



Do you know the meaning of all emoticons available?

37 responses





From the above form it can be concluded that the Emoticons can be used as a way to express sentiments and can be a good measure for the same.

III. Literature Review/ Related Work

1.	**Sentiment Analysis of Twitter Data containing Emoticons: A Survey**
Author	*Jayanthi T, Sreeja K*
Dataset	Data from blogs, review sites, web discourse, news articles and 96,269,892 tweets containing emoticons.
Objective / Work done	In this paper, the authors tried to review the presence of emoticons in micro-blogging site twitter and the different techniques for classifying these emoticons in order to predict the sentiment of the given tweet.
Methodology	Different classification techniques are applied, namely, lexicon based method which comes under unsupervised learning, Naive based method and SVM (Support Vector Machines) which come under supervised learning method. Various techniques used in achieving this are preprocessing, POS Tagging, Stemming, Exaggerated word shortening, Emoticon detection, Hashtag Detection and stop words.
Result / Achievement	In this paper, the authors reviewed fundamental aspects of three classification methods in detail and compared and analysed them. They also surveyed reviews, blogs, news articles and tweets to evaluate the opinions

2.	**Exploiting Emoticons in Sentiment Analysis**
Author	*Alexander Hogenboom, Daniella Bal, Flavius Frasincar, Malissa Bal, Franciska de Jong, Uzay Kaymak*
Dataset	A proprietary maximum-entropy based POS tagger for Dutch and a proprietary sentiment lexicon for Dutch words
Objective / Work done	They proposed a novel framework for automated sentiment analysis, which takes into account the information conveyed by emoticons. The goal of this framework was to detect emoticons, determine their sentiment, and assign the associated sentiment to the affected text in order to correctly classify the polarity of natural language text as either positive or negative.
Methodology	Their framework is essentially a pipeline in which each component fulfills a specific task in analyzing the sentiment of an arbitrary document. After splitting the document, each text segment is checked for the presence of emoticons. These emoticons may be organized into emoticon synsets, which we define as groups of emoticons denoting the same emotion. After determining the sentiment conveyed by each individual text segment, all text segments are recombined into a single document
Result / Achievement	The considered sentiment analysis approaches exhibit clear differences in terms of performance. Overall, sentence level accounting for emoticon sentiment yields an increase in accuracy and macro-level F1 from 22% to 59% and from 22% to 65%, respectively. Assuming the sentiment conveyed by emoticons to affect the surrounding text on a paragraph level increases both overall polarity classification accuracy and macro-level F1 even further to 94%.

3.	**Joint Embedding of Emoticons and Labels Based on CNN for Microblog Sentiment Analysis**
Author	*Yongcai Toa, Xinqian Zhang, Lei Shi, Lin Wei, Zhaoyang Hai, Junaid Abdul Wahid*
Dataset	Two public chinese microblog benchmark corpora consisting of training and testing sets each, including eight different emotion labels.
Objective / Work done	To avoid inadequacies in the sentimental analysis, this research proposes EL-CNN, which basically contains three parts, namely, input encoder, label encoder, and matcher.
Methodology	The authors conduct four experiments in Google Colaboratory which are preprocessing which is done to improve segmentation accuracy, Distributed Representation of words which uses chinese word vectors to achieve the distribution, Model Hyperparameter Settings which are applied on the four experiments.
Result / Achievement	The authors compare the parameters and calculation speeds of CNN, EMCNN, DAM and EL-CNN. They also compare the number of compositional parameters

	in the models, computational complexity, and sequential steps of these four models. For the computational complexity, their model was almost the same as the most simple CNN model and is also smaller than EMCNN or DAM models

4.	**Sentiment Analysis of Emoticon Based Neuro Fuzzy System**
Author	*Ritu Agarwal, Sourabh Shreshth*
Dataset	14123 tweets were collected
Objective / Work done	The research work primarily shows the use of a neurofuzzy system in the sentimental analysis of twitter data using emoticons as a core ingredient of the data set and compare the performance with other classifiers. The methodology used can be primarily divided into three steps data pre-processing, Sentiment score generation and neuro fuzzy classifier based model generation
Methodology	Using a python library tweepy, we collected recent tweets from tweet streams and filtered the tweets based on emoticons, language and retweets. A total of 14123 tweets were collected and processed in a preprocessing step to generate our input data set and were then stemmed. After the stemming process, nearly 5000 tweets were left. They used a combination of regular expression and control statements in our code to identify POS tags to generate sentiment scores required as an input to a neuro fuzzy system.
Result / Achievement	They compared the accuracy of their proposed model in comparison to other well-known classifiers, and the result was satisfactory. Emoticon based neuro fuzzy systems perform with better accuracy and minimal execution time in comparison to other classifiers in their project. They also observed an accuracy of 87.14% with the proposed system.

5.	**Thumbs up? Sentiment Classification using Machine Learning Techniques**
Author	*Shivakumar Vaithyanathan, Bo Pang and Lillian Lee*
Dataset	Internet Movie Database (IMDb) archive of the rec.arts.movies.reviews newsgroup
Objective / Work done	The main aim in this work was to examine whether it suffices to treat sentiment classification simply as a special case of topic-based categorization (with the two “topics” being positive sentiment and negative sentiment), or whether special sentiment-categorization methods need to be developed.
Methodology	They experimented with three standard algorithms: Naive Bayes classification, maximum entropy classification, and support vector machines. They used documents from the movie-review corpus to create a data set with uniform class distribution and then randomly selected 700 positive sentiment and 700 negative sentiment documents. They focused on features based on unigrams (with negation tagging) and bigrams. Because training MaxEnt is expensive in the numb
Result /	As a whole, the machine learning algorithms clearly surpass the

Achievement	random-choice baseline of 50%. They also handily beat our two human-selected-unigram baselines of 58% and 64%, and, furthermore, perform well in comparison to the 69% baseline achieved via limited access to the test-data statistics, although the improvement in the case of SVMs is not so large. This provides suggestive evidence that sentiment categorization is more difficult than topic classification, which corresponds to the intuitions of the text categorization.

6.	**Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web**
Author	*Sanjiv R. Das, Mike Y. Chen*
Dataset	First, an electronic English "dictionary", which provides base language data. Second, a "lexicon" which is a hand-picked collection of finance words. The third database is the "grammar" or the training corpus. It is a base set of rules used in statistical analysis.
Objective / Work done	The first part of the paper is the extraction of opinions from message board postings to build a sentiment index. Messages are classified by our algorithms into one of 3 types: bullish (optimistic), bearish (pessimistic) and neutral (comprising either spam or messages that are neither bullish nor bearish). They have used five algorithms, each with different conceptual underpinnings, to classify each message. The algorithms "learn" sentiment classification rules from the pre-classified data set, and then apply these rules.
Methodology	Each of the five classifier algorithms used by the authors relies on a different approach to message interpretation. Some of them are language independent, and some are not. Each approach is intuitive. They are all analytical, and do not require any lengthy optimization, or convergence issues, hence they are computationally efficient, making possible the processing of huge volumes of data in real time.
Result / Achievement	They developed a methodology for extracting small investor sentiment from stock message boards. Time series and cross sectional aggregation of message information improves the quality of the sentiment index. Empirical applications evidence a relationship with stock returns. Sentiment aggregated across stocks tracks index returns more strongly than with individual stocks. Preliminary evidence suggests that market activity influences small investor sentiment. Thus, the algorithms developed in this paper can be used to assess the impact on investor opinion of management announcements, press releases, third-party news, and regulatory changes.

7.	**A Twitter Based Opinion Mining to Perform Analysis Geographically**
Author	*Dhruvi K. Zala, Ankita Gandhi*
Dataset	First, Amazon product reviews for Unigram, 2-gram and 3-gram analysis. Second, the Twitter dataset which consists of all the tweets for N-gram analysis.
Objective / Work done	In this research paper, Sentiment analysis on Twitter dataset is performed to predict the current affairs on Twitter data to make surveys and prediction about the current affairs, user's behaviour and opinion using twitter dataset. The analysis is performed in a geographically state-wide manner.
Methodology	The n-gram algorithm discussed by the author has been developed which combines all the features of unigram, 2-gram and 3-gram. This makes the prediction more accurate.The analysis also includes hybrid features which considers emoticons, synonyms and acronyms to make the prediction process even more accurate.The new proposed system uses a hybrid dictionary that is a combination of lexicon and Afinn dictionaries.
Result / Achievement	They developed the NSR plot for N-gram with all features including Emoticons, Synonyms, and Acronyms with its NSR value. If a N-gram model is used, then at the time of feature extraction it will consider a combination of unigram, bigram and trigram and also it will consider all features at the feature extraction phase which gives better results compared to all other methods.
8.	**Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews**
Author	*Peter D. Turney*
Dataset	410 reviews from Epinions 2 , randomly sampled from four different domains: reviews of automobiles, banks, movies, and travel destinations.
Objective / Work done	In this paper, the author presents a simple unsupervised learning algorithm for classifying a review as recommended or not recommended. The algorithm takes a written review as input and produces a classification as output.
Methodology	The algorithm has three steps: (1) extract phrases containing adjectives or adverbs, (2) estimate the semantic orientation of each phrase, and (3) classify the review based on the average semantic orientation of the phrases and classify the review as recommended if the average is positive and otherwise not recommended. The core of the algorithm is the second step, which uses PMI-IR to calculate semantic orientation.
Result / Achievement	The algorithm attains an average accuracy of 74%. It appears that movie reviews are difficult to classify, because the whole is not necessarily the sum of the parts; thus the accuracy on movie reviews is about 66%. On the other hand, for banks and automobiles, it seems that the whole is the sum of the parts, and the accuracy is 80% to 84%.

9.	**Emoji and Emoticon in Tweet Sentiment Classification**
Author	*Amalia Anjani Arifiyanti, Eka Dyar Wahyuni*
Dataset	Dataset of tweets it comprises 818 tweets consisting of 345 positive tweets and 473 negative tweets.
Objective / Work done	This paper covers many emoticons and emojis that occur in sentiment tweets with Indonesian Twitter's users. It also examines how emoticons and emojis affect the text-based classification process regarding how emojis and emoticons are converted.
Methodology	The steps include data collection, data preparation, text preprocessing, which consists of steps for removing noise and selecting important terms, building classification model, and evaluating classification model performance.
Result / Achievement	SVM has the best and stable performance model from the three learning algorithms used in this study compared to BNB and MNB. This was followed by BNB and MNB in the last position.

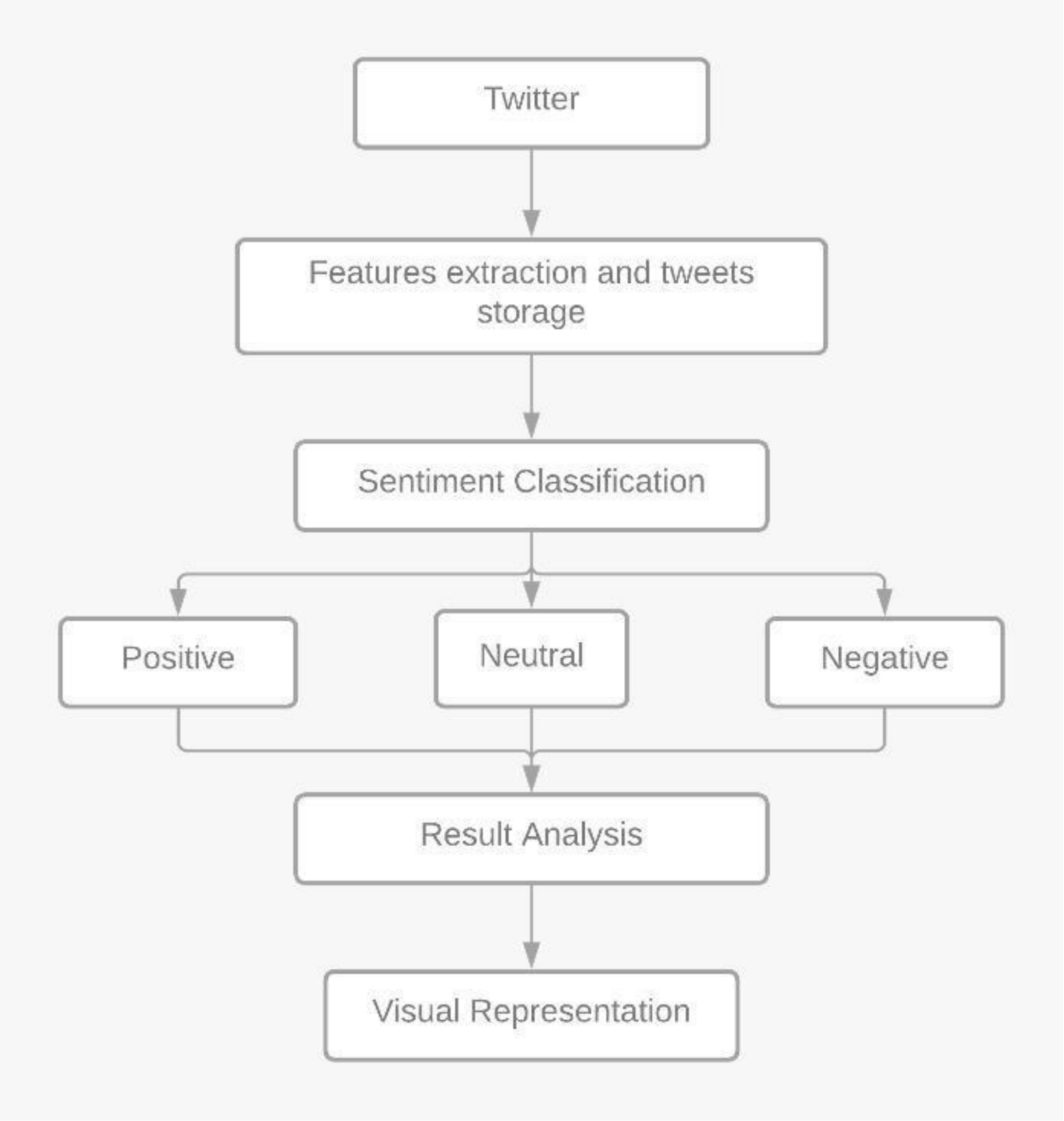
10.	**Newly-Coined Words and Emoticon Polarity for Social Emotional Opinion Decision**
Author	*Jin Sol Yang, Kwang Sik Chung*
Dataset	Korean-based newly-coined words and global emoticons are extracted from Twitter. The sensibility polarity of the coined word was input.
Objective / Work done	Proposed emoticon and newly-coined words polarity decision system calculates the polarity of newly-coined words and emoticons so that social data opinion polarity for keywords is decided. For the social data opinion polarity decision, polarity of emoticons and newly-coined words is calculated and registered as newly-coined words and emotion dictionary. And Proposed emoticon and newly coined words polarity decision system consists of Collection Keyword Input, Social data collection, Contents Pre-processing stage, Morphological Extraction Process and Emotional Classification Process
Methodology	Firstly words and emoticons are extracted by the existing morphemes and words analysis method and emotional data are extracted by comparing them with the basic emotional dictionary from the extracted morpheme. In the second step, the newly-coined words and emoticons are extracted from the extracted emotional data before the proposed morphological analysis using the newly-coined word and emoticon dictionary. Emotion words, newly-coined words, and emoticons extracted from the extracted emotional data include polarity and weight, and polarity and weight are used for emotional classification.
Result / Achievement	926 positive tweets and 329 negative tweets are randomly selected from about 700,000 tweets and analyzed by proposed method. As a result of the analysis, when the newly-coined words and emoticon were added to the existing emotion dictionary, the accuracy measurement results were improved to 14% for positive tweets and 11% for negative tweets.

IV. PROBLEM STATEMENT

- For analysing sentiment about a specific query using tweets from Twitter.
- Applying K means to the dataset collected from Twitter to find the most accurate machine learning model

V. PROCEDURES FOR PROPOSED METHOD WITH AN ALGORITHM

Flow diagram

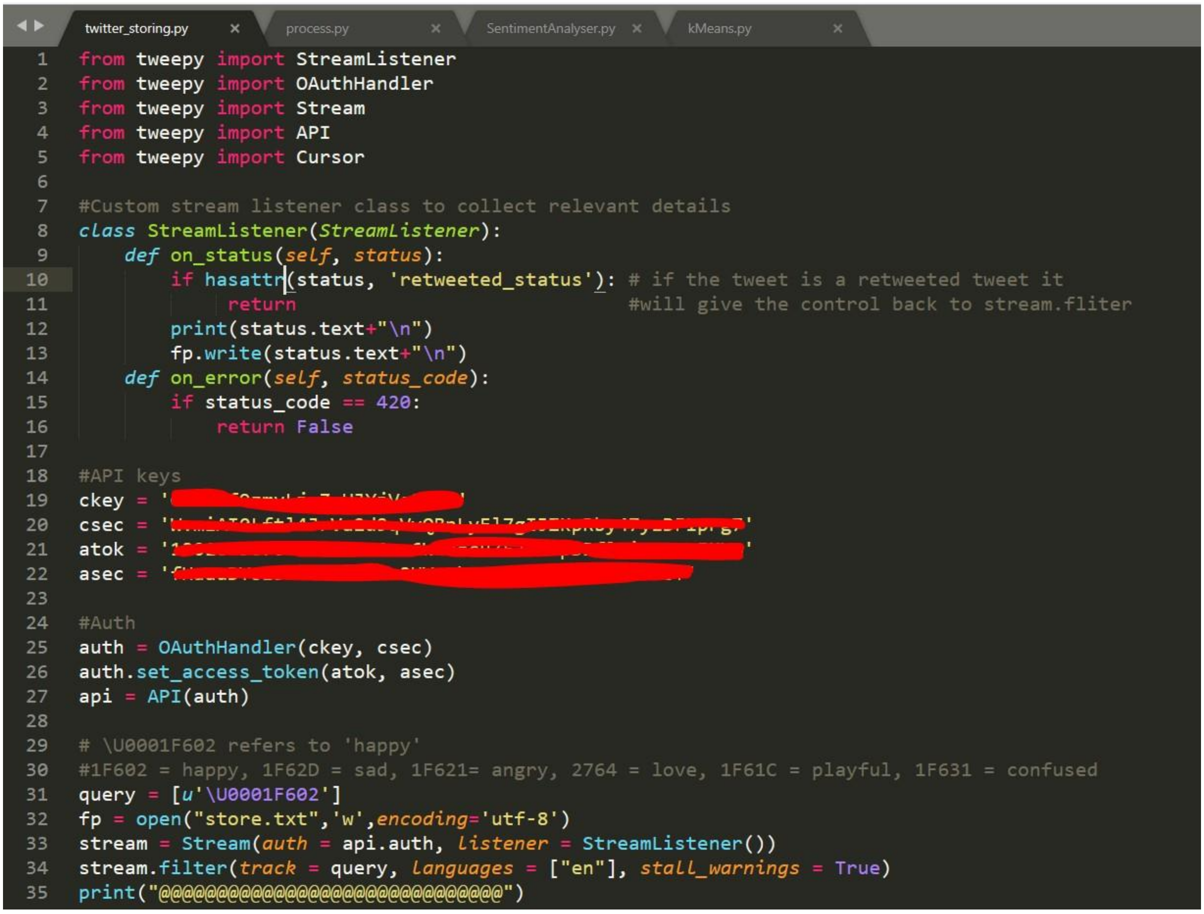


Algorithm

- Extracting data from Twitter using a Twitter API about a specific query.
- The data obtained will be given to the data wrangling model to remove unnecessary details like @username, #topic, links etc.
- The data will then sent to a data pre-processing model to encode the emoticons.
- The pre-processed data will be then sent to Vander Sentiment Analysis model.
- The model assigns specific Negative, Neutral, Positive score to the tweet .
- Then a cluster will be formed using these scores by using K-Means algorithm.

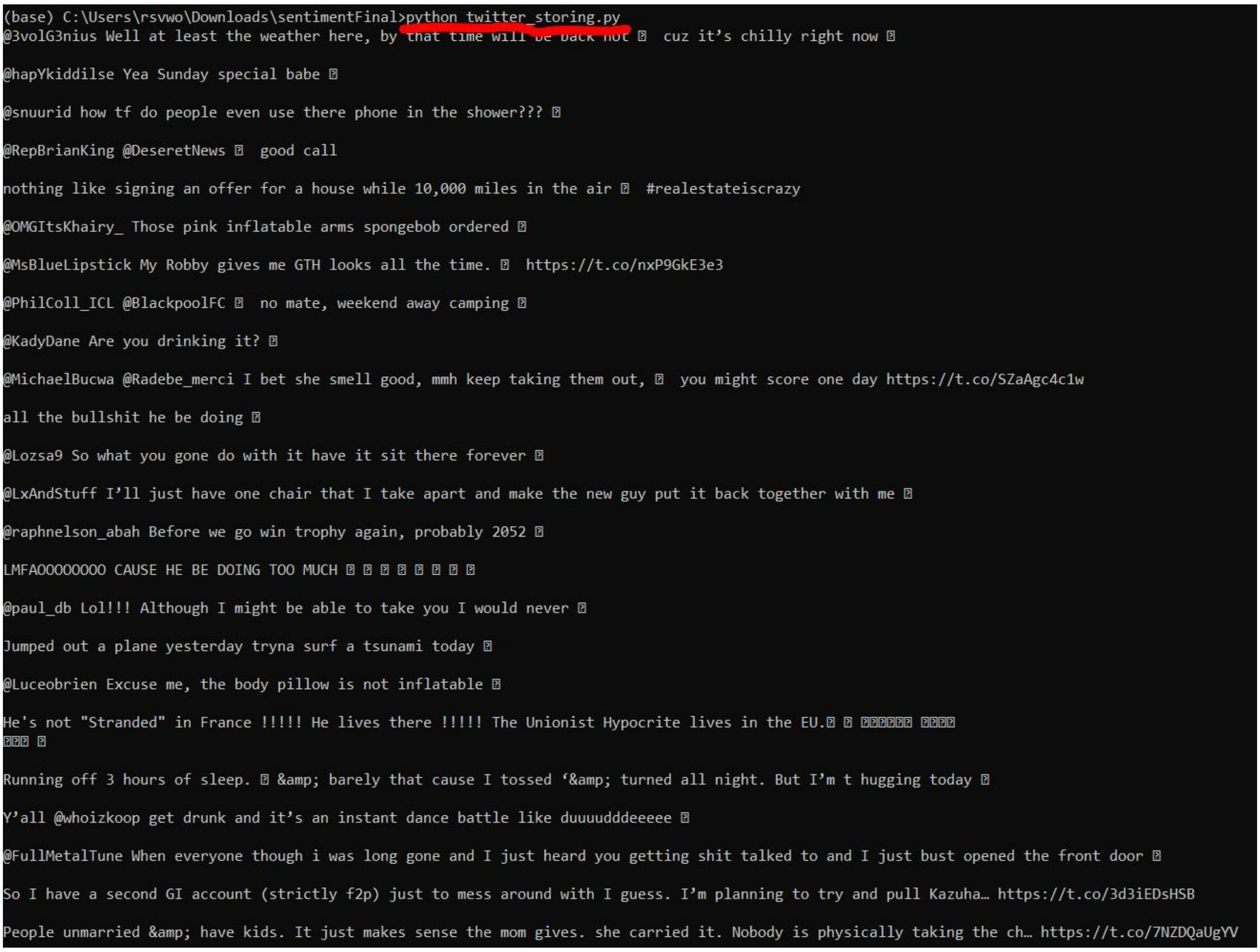
VI. IMPLEMENTATION

First twitter_storing.py python file is executed, it extract the real time twitter tweets from the twitter using the authentication keys which we got by making Twitter Developer Account. The program store all the data in store.txt file.



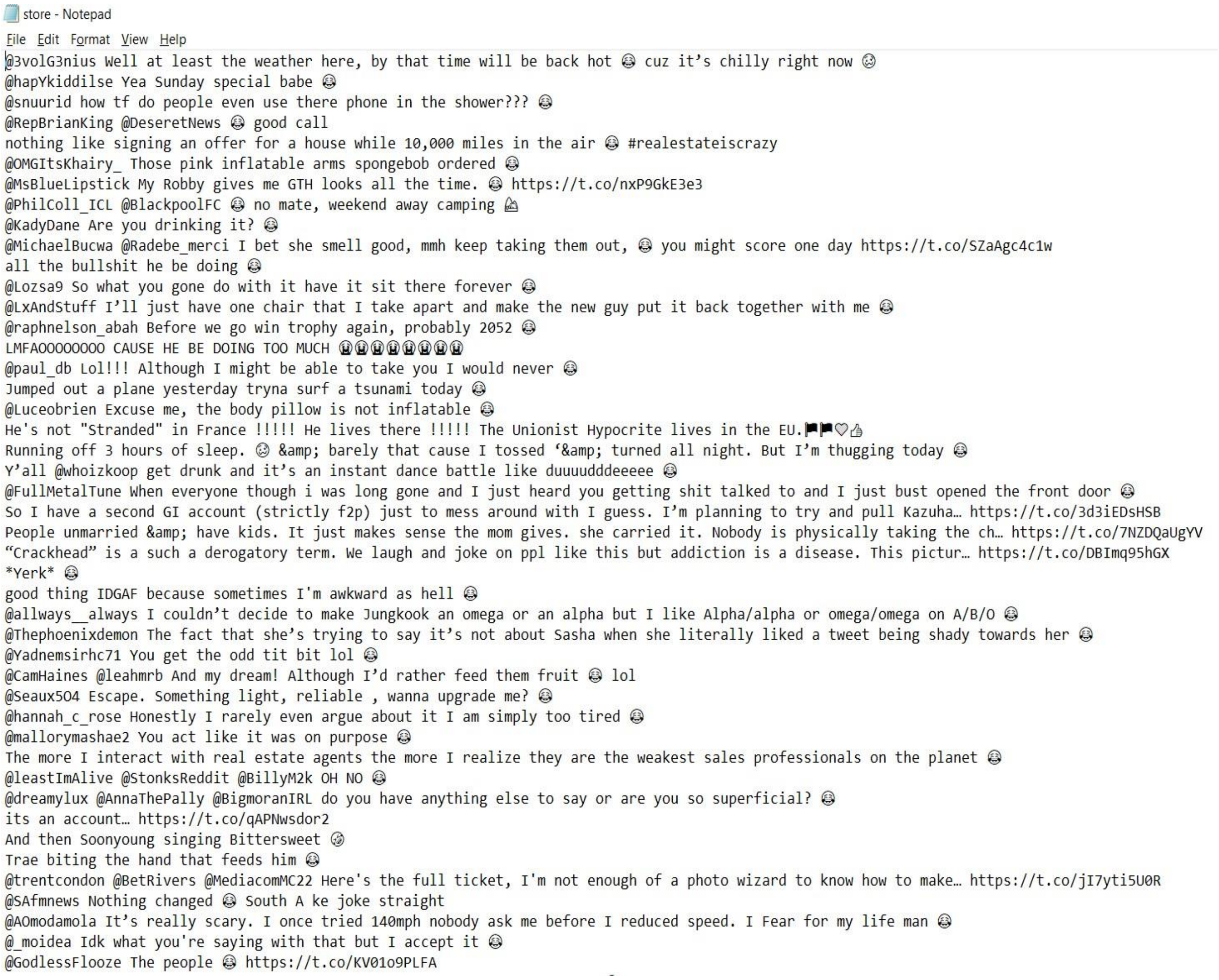
```python
from tweepy import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy import API
from tweepy import Cursor

#Custom stream listener class to collect relevant details
class StreamListener(StreamListener):
    def on_status(self, status):
        if hasattr(status, 'retweeted_status'): # if the tweet is a retweeted tweet it
            return                              #will give the control back to stream.fliter
        print(status.text+"\n")
        fp.write(status.text+"\n")
    def on_error(self, status_code):
        if status_code == 420:
            return False

#API keys
ckey = '[redacted]'
csec = '[redacted]'
atok = '[redacted]'
asec = '[redacted]'

#Auth
auth = OAuthHandler(ckey, csec)
auth.set_access_token(atok, asec)
api = API(auth)

# \U0001F602 refers to 'happy'
#1F602 = happy, 1F62D = sad, 1F621= angry, 2764 = love, 1F61C = playful, 1F631 = confused
query = [u'\U0001F602']
fp = open("store.txt",'w',encoding='utf-8')
stream = Stream(auth = api.auth, listener = StreamListener())
stream.filter(track = query, languages = ["en"], stall_warnings = True)
print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@")
```

Figure 1 Snip of twitter_storing.py. The authentication keys of Twitter Developer's account are hidden for security purpose.



```
(base) C:\Users\rsvwo\Downloads\sentimentFinal>python twitter_storing.py
@3volG3nius Well at least the weather here, by that time will be back hot ⍰  cuz it’s chilly right now ⍰

@hapYkiddilse Yea Sunday special babe ⍰

@snuurid how tf do people even use there phone in the shower??? ⍰

@RepBrianKing @DeseretNews ⍰  good call

nothing like signing an offer for a house while 10,000 miles in the air ⍰  #realestateiscrazy

@OMGItsKhairy_ Those pink inflatable arms spongebob ordered ⍰

@MsBlueLipstick My Robby gives me GTH looks all the time. ⍰  https://t.co/nxP9GkE3e3

@PhilColl_ICL @BlackpoolFC ⍰  no mate, weekend away camping ⍰

@KadyDane Are you drinking it? ⍰

@MichaelBucwa @Radebe_merci I bet she smell good, mmh keep taking them out, ⍰  you might score one day https://t.co/SZaAgc4c1w

all the bullshit he be doing ⍰

@Lozsa9 So what you gone do with it have it sit there forever ⍰

@LxAndStuff I’ll just have one chair that I take apart and make the new guy put it back together with me ⍰

@raphnelson_abah Before we go win trophy again, probably 2052 ⍰

LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH ⍰ ⍰ ⍰ ⍰ ⍰ ⍰ ⍰ ⍰

@paul_db Lol!!! Although I might be able to take you I would never ⍰

Jumped out a plane yesterday tryna surf a tsunami today ⍰

@Luceobrien Excuse me, the body pillow is not inflatable ⍰

He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the EU.⍰ ⍰ ⍰⍰⍰⍰⍰⍰ ⍰⍰⍰⍰
⍰⍰⍰ ⍰

Running off 3 hours of sleep. ⍰ &amp; barely that cause I tossed ‘&amp; turned all night. But I’m t hugging today ⍰

Y’all @whoizkoop get drunk and it’s an instant dance battle like duuuudddeeeee ⍰

@FullMetalTune When everyone though i was long gone and I just heard you getting shit talked to and I just bust opened the front door ⍰

So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning to try and pull Kazuha… https://t.co/3d3iEDsHSB

People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody is physically taking the ch… https://t.co/7NZDQaUgYV
```

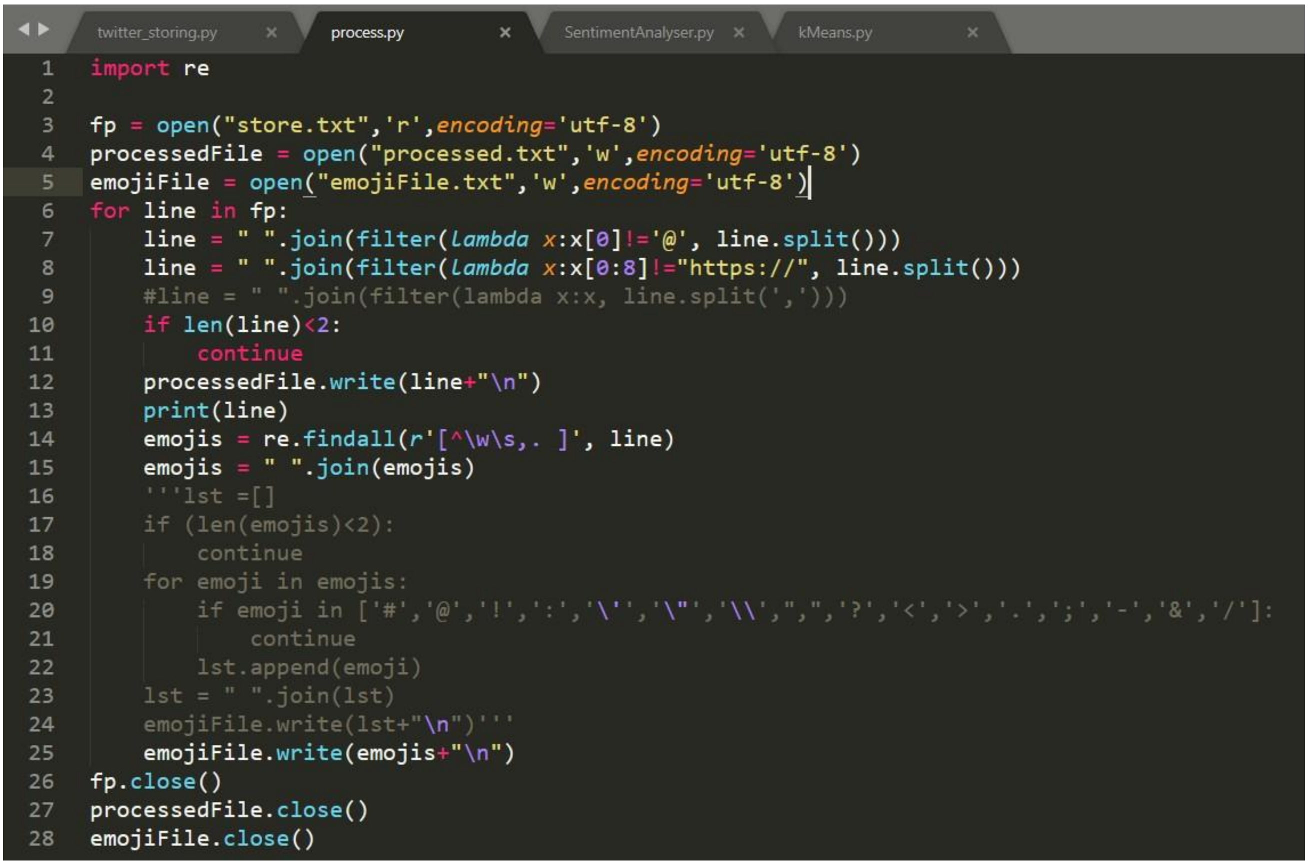
Figure 2 Implementation of twitter_storing.py



```
@3volG3nius Well at least the weather here, by that time will be back hot 😂 cuz it’s chilly right now 😥
@hapYkiddilse Yea Sunday special babe 😂
@snuurid how tf do people even use there phone in the shower??? 😂
@RepBrianKing @DeseretNews 😂 good call
nothing like signing an offer for a house while 10,000 miles in the air 😂 #realestateiscrazy
@OMGItsKhairy_ Those pink inflatable arms spongebob ordered 😂
@MsBlueLipstick My Robby gives me GTH looks all the time. 😂 https://t.co/nxP9GkE3e3
@PhilColl_ICL @BlackpoolFC 😂 no mate, weekend away camping ⛺
@KadyDane Are you drinking it? 😂
@MichaelBucwa @Radebe_merci I bet she smell good, mmh keep taking them out, 😂 you might score one day https://t.co/SZaAgc4c1w
all the bullshit he be doing 😂
@Lozsa9 So what you gone do with it have it sit there forever 😂
@LxAndStuff I’ll just have one chair that I take apart and make the new guy put it back together with me 😂
@raphnelson_abah Before we go win trophy again, probably 2052 😂
LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH 😭😭😭😭😭😭😭😭
@paul_db Lol!!! Although I might be able to take you I would never 😂
Jumped out a plane yesterday tryna surf a tsunami today 😂
@Luceobrien Excuse me, the body pillow is not inflatable 😂
He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the EU.🏴🏴♡👍
Running off 3 hours of sleep. 😥 &amp; barely that cause I tossed ‘&amp; turned all night. But I’m thugging today 😂
Y’all @whoizkoop get drunk and it’s an instant dance battle like duuuudddeeeee 😂
@FullMetalTune When everyone though i was long gone and I just heard you getting shit talked to and I just bust opened the front door 😂
So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning to try and pull Kazuha… https://t.co/3d3iEDsHSB
People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody is physically taking the ch… https://t.co/7NZDQaUgYV
“Crackhead” is a such a derogatory term. We laugh and joke on ppl like this but addiction is a disease. This pictur… https://t.co/DBImq95hGX
*Yerk* 😂
good thing IDGAF because sometimes I'm awkward as hell 😂
@allways__always I couldn’t decide to make Jungkook an omega or an alpha but I like Alpha/alpha or omega/omega on A/B/O 😂
@Thephoenixdemon The fact that she’s trying to say it’s not about Sasha when she literally liked a tweet being shady towards her 😂
@Yadnemsirhc71 You get the odd tit bit lol 😂
@CamHaines @leahmrb And my dream! Although I’d rather feed them fruit 😂 lol
@Seaux504 Escape. Something light, reliable , wanna upgrade me? 😂
@hannah_c_rose Honestly I rarely even argue about it I am simply too tired 😂
@mallorymashae2 You act like it was on purpose 😂
The more I interact with real estate agents the more I realize they are the weakest sales professionals on the planet 😂
@leastImAlive @StonksReddit @BillyM2k OH NO 😂
@dreamylux @AnnaThePally @BigmoranIRL do you have anything else to say or are you so superficial? 😂
its an account… https://t.co/qAPNwsdor2
And then Soonyoung singing Bittersweet 🤣
Trae biting the hand that feeds him 😂
@trentcondon @BetRivers @MediacomMC22 Here's the full ticket, I'm not enough of a photo wizard to know how to make… https://t.co/jI7yti5U0R
@SAfmnews Nothing changed 😂 South A ke joke straight
@AOmodamola It’s really scary. I once tried 140mph nobody ask me before I reduced speed. I Fear for my life man 😂
@_moidea Idk what you're saying with that but I accept it 😂
@GodlessFlooze The people 😂 https://t.co/KV01o9PLFA
```

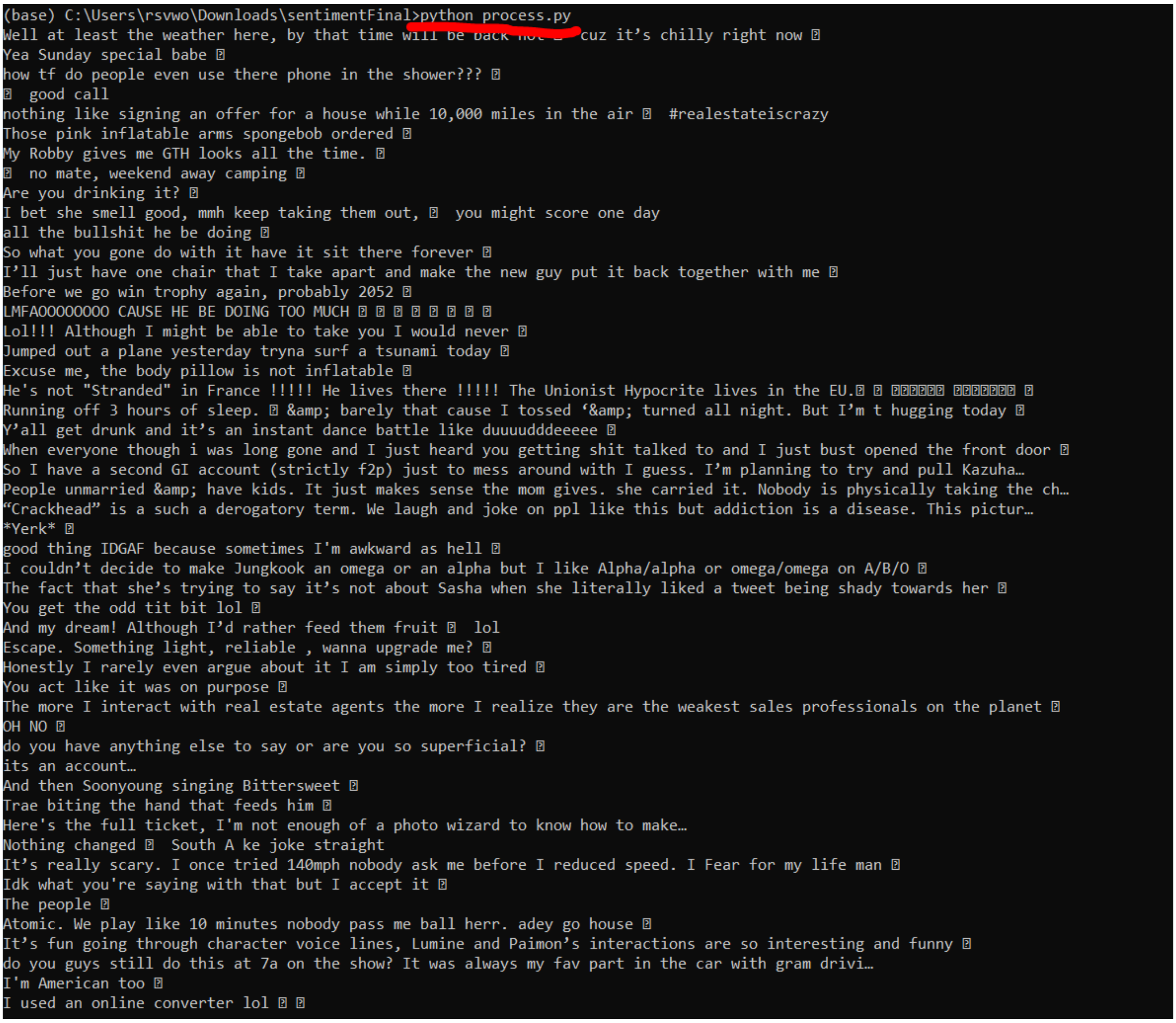
Figure 3 store.txt which stores the output of twitter_storing.py

After the required number of tweets are stored in store.txt file the file is then sent for pre processing so as to remove all the unnecessary details like username(@), links(https://). The processed data is then stored in a new text file processed.txt .



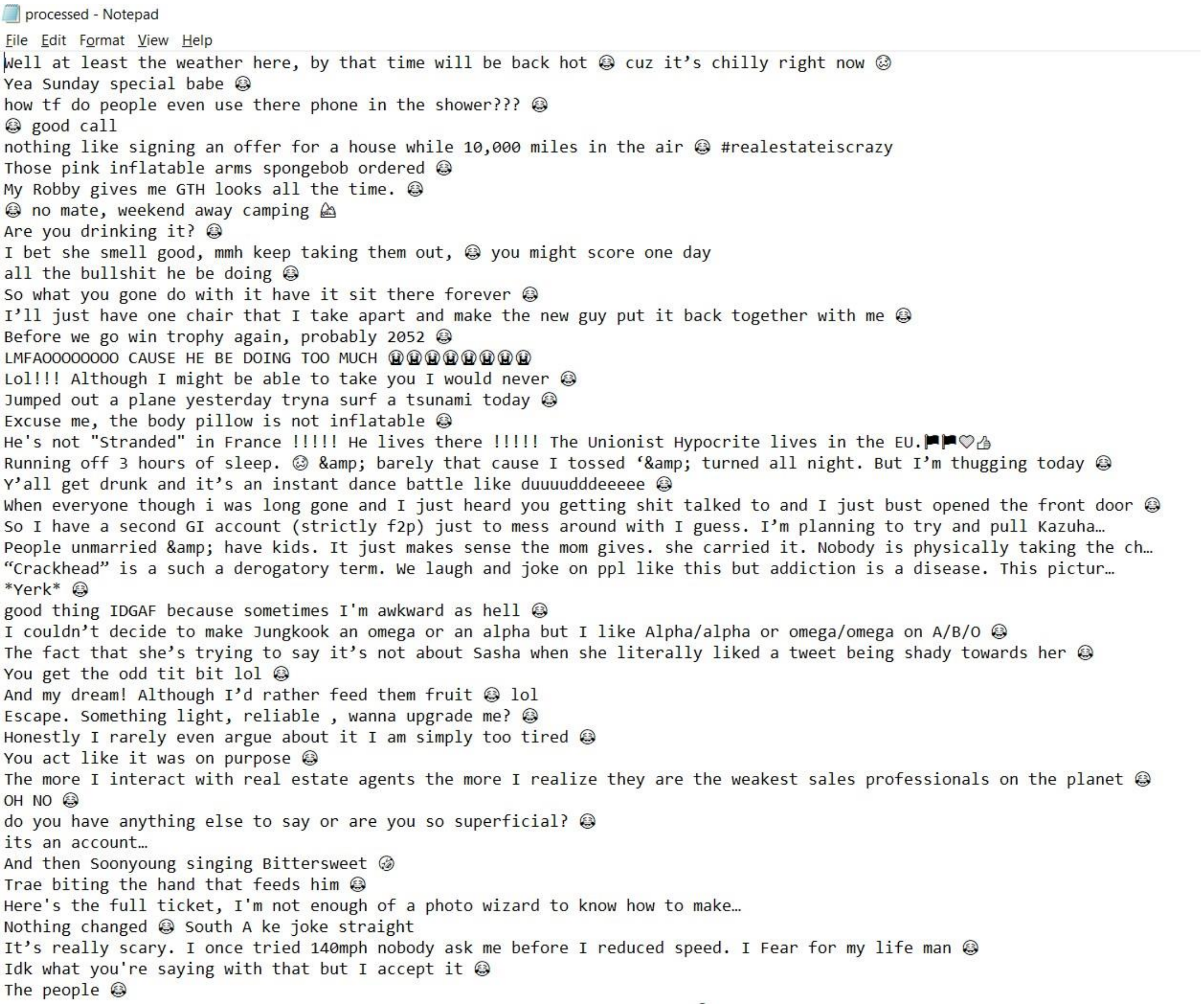
```python
import re

fp = open("store.txt",'r',encoding='utf-8')
processedFile = open("processed.txt",'w',encoding='utf-8')
emojiFile = open("emojiFile.txt",'w',encoding='utf-8')
for line in fp:
    line = " ".join(filter(lambda x:x[0]!='@', line.split()))
    line = " ".join(filter(lambda x:x[0:8]!="https://", line.split()))
    #line = " ".join(filter(lambda x:x, line.split(',')))
    if len(line)<2:
        continue
    processedFile.write(line+"\n")
    print(line)
    emojis = re.findall(r'[^\w\s,. ]', line)
    emojis = " ".join(emojis)
    '''lst =[]
    if (len(emojis)<2):
        continue
    for emoji in emojis:
        if emoji in ['#','@','!',':','\'','\"','\\',",",'?','<','>','.',';','-','&','/']:
            continue
        lst.append(emoji)
    lst = " ".join(lst)
    emojiFile.write(lst+"\n")'''
    emojiFile.write(emojis+"\n")
fp.close()
processedFile.close()
emojiFile.close()
```

Figure 4 Snip of process.py



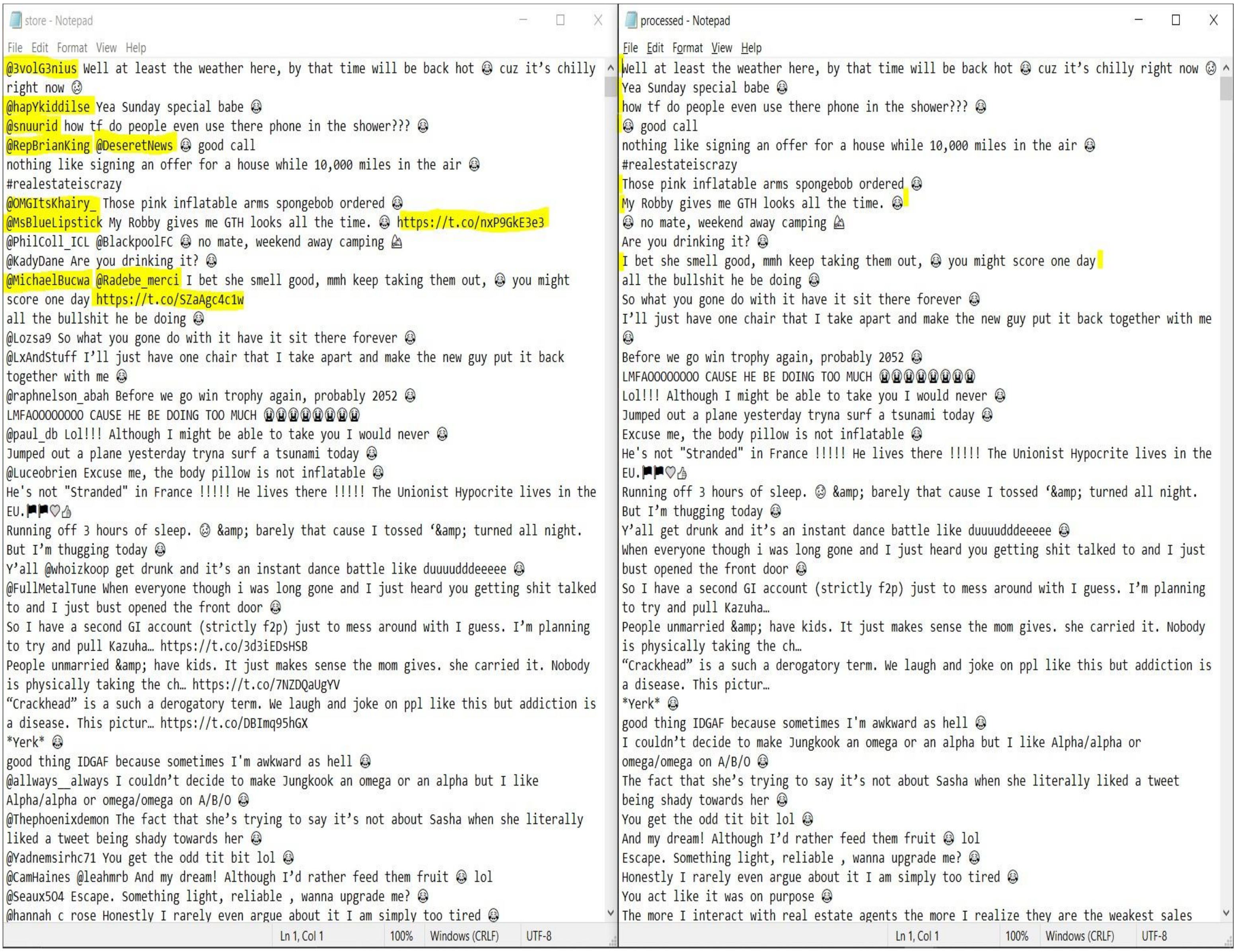
```
(base) C:\Users\rsvwo\Downloads\sentimentFinal>python process.py
Well at least the weather here, by that time will be back hot ⍰ cuz it’s chilly right now ⍰
Yea Sunday special babe ⍰
how tf do people even use there phone in the shower??? ⍰
⍰  good call
nothing like signing an offer for a house while 10,000 miles in the air ⍰  #realestateiscrazy
Those pink inflatable arms spongebob ordered ⍰
My Robby gives me GTH looks all the time. ⍰
⍰  no mate, weekend away camping ⍰
Are you drinking it? ⍰
I bet she smell good, mmh keep taking them out, ⍰  you might score one day
all the bullshit he be doing ⍰
So what you gone do with it have it sit there forever ⍰
I’ll just have one chair that I take apart and make the new guy put it back together with me ⍰
Before we go win trophy again, probably 2052 ⍰
LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH ⍰ ⍰ ⍰ ⍰ ⍰ ⍰ ⍰ ⍰
Lol!!! Although I might be able to take you I would never ⍰
Jumped out a plane yesterday tryna surf a tsunami today ⍰
Excuse me, the body pillow is not inflatable ⍰
He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the EU.⍰ ⍰ ⍰⍰⍰⍰⍰⍰ ⍰⍰⍰⍰⍰⍰⍰ ⍰
Running off 3 hours of sleep. ⍰ &amp; barely that cause I tossed ‘&amp; turned all night. But I’m t hugging today ⍰
Y’all get drunk and it’s an instant dance battle like duuuudddeeeee ⍰
When everyone though i was long gone and I just heard you getting shit talked to and I just bust opened the front door ⍰
So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning to try and pull Kazuha…
People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody is physically taking the ch…
“Crackhead” is a such a derogatory term. We laugh and joke on ppl like this but addiction is a disease. This pictur…
*Yerk* ⍰
good thing IDGAF because sometimes I'm awkward as hell ⍰
I couldn’t decide to make Jungkook an omega or an alpha but I like Alpha/alpha or omega/omega on A/B/O ⍰
The fact that she’s trying to say it’s not about Sasha when she literally liked a tweet being shady towards her ⍰
You get the odd tit bit lol ⍰
And my dream! Although I’d rather feed them fruit ⍰  lol
Escape. Something light, reliable , wanna upgrade me? ⍰
Honestly I rarely even argue about it I am simply too tired ⍰
You act like it was on purpose ⍰
The more I interact with real estate agents the more I realize they are the weakest sales professionals on the planet ⍰
OH NO ⍰
do you have anything else to say or are you so superficial? ⍰
its an account…
And then Soonyoung singing Bittersweet ⍰
Trae biting the hand that feeds him ⍰
Here's the full ticket, I'm not enough of a photo wizard to know how to make…
Nothing changed ⍰  South A ke joke straight
It’s really scary. I once tried 140mph nobody ask me before I reduced speed. I Fear for my life man ⍰
Idk what you're saying with that but I accept it ⍰
The people ⍰
Atomic. We play like 10 minutes nobody pass me ball herr. adey go house ⍰
It’s fun going through character voice lines, Lumine and Paimon’s interactions are so interesting and funny ⍰
do you guys still do this at 7a on the show? It was always my fav part in the car with gram drivi…
I'm American too ⍰
I used an online converter lol ⍰ ⍰
```

Figure 5 Implementation of process.py



```
Well at least the weather here, by that time will be back hot 😂 cuz it’s chilly right now 😢
Yea Sunday special babe 😂
how tf do people even use there phone in the shower??? 😂
😂 good call
nothing like signing an offer for a house while 10,000 miles in the air 😂 #realestateiscrazy
Those pink inflatable arms spongebob ordered 😂
My Robby gives me GTH looks all the time. 😂
😂 no mate, weekend away camping 🏕
Are you drinking it? 😂
I bet she smell good, mmh keep taking them out, 😂 you might score one day
all the bullshit he be doing 😂
So what you gone do with it have it sit there forever 😂
I’ll just have one chair that I take apart and make the new guy put it back together with me 😂
Before we go win trophy again, probably 2052 😂
LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH 😭😭😭😭😭😭😭😭
Lol!!! Although I might be able to take you I would never 😂
Jumped out a plane yesterday tryna surf a tsunami today 😂
Excuse me, the body pillow is not inflatable 😂
He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the EU.🏴🏴♡👍
Running off 3 hours of sleep. 😢 &amp; barely that cause I tossed ‘&amp; turned all night. But I’m thugging today 😂
Y’all get drunk and it’s an instant dance battle like duuuudddeeeee 😂
When everyone though i was long gone and I just heard you getting shit talked to and I just bust opened the front door 😂
So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning to try and pull Kazuha…
People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody is physically taking the ch…
“Crackhead” is a such a derogatory term. We laugh and joke on ppl like this but addiction is a disease. This pictur…
*Yerk* 😂
good thing IDGAF because sometimes I'm awkward as hell 😂
I couldn’t decide to make Jungkook an omega or an alpha but I like Alpha/alpha or omega/omega on A/B/O 😂
The fact that she’s trying to say it’s not about Sasha when she literally liked a tweet being shady towards her 😂
You get the odd tit bit lol 😂
And my dream! Although I’d rather feed them fruit 😂 lol
Escape. Something light, reliable , wanna upgrade me? 😂
Honestly I rarely even argue about it I am simply too tired 😂
You act like it was on purpose 😂
The more I interact with real estate agents the more I realize they are the weakest sales professionals on the planet 😂
OH NO 😂
do you have anything else to say or are you so superficial? 😂
its an account…
And then Soonyoung singing Bittersweet 🤣
Trae biting the hand that feeds him 😂
Here's the full ticket, I'm not enough of a photo wizard to know how to make…
Nothing changed 😂 South A ke joke straight
It’s really scary. I once tried 140mph nobody ask me before I reduced speed. I Fear for my life man 😂
Idk what you're saying with that but I accept it 😂
The people 😂
```

Figure 6 processed.txt, which keeps the output of process.py

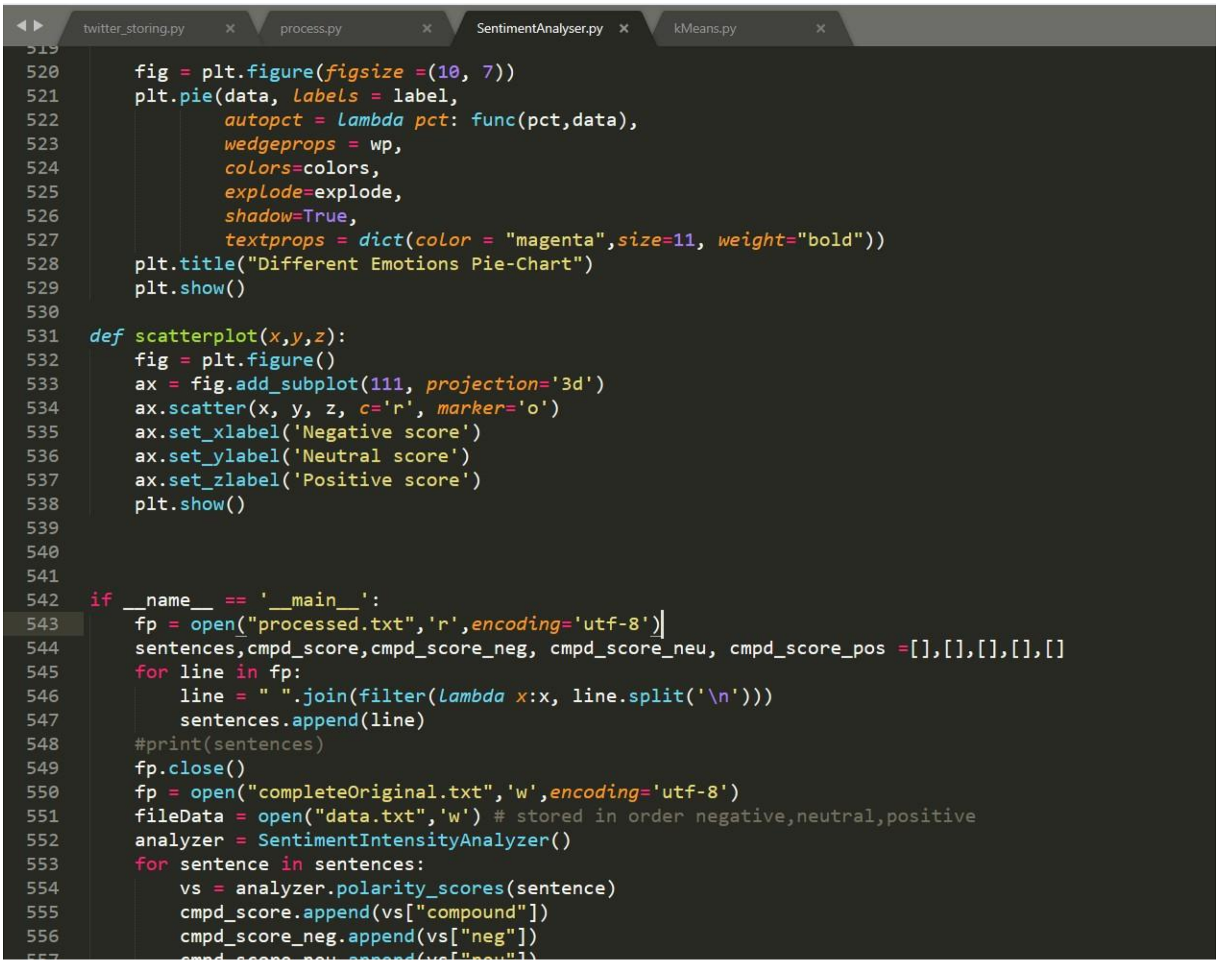


```
@3volG3nius Well at least the weather here, by that time will be back hot 😂 cuz it’s chilly
right now 😢
@hapYkiddilse Yea Sunday special babe 😂
@snuurid how tf do people even use there phone in the shower??? 😂
@RepBrianKing @DeseretNews 😂 good call
nothing like signing an offer for a house while 10,000 miles in the air 😂
#realestateiscrazy
@OMGItsKhairy_ Those pink inflatable arms spongebob ordered 😂
@MsBlueLipstick My Robby gives me GTH looks all the time. 😂 https://t.co/nxP9GkE3e3
@PhilColl_ICL @BlackpoolFC 😂 no mate, weekend away camping ⛺
@KadyDane Are you drinking it? 😂
@MichaelBucwa @Radebe_merci I bet she smell good, mmh keep taking them out, 😂 you might
score one day https://t.co/SZaAgc4c1w
all the bullshit he be doing 😂
@Lozsa9 So what you gone do with it have it sit there forever 😂
@LxAndStuff I’ll just have one chair that I take apart and make the new guy put it back
together with me 😂
@raphnelson_abah Before we go win trophy again, probably 2052 😂
LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH 😭😭😭😭😭😭😭😭
@paul_db Lol!!! Although I might be able to take you I would never 😂
Jumped out a plane yesterday tryna surf a tsunami today 😂
@Luceobrien Excuse me, the body pillow is not inflatable 😂
He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the
EU.🏴🏴♡👍
Running off 3 hours of sleep. 😢 &amp; barely that cause I tossed ‘&amp; turned all night.
But I’m thugging today 😂
Y’all @whoizkoop get drunk and it’s an instant dance battle like duuuudddeeeee 😂
@FullMetalTune When everyone though i was long gone and I just heard you getting shit talked
to and I just bust opened the front door 😂
So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning
to try and pull Kazuha… https://t.co/3d3iEDsHSB
People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody
is physically taking the ch… https://t.co/7NZDQaUgYV
“Crackhead” is a such a derogatory term. We laugh and joke on ppl like this but addiction is
a disease. This pictur… https://t.co/DBImq95hGX
*Yerk* 😂
good thing IDGAF because sometimes I'm awkward as hell 😂
@allways__always I couldn’t decide to make Jungkook an omega or an alpha but I like
Alpha/alpha or omega/omega on A/B/O 😂
@Thephoenixdemon The fact that she’s trying to say it’s not about Sasha when she literally
liked a tweet being shady towards her 😂
@Yadnemsirhc71 You get the odd tit bit lol 😂
@CamHaines @leahmrb And my dream! Although I’d rather feed them fruit 😂 lol
@Seaux504 Escape. Something light, reliable , wanna upgrade me? 😂
@hannah_c_rose Honestly I rarely even argue about it I am simply too tired 😂
```

```
Well at least the weather here, by that time will be back hot 😂 cuz it’s chilly right now 😢
Yea Sunday special babe 😂
how tf do people even use there phone in the shower??? 😂
😂 good call
nothing like signing an offer for a house while 10,000 miles in the air 😂
#realestateiscrazy
Those pink inflatable arms spongebob ordered 😂
My Robby gives me GTH looks all the time. 😂
😂 no mate, weekend away camping ⛺
Are you drinking it? 😂
I bet she smell good, mmh keep taking them out, 😂 you might score one day
all the bullshit he be doing 😂
So what you gone do with it have it sit there forever 😂
I’ll just have one chair that I take apart and make the new guy put it back together with me
😂
Before we go win trophy again, probably 2052 😂
LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH 😭😭😭😭😭😭😭😭
Lol!!! Although I might be able to take you I would never 😂
Jumped out a plane yesterday tryna surf a tsunami today 😂
Excuse me, the body pillow is not inflatable 😂
He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the
EU.🏴🏴♡👍
Running off 3 hours of sleep. 😢 &amp; barely that cause I tossed ‘&amp; turned all night.
But I’m thugging today 😂
Y’all get drunk and it’s an instant dance battle like duuuudddeeeee 😂
When everyone though i was long gone and I just heard you getting shit talked to and I just
bust opened the front door 😂
So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning
to try and pull Kazuha…
People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody
is physically taking the ch…
“Crackhead” is a such a derogatory term. We laugh and joke on ppl like this but addiction is
a disease. This pictur…
*Yerk* 😂
good thing IDGAF because sometimes I'm awkward as hell 😂
I couldn’t decide to make Jungkook an omega or an alpha but I like Alpha/alpha or
omega/omega on A/B/O 😂
The fact that she’s trying to say it’s not about Sasha when she literally liked a tweet
being shady towards her 😂
You get the odd tit bit lol 😂
And my dream! Although I’d rather feed them fruit 😂 lol
Escape. Something light, reliable , wanna upgrade me? 😂
Honestly I rarely even argue about it I am simply too tired 😂
You act like it was on purpose 😂
The more I interact with real estate agents the more I realize they are the weakest sales
```

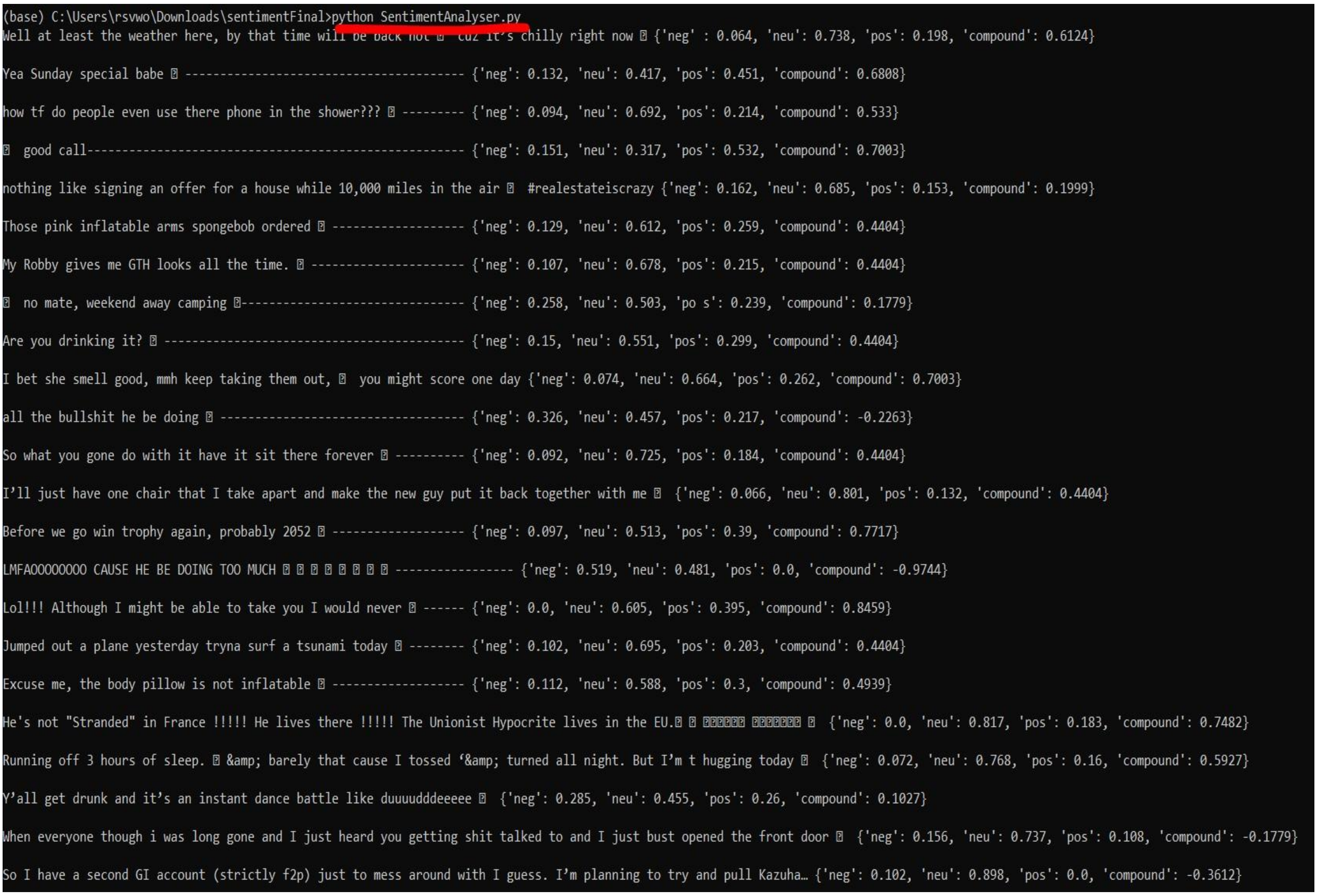
Figure 7 Comparison of store.txt and processed.txt. As it can be observed that all the username(@userName) and links(https://iamalink.com) are removed

As the data has been filtered, the SentimentAnalysis.py is executed on the processed data. The program give a positive, negative, neutral score and a overall compound score to the tweets. The program works by converting the emojis to their text description from emoji_ut8_lexicon.txt*(given in appendix section)***and then it assigns a valence score to each word by making use of vander_lexicon.txt***(given in appendix section)*** file. This gives a overall valence score to the tweets. The output is then stored into a new text file completeOriginal.txt**



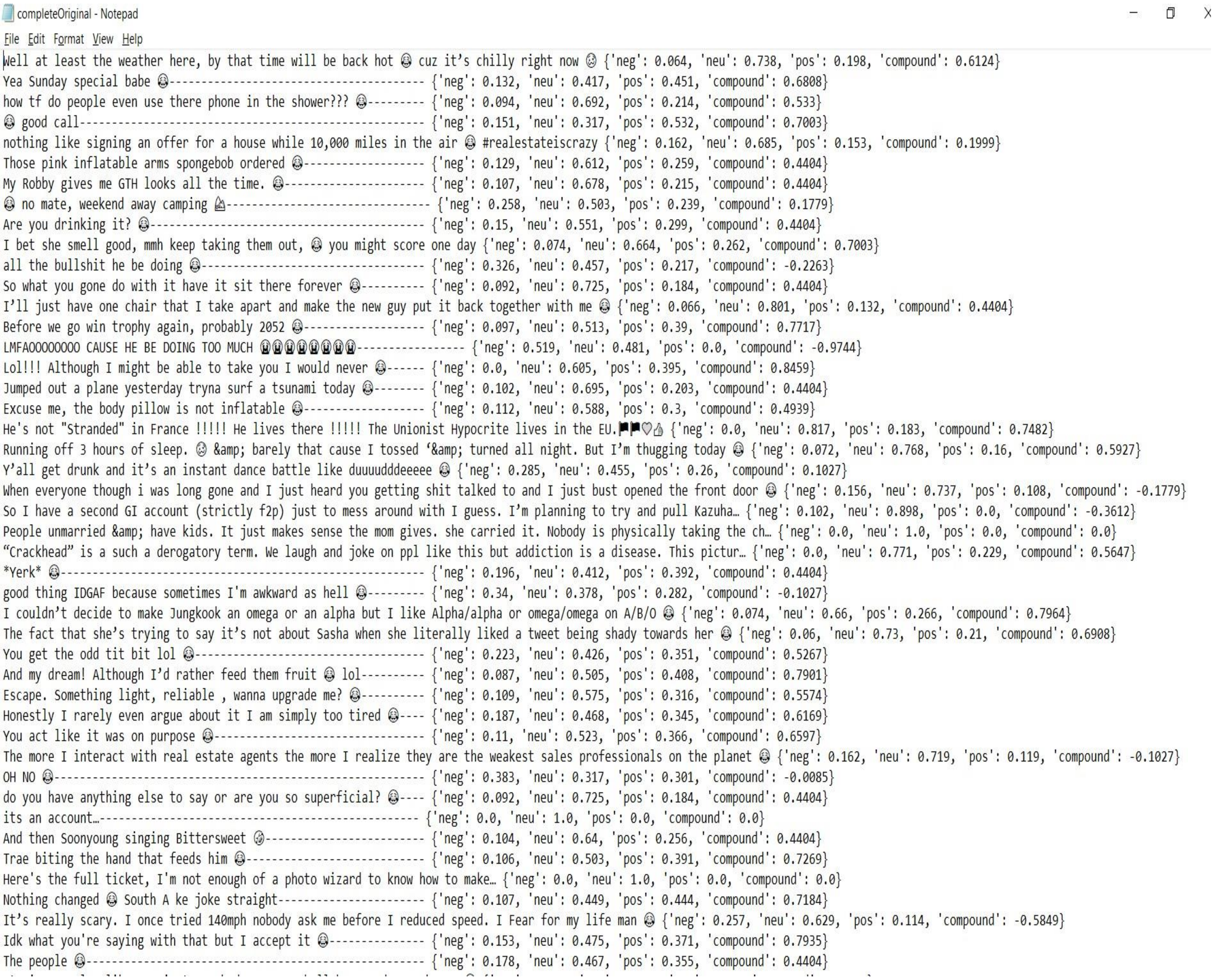
```
    fig = plt.figure(figsize =(10, 7))
    plt.pie(data, labels = label,
            autopct = lambda pct: func(pct,data),
            wedgeprops = wp,
            colors=colors,
            explode=explode,
            shadow=True,
            textprops = dict(color = "magenta",size=11, weight="bold"))
    plt.title("Different Emotions Pie-Chart")
    plt.show()

def scatterplot(x,y,z):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, z, c='r', marker='o')
    ax.set_xlabel('Negative score')
    ax.set_ylabel('Neutral score')
    ax.set_zlabel('Positive score')
    plt.show()



if __name__ == '__main__':
    fp = open("processed.txt",'r',encoding='utf-8')
    sentences,cmpd_score,cmpd_score_neg, cmpd_score_neu, cmpd_score_pos =[],[],[],[],[]
    for line in fp:
        line = " ".join(filter(lambda x:x, line.split('\n')))
        sentences.append(line)
    #print(sentences)
    fp.close()
    fp = open("completeOriginal.txt",'w',encoding='utf-8')
    fileData = open("data.txt",'w') # stored in order negative,neutral,positive
    analyzer = SentimentIntensityAnalyzer()
    for sentence in sentences:
        vs = analyzer.polarity_scores(sentence)
        cmpd_score.append(vs["compound"])
        cmpd_score_neg.append(vs["neg"])
        cmpd_score_neu.append(vs["neu"])
```

Figure 8 Snip of SentimentAnalyser.py. Complete code in appendix section



```
(base) C:\Users\rsvwo\Downloads\sentimentFinal>python SentimentAnalyser.py
Well at least the weather here, by that time will be back hot □  cuz it's chilly right now □ {'neg' : 0.064, 'neu': 0.738, 'pos': 0.198, 'compound': 0.6124}

Yea Sunday special babe □ ---------------------------------------- {'neg': 0.132, 'neu': 0.417, 'pos': 0.451, 'compound': 0.6808}

how tf do people even use there phone in the shower??? □ --------- {'neg': 0.094, 'neu': 0.692, 'pos': 0.214, 'compound': 0.533}

□  good call------------------------------------------------------ {'neg': 0.151, 'neu': 0.317, 'pos': 0.532, 'compound': 0.7003}

nothing like signing an offer for a house while 10,000 miles in the air □  #realestateiscrazy {'neg': 0.162, 'neu': 0.685, 'pos': 0.153, 'compound': 0.1999}

Those pink inflatable arms spongebob ordered □ ------------------- {'neg': 0.129, 'neu': 0.612, 'pos': 0.259, 'compound': 0.4404}

My Robby gives me GTH looks all the time. □ ---------------------- {'neg': 0.107, 'neu': 0.678, 'pos': 0.215, 'compound': 0.4404}

□  no mate, weekend away camping □-------------------------------- {'neg': 0.258, 'neu': 0.503, 'po s': 0.239, 'compound': 0.1779}

Are you drinking it? □ ------------------------------------------- {'neg': 0.15, 'neu': 0.551, 'pos': 0.299, 'compound': 0.4404}

I bet she smell good, mmh keep taking them out, □  you might score one day {'neg': 0.074, 'neu': 0.664, 'pos': 0.262, 'compound': 0.7003}

all the bullshit he be doing □ ----------------------------------- {'neg': 0.326, 'neu': 0.457, 'pos': 0.217, 'compound': -0.2263}

So what you gone do with it have it sit there forever □ ---------- {'neg': 0.092, 'neu': 0.725, 'pos': 0.184, 'compound': 0.4404}

I'll just have one chair that I take apart and make the new guy put it back together with me □  {'neg': 0.066, 'neu': 0.801, 'pos': 0.132, 'compound': 0.4404}

Before we go win trophy again, probably 2052 □ ------------------- {'neg': 0.097, 'neu': 0.513, 'pos': 0.39, 'compound': 0.7717}

LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH □ □ □ □ □ □ □ □ ----------------- {'neg': 0.519, 'neu': 0.481, 'pos': 0.0, 'compound': -0.9744}

Lol!!! Although I might be able to take you I would never □ ------ {'neg': 0.0, 'neu': 0.605, 'pos': 0.395, 'compound': 0.8459}

Jumped out a plane yesterday tryna surf a tsunami today □ -------- {'neg': 0.102, 'neu': 0.695, 'pos': 0.203, 'compound': 0.4404}

Excuse me, the body pillow is not inflatable □ ------------------- {'neg': 0.112, 'neu': 0.588, 'pos': 0.3, 'compound': 0.4939}

He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the EU.□ □ □□□□□□ □□□□□□□ □  {'neg': 0.0, 'neu': 0.817, 'pos': 0.183, 'compound': 0.7482}

Running off 3 hours of sleep. □ &amp; barely that cause I tossed '&amp; turned all night. But I'm t hugging today □  {'neg': 0.072, 'neu': 0.768, 'pos': 0.16, 'compound': 0.5927}

Y'all get drunk and it's an instant dance battle like duuuudddeeeee □  {'neg': 0.285, 'neu': 0.455, 'pos': 0.26, 'compound': 0.1027}

When everyone though i was long gone and I just heard you getting shit talked to and I just bust opened the front door □  {'neg': 0.156, 'neu': 0.737, 'pos': 0.108, 'compound': -0.1779}

So I have a second GI account (strictly f2p) just to mess around with I guess. I'm planning to try and pull Kazuha… {'neg': 0.102, 'neu': 0.898, 'pos': 0.0, 'compound': -0.3612}
```

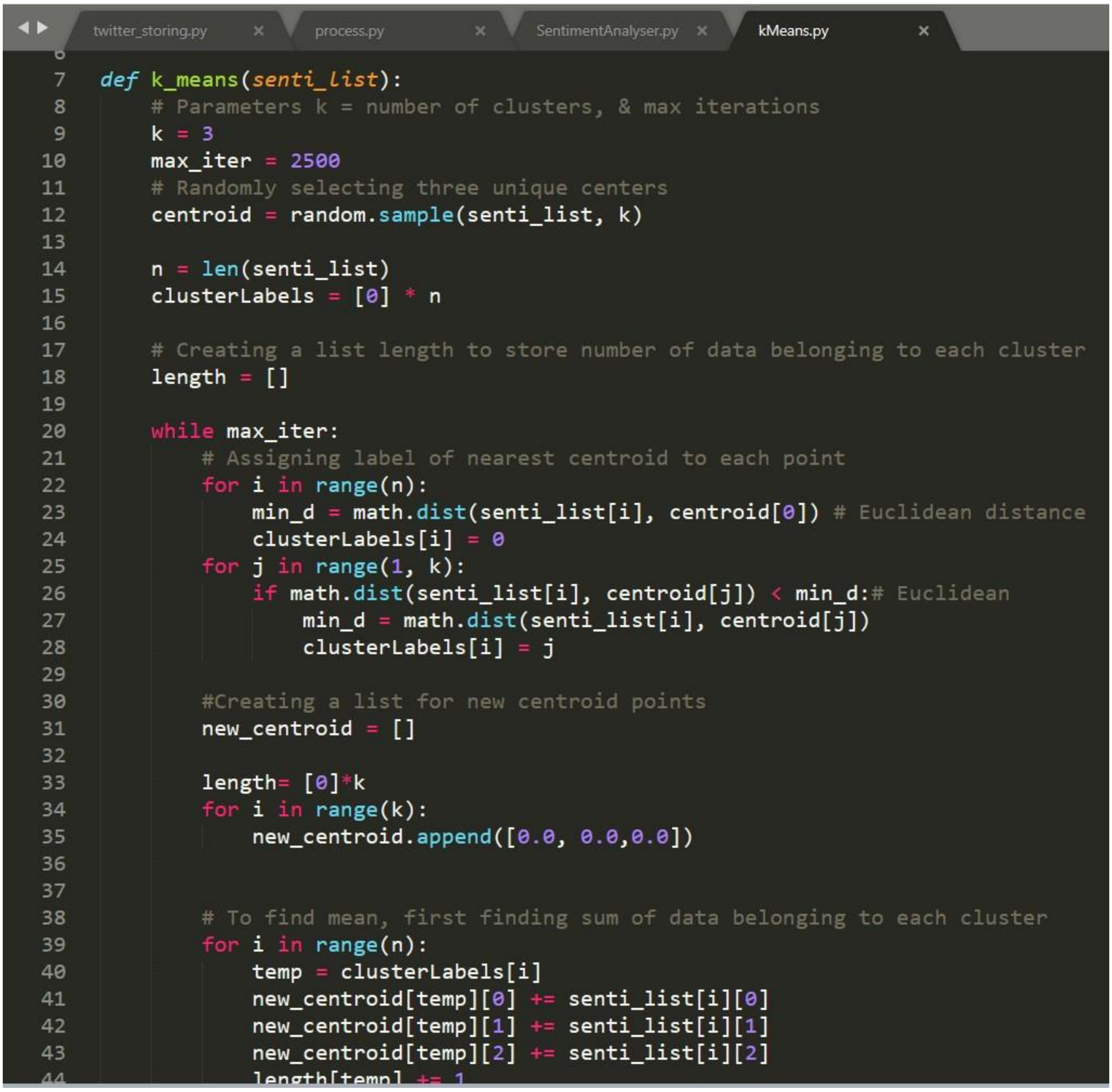
Figure 9 Implementation of SentimentAnalyser.py



```
Well at least the weather here, by that time will be back hot 😂 cuz it’s chilly right now 🥺 {'neg': 0.064, 'neu': 0.738, 'pos': 0.198, 'compound': 0.6124}
Yea Sunday special babe 😂---------------------------------------- {'neg': 0.132, 'neu': 0.417, 'pos': 0.451, 'compound': 0.6808}
how tf do people even use there phone in the shower??? 😂--------- {'neg': 0.094, 'neu': 0.692, 'pos': 0.214, 'compound': 0.533}
😂 good call------------------------------------------------------ {'neg': 0.151, 'neu': 0.317, 'pos': 0.532, 'compound': 0.7003}
nothing like signing an offer for a house while 10,000 miles in the air 😂 #realestateiscrazy {'neg': 0.162, 'neu': 0.685, 'pos': 0.153, 'compound': 0.1999}
Those pink inflatable arms spongebob ordered 😂------------------- {'neg': 0.129, 'neu': 0.612, 'pos': 0.259, 'compound': 0.4404}
My Robby gives me GTH looks all the time. 😂--------------------- {'neg': 0.107, 'neu': 0.678, 'pos': 0.215, 'compound': 0.4404}
😂 no mate, weekend away camping ⛺-------------------------------- {'neg': 0.258, 'neu': 0.503, 'pos': 0.239, 'compound': 0.1779}
Are you drinking it? 😂------------------------------------------- {'neg': 0.15, 'neu': 0.551, 'pos': 0.299, 'compound': 0.4404}
I bet she smell good, mmh keep taking them out, 😂 you might score one day {'neg': 0.074, 'neu': 0.664, 'pos': 0.262, 'compound': 0.7003}
all the bullshit he be doing 😂----------------------------------- {'neg': 0.326, 'neu': 0.457, 'pos': 0.217, 'compound': -0.2263}
So what you gone do with it have it sit there forever 😂--------- {'neg': 0.092, 'neu': 0.725, 'pos': 0.184, 'compound': 0.4404}
I’ll just have one chair that I take apart and make the new guy put it back together with me 😂 {'neg': 0.066, 'neu': 0.801, 'pos': 0.132, 'compound': 0.4404}
Before we go win trophy again, probably 2052 😂------------------ {'neg': 0.097, 'neu': 0.513, 'pos': 0.39, 'compound': 0.7717}
LMFAOOOOOOOO CAUSE HE BE DOING TOO MUCH 😭😭😭😭😭😭😭😭---------------- {'neg': 0.519, 'neu': 0.481, 'pos': 0.0, 'compound': -0.9744}
Lol!!! Although I might be able to take you I would never 😂------ {'neg': 0.0, 'neu': 0.605, 'pos': 0.395, 'compound': 0.8459}
Jumped out a plane yesterday tryna surf a tsunami today 😂-------- {'neg': 0.102, 'neu': 0.695, 'pos': 0.203, 'compound': 0.4404}
Excuse me, the body pillow is not inflatable 😂------------------ {'neg': 0.112, 'neu': 0.588, 'pos': 0.3, 'compound': 0.4939}
He's not "Stranded" in France !!!!! He lives there !!!!! The Unionist Hypocrite lives in the EU.🏴🏴♡👍 {'neg': 0.0, 'neu': 0.817, 'pos': 0.183, 'compound': 0.7482}
Running off 3 hours of sleep. 🥺 &amp; barely that cause I tossed ‘&amp; turned all night. But I’m thugging today 😂 {'neg': 0.072, 'neu': 0.768, 'pos': 0.16, 'compound': 0.5927}
Y’all get drunk and it’s an instant dance battle like duuuudddeeeee 😂 {'neg': 0.285, 'neu': 0.455, 'pos': 0.26, 'compound': 0.1027}
When everyone though i was long gone and I just heard you getting shit talked to and I just bust opened the front door 😂 {'neg': 0.156, 'neu': 0.737, 'pos': 0.108, 'compound': -0.1779}
So I have a second GI account (strictly f2p) just to mess around with I guess. I’m planning to try and pull Kazuha… {'neg': 0.102, 'neu': 0.898, 'pos': 0.0, 'compound': -0.3612}
People unmarried &amp; have kids. It just makes sense the mom gives. she carried it. Nobody is physically taking the ch… {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
“Crackhead” is a such a derogatory term. We laugh and joke on ppl like this but addiction is a disease. This pictur… {'neg': 0.0, 'neu': 0.771, 'pos': 0.229, 'compound': 0.5647}
*Yerk* 😂--------------------------------------------------------- {'neg': 0.196, 'neu': 0.412, 'pos': 0.392, 'compound': 0.4404}
good thing IDGAF because sometimes I'm awkward as hell 😂--------- {'neg': 0.34, 'neu': 0.378, 'pos': 0.282, 'compound': -0.1027}
I couldn’t decide to make Jungkook an omega or an alpha but I like Alpha/alpha or omega/omega on A/B/O 😂 {'neg': 0.074, 'neu': 0.66, 'pos': 0.266, 'compound': 0.7964}
The fact that she’s trying to say it’s not about Sasha when she literally liked a tweet being shady towards her 😂 {'neg': 0.06, 'neu': 0.73, 'pos': 0.21, 'compound': 0.6908}
You get the odd tit bit lol 😂------------------------------------ {'neg': 0.223, 'neu': 0.426, 'pos': 0.351, 'compound': 0.5267}
And my dream! Although I’d rather feed them fruit 😂 lol---------- {'neg': 0.087, 'neu': 0.505, 'pos': 0.408, 'compound': 0.7901}
Escape. Something light, reliable , wanna upgrade me? 😂---------- {'neg': 0.109, 'neu': 0.575, 'pos': 0.316, 'compound': 0.5574}
Honestly I rarely even argue about it I am simply too tired 😂---- {'neg': 0.187, 'neu': 0.468, 'pos': 0.345, 'compound': 0.6169}
You act like it was on purpose 😂--------------------------------- {'neg': 0.11, 'neu': 0.523, 'pos': 0.366, 'compound': 0.6597}
The more I interact with real estate agents the more I realize they are the weakest sales professionals on the planet 😂 {'neg': 0.162, 'neu': 0.719, 'pos': 0.119, 'compound': -0.1027}
OH NO 😂---------------------------------------------------------- {'neg': 0.383, 'neu': 0.317, 'pos': 0.301, 'compound': -0.0085}
do you have anything else to say or are you so superficial? 😂---- {'neg': 0.092, 'neu': 0.725, 'pos': 0.184, 'compound': 0.4404}
its an account…--------------------------------------------------- {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
And then Soonyoung singing Bittersweet 🤣------------------------- {'neg': 0.104, 'neu': 0.64, 'pos': 0.256, 'compound': 0.4404}
Trae biting the hand that feeds him 😂---------------------------- {'neg': 0.106, 'neu': 0.503, 'pos': 0.391, 'compound': 0.7269}
Here's the full ticket, I'm not enough of a photo wizard to know how to make… {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Nothing changed 😂 South A ke joke straight----------------------- {'neg': 0.107, 'neu': 0.449, 'pos': 0.444, 'compound': 0.7184}
It’s really scary. I once tried 140mph nobody ask me before I reduced speed. I Fear for my life man 😂 {'neg': 0.257, 'neu': 0.629, 'pos': 0.114, 'compound': -0.5849}
Idk what you're saying with that but I accept it 😂--------------- {'neg': 0.153, 'neu': 0.475, 'pos': 0.371, 'compound': 0.7935}
The people 😂----------------------------------------------------- {'neg': 0.178, 'neu': 0.467, 'pos': 0.355, 'compound': 0.4404}
```

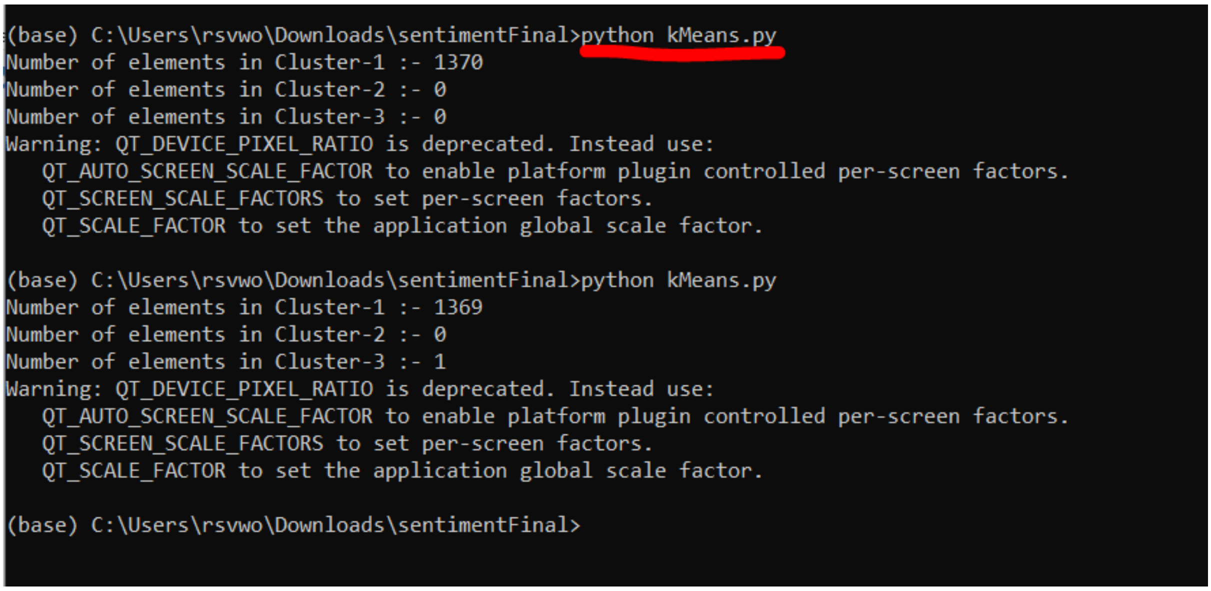
Figure 10 completeOriginal.txt, with assigned negative, neutral, positive and compound score.

To improve the model, the tweets are then clustered into three groups for better accuracy using the sentiment score we got in the previous step. To achieve this we use the K-Means unsupervised machine learning algorithm. The python file for the same is kMeans.py



```
def k_means(senti_list):
    # Parameters k = number of clusters, & max iterations
    k = 3
    max_iter = 2500
    # Randomly selecting three unique centers
    centroid = random.sample(senti_list, k)

    n = len(senti_list)
    clusterLabels = [0] * n

    # Creating a list length to store number of data belonging to each cluster
    length = []

    while max_iter:
        # Assigning label of nearest centroid to each point
        for i in range(n):
            min_d = math.dist(senti_list[i], centroid[0]) # Euclidean distance
            clusterLabels[i] = 0
        for j in range(1, k):
            if math.dist(senti_list[i], centroid[j]) < min_d:# Euclidean
                min_d = math.dist(senti_list[i], centroid[j])
                clusterLabels[i] = j

        #Creating a list for new centroid points
        new_centroid = []

        length= [0]*k
        for i in range(k):
            new_centroid.append([0.0, 0.0,0.0])


        # To find mean, first finding sum of data belonging to each cluster
        for i in range(n):
            temp = clusterLabels[i]
            new_centroid[temp][0] += senti_list[i][0]
            new_centroid[temp][1] += senti_list[i][1]
            new_centroid[temp][2] += senti_list[i][2]
            length[temp] += 1
```

Figure 11 Snip of kMeans.py. Complete code in appendix section



```
(base) C:\Users\rsvwo\Downloads\sentimentFinal>python kMeans.py
Number of elements in Cluster-1 :- 1370
Number of elements in Cluster-2 :- 0
Number of elements in Cluster-3 :- 0
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.

(base) C:\Users\rsvwo\Downloads\sentimentFinal>python kMeans.py
Number of elements in Cluster-1 :- 1369
Number of elements in Cluster-2 :- 0
Number of elements in Cluster-3 :- 1
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.

(base) C:\Users\rsvwo\Downloads\sentimentFinal>
```

Figure 12 Implementation of kMeans.py

VII RESULT AND DISCUSSION

Table 1 Sentiments polarity after applying the Sentiment analysis

Type of Emotion	Number	Percentage	High Value	Low Value
Positive	1053	76.8%	1.0	0.0
Negative	196	14.3%	1.0	0.0
Neutral	123	8.9%	1.0	0.0
Total	1372			

Table 2 Sentiments polarity after applying K-Means clustering Algorithm

Cluster	Number	Percentage	High Value	Low Value
Cluster-1	1370	99.9%	1.0	0.0
Cluster-2	0	0.0%	1.0	0.0
Cluster-3	2	0.1%	1.0	0.0
Total	1372			

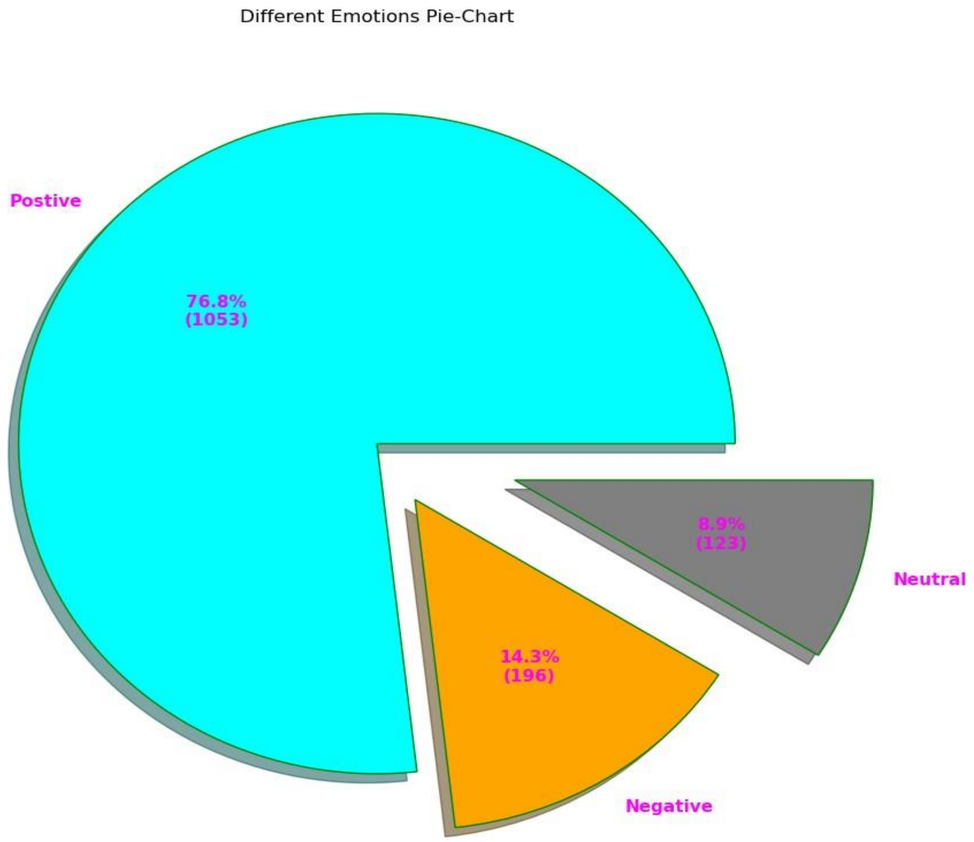


Figure 13 Pie-chart after applying Sentiment Analysis

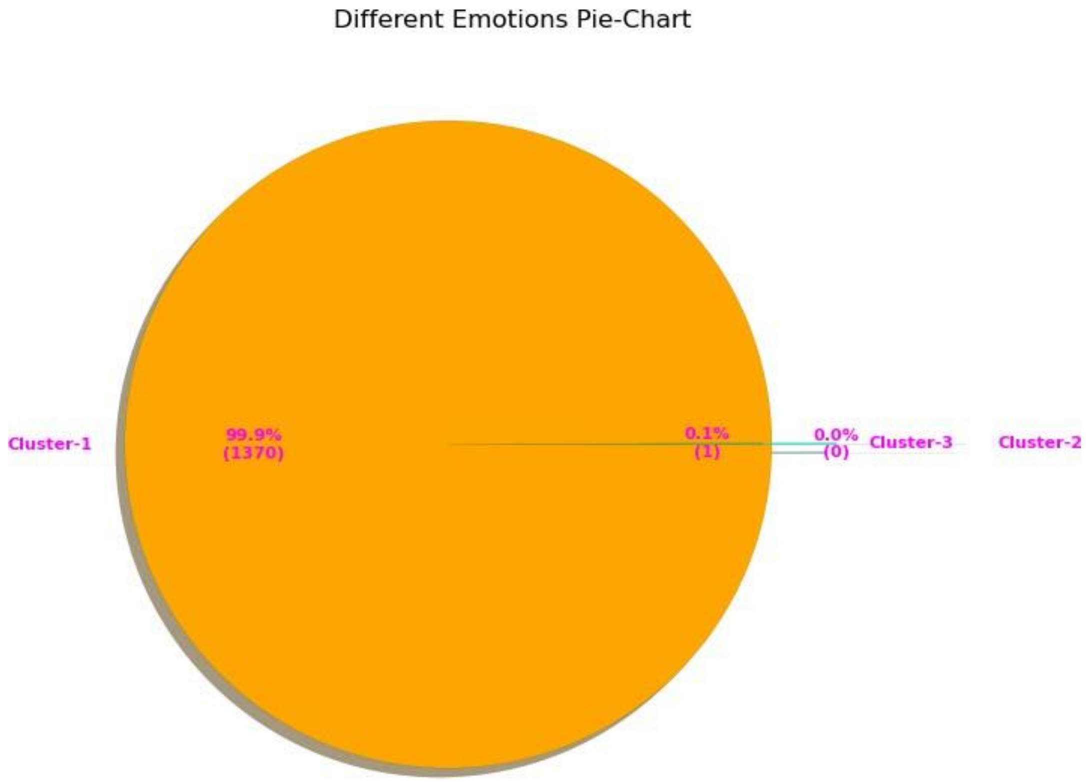


Figure 14Pie-chart for Sentiments polarity after applying K-Means clustering Algorithm

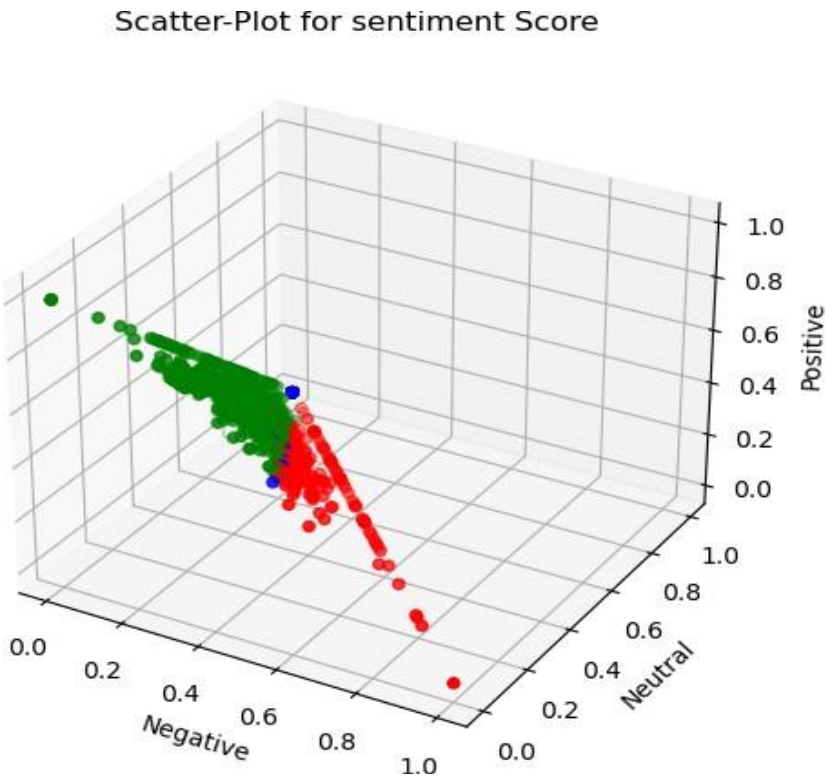


Figure 15Scatte-Plot Sentiments polarity after applying the Sentiment analysis

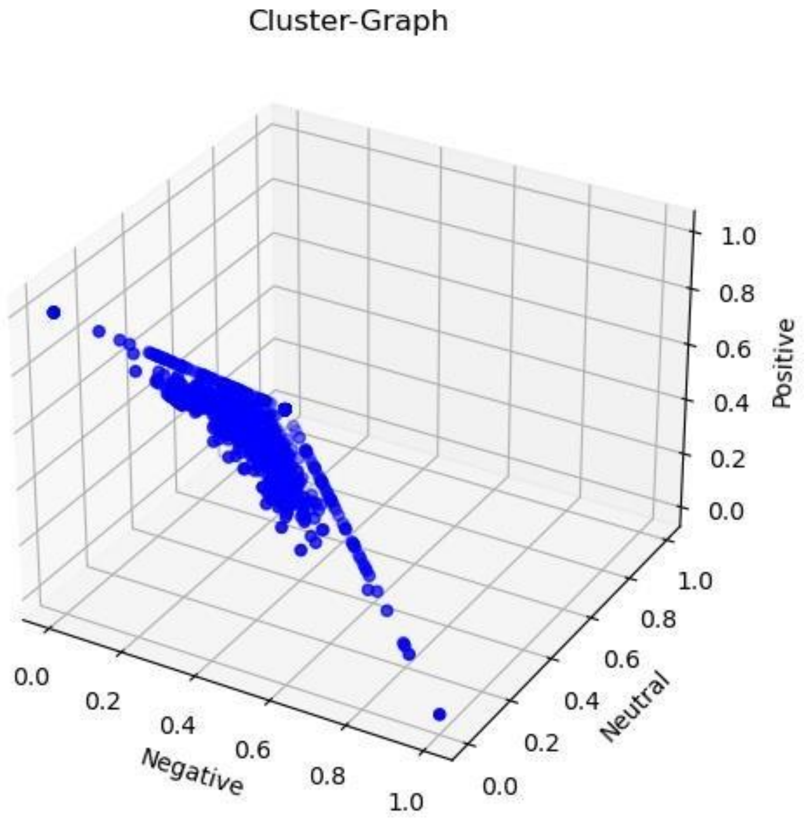


Figure 16 Scatter plot after clustering the sentiments

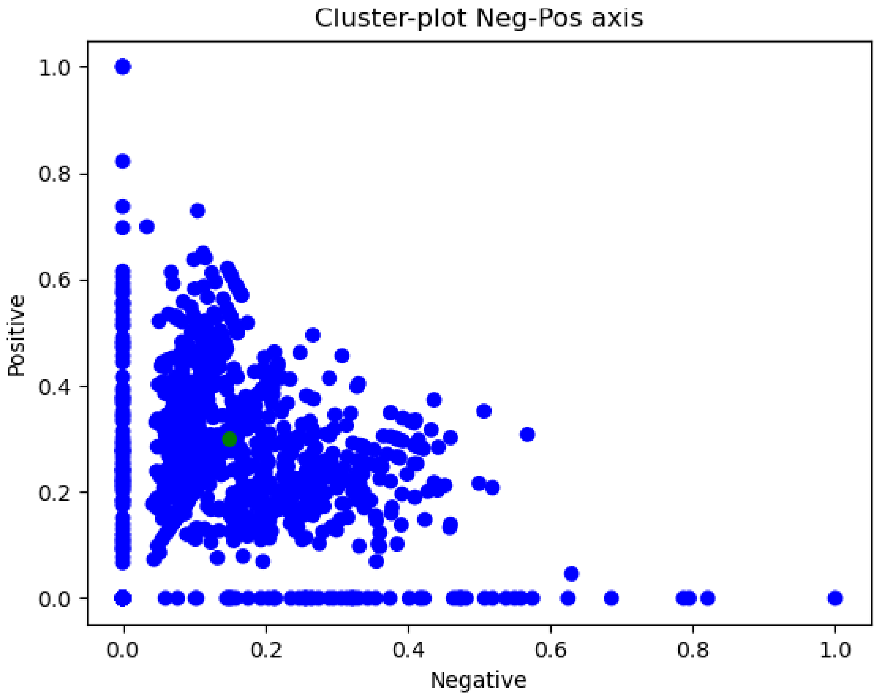


Figure 17 3-D cluster graph on 2-D plane along Negative and Positive axes

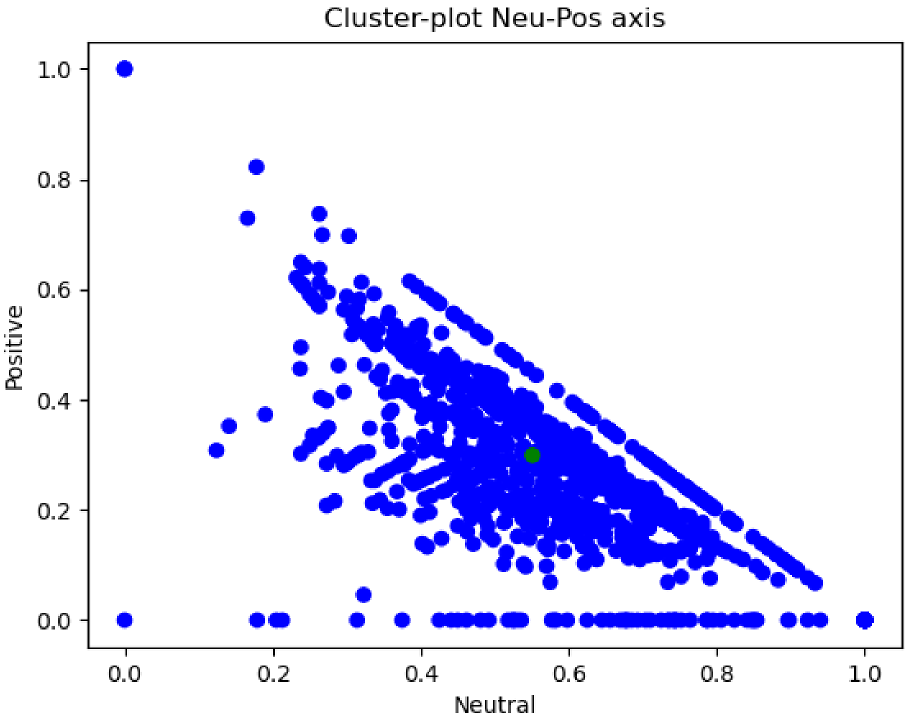


Figure 18 3-D cluster graph on 2-D plane along Positive and Neutral axes

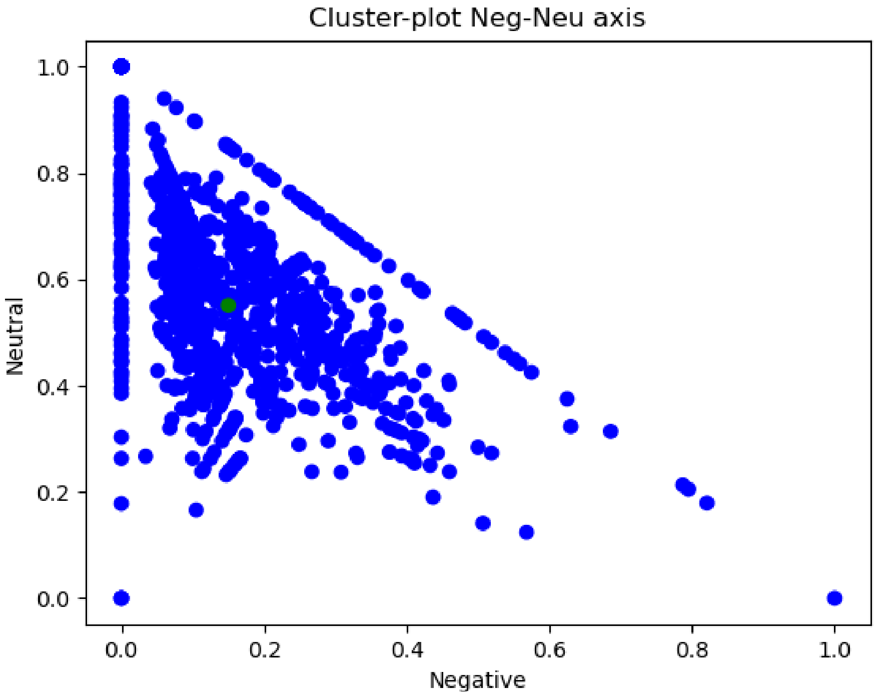


Figure 19 3-D cluster graph on 2-D plane along Negative and Neutral axes

For the better modal which can further predict different types of sentiments, we will need a labeled user data which can be used to determine the accuracy of the model. As the tweets extracted from the twitter are not marked its not simple to give a good analysis of the model.

This what makes analyzing sentiment not an easy task. As the data has to be first marked by a Human user, to determine the modal accuracy. And from the small marked data which is used to determine the model accuracy it was concluded that only Analysis of emoticons does not give the subtle idea of the sentiment. Because the same emoticon can be used with different texts of different meaning. Hence the model for later extended to incorporate texts and emojis along with emoticons.

VIII CONCLUSION

1. The model largely agrees with the results proposed in [1] and [2]. The emoticons may be hard to analyse, however serve as a strong means to derive the emotion conveyed in the tweet.

2. Most algorithms rely on the bag of words technique and then correlating a word or phrase to a certain emotion/sentiment, weighing them and thereby predicting the most dominant emotion. Our algorithm extends the idea of bag of words to include emoticons given the fact that human speech does not necessarily convey what it conveys literally.

3. The algorithm proposed is simpler than the ones proposed in the survey literature – needs lesser hardware capabilities while the only extra tool used for the research is the survey.

Analysis of Emoticons can give the vague idea of public view about a topic. But use of just the emoticon doesn't provide precise view about public opinion on a specific issue. To enhance the model both text and emoticons analysis have been combined into a single model.

IX REFERENCES

[1] Jayanthi T, Sreeja K, "*Sentiment Analysis of Twitter Data containing Emoticons: A Survey*," International Journal of Innovations in Engineering and Tech, 2016

[2] Hogenboom, A et al. (2013) "*Exploiting Emoticons in Sentiment Analysis*" 28th Annual ACM Symposium on Applied Computing, SAC, 2013

[3] Y. Tao, X. Zhang, L. Shi, L. Wei, Z. Hai and J. A. Wahid, "*Joint Embedding of Emoticons and Labels Based on CNN for Microblog Sentiment Analysis,*" IEEE Fourth International Conference on Data Science in Cyberspace (DSC), 2019

[4] R. Agarwal and S. Shreshth, "Sentiment Analysis of Emoticon Based Neuro Fuzzy System," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bengaluru, India, 2020, pp. 1-6, doi: 10.1109/INOCON50539.2020.9298351.

[5] Pang, B et al.,"*Thumbs up? Sentiment Classification using Machine Learning Techniques*", Proceedings of the Conference on Empirical Methods in Natural, Language Processing (EMNLP), Philadelphia, (Association for Computational Linguistics.).

[6] Das, S. R., & Chen, M. Y. "*Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web"*

[7] D. K. Zala, and A. Gandhi, "*A Twitter Based Opinion Mining to Perform Analysis Geographically,*" 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019

[8] Turney, P. D., *"Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews"* (Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)

[9] A. A. Arifiyanti and E. D. Wahyuni, "Emoji and Emoticon in Tweet Sentiment Classification," 2020 6th Information Technology International Seminar (ITIS), Surabaya, Indonesia, 2020, pp. 145-150, doi: 10.1109/ITIS50118.2020.9320988.

[10] J. S. Yang and K. S. Chung, "Newly-Coined Words and Emoticon Polarity for Social Emotional Opinion Decision," 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), Kahului, HI, USA, 2019, pp. 76-79, doi: 10.1109/INFOCT.2019.8711413.

APPENDIX:

Complete Code

twitter_storing.py

```
from tweepy import
StreamListener from tweepy
import OAuthHandler from
tweepy import Stream
from tweepy import API
from tweepy import
Cursor

#Custom stream listener class to collect relevant
details class StreamListener(StreamListener):
  def on_status(self, status):
    if hasattr(status, 'retweeted_status'): # if the tweet is a retweeted tweet it
        return                      #will give the control back to
        stream.fliter
    print(status.text+"\n")
    fp.write(status.text+"\n")
  def on_error(self, status_code):
    if status_code == 420:
      return False

#API keys
ckey = '~~######################HJYjVs2sD9#########~~'
csec =
'~~#######################VvQBnLyEl7g############~~
#' atok =
'~~#######################ocuZF7sLUpDBflKip#######~~'
asec = '~~######################35qGWNRKdZ3J###'~~

#Auth
auth = OAuthHandler(ckey,
csec)
auth.set_access_token(atok,
asec) api = API(auth)

# \U0001F631 refers to 'confused'
#1F602 = happy, 1F62D = sad, 1F621= angry, 2764 = love, 1F61C = playful, 1F631
= confused query = [u'\U0001F602']
```

```
fp = open("store.txt",'w',encoding='utf-8')
stream = Stream(auth = api.auth, listener = StreamListener())
stream.filter(track = query, languages = ["en"], stall_warnings = True)
print("@@ @ @ @ @@ @ @ @ @ @ @@ @ @ @ @ @ @@ @ @ @ @ @@ @ @
@")
```

process.py

```
import re

fp = open("store.txt",'r',encoding='utf-8')
processedFile =
open("processed.txt",'w',encoding='utf-8')  emojiFile =
open("emojiFile.txt",'w',encoding='utf-8')
for line in fp:
   line = " ".join(filter(lambda  x:x[0]!='@', line.split()))
   line = " ".join(filter(lambda  x:x[0:8]!="https://", line.split()))
   #line = " ".join(filter(lambda  x:x,
   line.split(','))) if len(line)<2:
            continue
   processedFile.write(line+"\n")
   print(line)
   emojis = re.findall(r'[^\w\s,.  ]',
   line) emojis = " ".join(emojis)
   '''lst =[]
   if (len(emojis)<2):
            continue
   for emoji in emojis:
            if emoji in ['#','@','!',':','\'','\"','\\',',','?','<','>','.',';','-','&','/']:
                        continu
            e
            lst.append(emoj
            i)
   lst = " ".join(lst)
   emojiFile.write(lst+"\n")'''
   emojiFile.write(emojis+"\n")
fp.close()
processedFile.close()
emojiFile.close()
```

SentimentAnalyser.py

```
import os
import math
import string
import
codecs
import json
from io import
open import
numpy as np
from itertools import product
from inspect import
getsourcefile
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
# sentiments intensity rating increase/decrease for booster
words. bIncr = 0.293
bDecr = -0.293

# sentiment intensity increases for ALL Caps
words cIncr = 0.733
nScalar = -0.74

#
http://en.wiktionary.org/wiki/Category:English_degree_adver
bs negate = \
   ["aint", "arent", "cannot", "cant", "couldnt", "darent", "didnt", "doesnt",
    "ain't", "aren't", "can't", "couldn't", "daren't", "didn't", "doesn't",
    "dont", "hadnt", "hasnt", "havent", "isnt", "mightnt", "mustnt", "neither",
    "don't", "hadn't", "hasn't", "haven't", "isn't", "mightn't", "mustn't",
    "neednt", "needn't", "never", "none", "nope", "nor", "not", "nothing", "nowhere",
    "oughtnt", "shant", "shouldnt", "uhuh", "wasnt", "werent",
    "oughtn't", "shan't", "shouldn't", "uh-uh", "wasn't", "weren't",
    "without", "wont", "wouldnt", "won't", "wouldn't", "rarely", "seldom", "despite"]




boosterDict = \
   {"absolutely": bIncr, "amazingly": bIncr, "awfully": bIncr,
    "completely": bIncr, "considerable": bIncr, "considerably":
    bIncr,
    "decidedly": bIncr, "deeply": bIncr, "effing": bIncr, "enormous": bIncr, "enormously":
    bIncr, "entirely": bIncr, "especially": bIncr, "exceptional": bIncr, "exceptionally":
    bIncr, "extreme": bIncr, "extremely": bIncr,
    "fabulously": bIncr, "flipping": bIncr, "flippin": bIncr, "frackin": bIncr, "fracking":
    bIncr, "fricking": bIncr, "frickin": bIncr, "frigging": bIncr, "friggin": bIncr, "fully":
    bIncr, "fuckin": bIncr, "fucking": bIncr, "fuggin": bIncr, "fugging": bIncr,
    "greatly": bIncr, "hella": bIncr, "highly": bIncr, "hugely":
    bIncr, "incredible": bIncr, "incredibly": bIncr, "intensely":
    bIncr,
    "major": bIncr, "majorly": bIncr, "more": bIncr, "most": bIncr, "particularly": bIncr,
    "purely": bIncr, "quite": bIncr, "really": bIncr, "remarkably": bIncr,
    "so": bIncr, "substantially": bIncr,
    "thoroughly": bIncr, "total": bIncr, "totally": bIncr, "tremendous": bIncr, "tremendously":
    bIncr, "uber": bIncr, "unbelievably": bIncr, "unusually": bIncr, "utter": bIncr, "utterly": bIncr,
    "very": bIncr,
    "almost": bDecr, "barely": bDecr, "hardly": bDecr, "just enough":
    bDecr, "kind of": bDecr, "kinda": bDecr, "kindof": bDecr, "kind-of":
    bDecr, "less": bDecr, "little": bDecr, "marginal": bDecr,
    "marginally": bDecr, "occasional": bDecr, "occasionally": bDecr,
    "partly": bDecr,
    "scarce": bDecr, "scarcely": bDecr, "slight": bDecr, "slightly": bDecr, "somewhat":
    bDecr, "sort of": bDecr, "sorta": bDecr, "sortof": bDecr, "sort-of": bDecr}

# some commonly used idoms
sentimentLadenIdioms = {"cut the mustard": 2, "hand to mouth": -2,
                    "back handed": -2, "blow smoke": -2, "blowing
                    smoke": -2, "upper hand": 1, "break a leg": 2,
                    "cooking with gas": 2, "in the black": 2, "in the red":
                    -2, "on the ball": 2, "under the weather": -2}

# check for special case idioms and phrases containing lexicon words
specialCases = {"the shit": 3, "the bomb": 3, "bad ass": 1.5, "badass": 1.5, "bus
             stop": 0.0, "yeah right": -2, "kiss of death": -1.5, "to die for": 3,
             "beating heart": 3.1, "broken heart": -2.9 }
```

```
def negated(inputWords, includeNt=True):
    """
    Determine if input contains negation
    words """
    inputWords = [str(w).lower() for w in
    inputWords] negWords = []
    negWords.extend(negate)
    for word in negWords:
        if word in inputWords:
            return True
    if includeNt: # shouldn't, couldn't; all the n't
        words for word in inputWords:
            if "n't" in word:
                return
    True return
    False


def normalize(score, alpha=15):
    """
    Normalize the score to be between -1 and 1 using an
    alpha that approximates the max expected value
    """
    normScore = score / math.sqrt((score * score) +
    alpha) if normScore < -1.0:
        return -1.0
    elif normScore > 1.0:
        return
    1.0 else:
        return normScore


def allCapitalDifferent(words):
    """
    Check whether just some words in the input are ALL CAPS
    :returns: `True` if some but not all items in `words` are ALL
    CAPS """
    allCapitalWords =
    0 for word in
    words:
        if word.isupper():
            allCapitalWords += 1
    cap_differential = len(words) -
    allCapitalWords if 0 < cap_differential <
    len(words):
        return
    True else:
        return False


def scalarIncDec(word, valence, isCapitalDiff):
    """
    Check if the preceding words increase, decrease, or
    negate/nullify the valence
    """
    scalar = 0.0
```

```
  wordLower =
  word.lower() if wordLower
  in boosterDict:
    scalar =
    boosterDict[wordLower] if
    valence < 0:
      scalar *= -1

    if word.isupper() and isCapitalDiff:
      if valence > 0:
        scalar +=
      cIncr else:
        scalar -=
  cIncr return scalar


class SentiText(object):
  """
  Identify sentiment-relevant string-level properties of input
  text. """

  def_init_(self, text):
    if not isinstance(text, str):
      text = str(text).encode('utf-
    8') self.text = text
    self.wordsAndEmoticons = self._wordsAndEmoticons()
    self.isCapitalDiff = allCapitalDifferent(self.wordsAndEmoticons) #Check whether some words are CAPS

  @staticmethod
  def _StripPuncInWord(token):
    """
    Removes all trailing and leading punctuation
    If the resulting string has two or fewer characters,
    then it was likely an emoticon, so return original
    string (ie ":)" stripped would be "", so just return
    ":)"
    """
    stripped =
    token.strip(string.punctuation) if
    len(stripped) <= 2:
      return token
    return
    stripped

  def _wordsAndEmoticons(self):
    """
    Removes leading and trailing
    puncutation Leaves contractions and
    most emoticons
      Does not preserve punc-plus-letter emoticons
    (e.g. :D) """
    wes = self.text.split()
    stripped = list(map(self._StripPuncInWord,
    wes)) return stripped

class SentimentIntensityAnalyzer(object):
  """
  Give a sentiment intensity score to
  sentences. """

  def_init_(self, lexiconFile="vader_lexicon.txt", emojiLexicon="emoji_utf8_lexicon.txt"):
```

```
_thisModuleFilePath_  = os.path.abspath(getsourcefile(lambda:  0))
lexiconFullFilePath  = os.path.join(os.path.dirname(_thisModuleFilePath_),  lexiconFile)
```

```
    with codecs.open(lexiconFullFilePath,  encoding='utf-8')
        as f: self.lexiconFullFilePath  = f.read()
    self.lexicon = self.makeLexiconDictionary()

    emojiFilePath  = os.path.join(os.path.dirname(_thisModuleFilePath_),
    emojiLexicon)  with codecs.open(emojiFilePath,  encoding='utf-8')  as f:
        self.emojiFilePath  = f.read()
    self.emojis =
    self.makeEmojisDictionary()

def makeLexiconDictionary(self):
    """
    Convert  lexicon file to a
    dictionary  """
    lexiconDict = {}
    for line in
        self.lexiconFullFilePath.rstrip('\n').split('\n'):   if
        not line:
            continue
        (word,  measure)  =
        line.strip().split('\t')[0:2]
        lexiconDict[word]  = float(measure)
    return  lexiconDict

def makeEmojisDictionary(self):
    """
    Convert  emoji lexicon file to a
    dictionary  """
    EmojisDict = {}
    for line in
        self.emojiFilePath.rstrip('\n').split('\n'):
        (emoji,  description)  =
        line.strip().split('\t')[0:2]
        EmojisDict[emoji]  =
    description  return  EmojisDict

def polarityScore(self,  text):
    """
    Return  a float for sentiment strength based on the input
    text. Positive  values  are positive  valence,  negative  value
    are negative  valence.
    """

    textAndEmoji   =
    ""   prevSpace   =
    True   for   chr   in
    text:
        if chr in self.emojis:

            description =
            self.emojis[chr]  if not
            prevSpace:
                textAndEmoji += ' '
            textAndEmoji +=
            description  prevSpace   =
            False
        else:
            textAndEmoji += chr
            prevSpace  = chr == ' '
    text =
textAndEmoji.strip()

sentitext =

SentiText(text)

sentiments = []
wordsAndEmoticons  = sentitext.wordsAndEmoticons
```

```
    for i, item in enumerate(wordsAndEmoticons):
      valence = 0

      if item.lower() in boosterDict:
        sentiments.append(valence
        ) continue
      if (i < len(wordsAndEmoticons) - 1 and item.lower() ==
          "kind" and wordsAndEmoticons[i + 1].lower() == "of"):
        sentiments.append(valence
        ) continue

      sentiments = self.sentimentValence(valence, sentitext, item, i,

    sentiments) sentiments = self._checkBut(wordsAndEmoticons,

    sentiments)

    valenceDict = self.valenceScore(sentiments,

    text) return valenceDict

  def sentimentValence(self, valence, sentitext, item, i, sentiments):
    isCapitalDiff = sentitext.isCapitalDiff
    wordsAndEmoticons =
    sentitext.wordsAndEmoticons itemLowerCase =
    item.lower()
    if itemLowerCase in self.lexicon:

      valence = self.lexicon[itemLowerCase]


      if itemLowerCase == "no" and i != len(wordsAndEmoticons)-1 and wordsAndEmoticons[i +
1].lower() in self.lexicon:

        valence = 0.0
      if (i > 0 and wordsAndEmoticons[i - 1].lower() ==
       "no") \ or (i > 1 and wordsAndEmoticons[i -
       2].lower() == "no") \
       or (i > 2 and wordsAndEmoticons[i - 3].lower() == "no" and wordsAndEmoticons[i - 1].lower() in ["or",
"nor"] ):
        valence = self.lexicon[itemLowerCase] * nScalar


      if item.isupper() and isCapitalDiff:
        if valence > 0:
          valence +=
        cIncr else:
          valence -= cIncr

      for startI in range(0,

      3):



        if i > startI and wordsAndEmoticons[i - (startI + 1)].lower() not in
          self.lexicon: s = scalarIncDec(wordsAndEmoticons[i - (startI + 1)],
          valence, isCapitalDiff) if startI == 1 and s != 0:
            s = s * 0.95
          if startI == 2 and s != 0:
            s = s * 0.9
```

```
valence  = valence  + s
valence  = self._negationCheck(valence,   wordsAndEmoticons,  startI, i)
```

```
        if startI == 2:
          valence = self._specialIdiomsCheck(valence, wordsAndEmoticons, i)

    valence = self._checkLeast(valence,
  wordsAndEmoticons, i) sentiments.append(valence)
  return sentiments

def _checkLeast(self, valence, wordsAndEmoticons, i):

  if i > 1 and wordsAndEmoticons[i - 1].lower() not in
      self.lexicon \ and wordsAndEmoticons[i - 1].lower() ==
      "least":
    if wordsAndEmoticons[i - 2].lower() != "at" and wordsAndEmoticons[i - 2].lower() !=
      "very": valence = valence * nScalar
  elif i > 0 and wordsAndEmoticons[i - 1].lower() not in
      self.lexicon \ and wordsAndEmoticons[i - 1].lower() ==
      "least":
    valence = valence *
  nScalar return valence

@staticmethod
def _checkBut(wordsAndEmoticons, sentiments):

  wordsAndEmoticonsLower = [str(w).lower() for w in
  wordsAndEmoticons] if 'but' in wordsAndEmoticonsLower:
    bi =
    wordsAndEmoticonsLower.index('but')
    for sentiment in sentiments:
      si =
      sentiments.index(sentiment) if
      si < bi:
        sentiments.pop(si)
        sentiments.insert(si, sentiment *
        0.5)
      elif si > bi:
        sentiments.pop(si)
        sentiments.insert(si, sentiment *
        1.5)
  return sentiments

@staticmethod
def _specialIdiomsCheck(valence, wordsAndEmoticons, i):
  wordsAndEmoticonsLower = [str(w).lower() for w in
  wordsAndEmoticons]
  onezero = "{0} {1}".format(wordsAndEmoticonsLower[i - 1], wordsAndEmoticonsLower[i])

  twoonezero = "{0} {1} {2}".format(wordsAndEmoticonsLower[i - 2],
                   wordsAndEmoticonsLower[i - 1],
                   wordsAndEmoticonsLower[i])

  twoone = "{0} {1}".format(wordsAndEmoticonsLower[i - 2], wordsAndEmoticonsLower[i - 1])

  threetwoone = "{0} {1} {2}".format(wordsAndEmoticonsLower[i - 3],
                 wordsAndEmoticonsLower[i - 2], wordsAndEmoticonsLower[i - 1])

  threetwo = "{0} {1}".format(wordsAndEmoticonsLower[i - 3],

  wordsAndEmoticonsLower[i - 2]) sequences = [onezero, twoonezero, twoone,

  threetwoone, threetwo]
```

```
for seq in sequences:
   if seq in specialCases:
      valence =
      specialCases[seq] break

if len(wordsAndEmoticonsLower)  - 1 > i:
      zeroone = "{0} {1}".format(wordsAndEmoticonsLower[i],
      wordsAndEmoticonsLower[i + 1]) if zeroone in specialCases:
        valence = specialCases[zeroone]
    if len(wordsAndEmoticonsLower) - 1 > i + 1:
      zeroonetwo = "{0} {1} {2}".format(wordsAndEmoticonsLower[i], wordsAndEmoticonsLower[i + 1],
                             wordsAndEmoticonsLower[i +
      2]) if zeroonetwo in specialCases:
        valence = specialCases[zeroonetwo]


    nGrams = [threetwoone, threetwo,
    twoone] for nGram in nGrams:
      if nGram in boosterDict:
        valence = valence +
    boosterDict[nGram] return valence

  @staticmethod
  def _sentimentLadenIdiomsCheck(valence, sentimentTextLower):


    idiomsValences = []
    for idiom in sentimentLadenIdioms:
      if idiom in sentimentTextLower:
        print(idiom, sentimentTextLower)
        valence =
        sentimentLadenIdioms[idiom]
        idiomsValences.append(valence)
    if len(idiomsValences) > 0:
      valence = sum(idiomsValences) /
    float(len(idiomsValences)) return valence

  @staticmethod
  def _negationCheck(valence, wordsAndEmoticons, startI, i):
    wordsAndEmoticonsLower = [str(w).lower() for w in
    wordsAndEmoticons] if startI == 0:
      if negated([wordsAndEmoticonsLower[i - (startI + 1)]]):
        valence = valence *
    nScalar if startI == 1:
      if wordsAndEmoticonsLower[i - 2] == "never"
          and \ (wordsAndEmoticonsLower[i - 1] ==
          "so" or wordsAndEmoticonsLower[i - 1] ==
          "this"):
        valence = valence * 1.25
      elif wordsAndEmoticonsLower[i - 2] == "without"
          and \ wordsAndEmoticonsLower[i - 1] ==
          "doubt":
        valence = valence
      elif negated([wordsAndEmoticonsLower[i - (startI + 1)]]):
        valence = valence *
    nScalar if startI == 2:
      if wordsAndEmoticonsLower[i - 3] == "never" and \
          (wordsAndEmoticonsLower[i - 2] == "so" or wordsAndEmoticonsLower[i - 2] ==
          "this") or \ (wordsAndEmoticonsLower[i - 1] == "so" or wordsAndEmoticonsLower[i
          - 1] == "this"):
        valence = valence * 1.25
      elif wordsAndEmoticonsLower[i - 3] == "without" and \
          (wordsAndEmoticonsLower[i - 2] == "doubt" or wordsAndEmoticonsLower[i - 1] ==
        "doubt"): valence = valence
      elif negated([wordsAndEmoticonsLower[i - (startI +
        1)]]): valence = valence * nScalar
    return valence
```

```
def _punctutaionEmphasis(self,  text):

  epAmp = self._ampEp(text)
  qmAmp = self._ampQm(text)
  puctEmphAmp = epAmp +
  qmAmp return puctEmphAmp

@staticmethod
def
_ampEp(text):

  epCount  =
  text.count("!") if
  epCount  > 4:
    epCount  = 4


  epAmp = epCount  *
  0.292 return epAmp

@staticmethod
def
_ampQm(text):

  qmCount  =
  text.count("?") qmAmp =
  0
  if qmCount  > 1:
    if qmCount  <= 3:


      qmAmp = qmCount  *
    0.18 else:
      qmAmp =
  0.96 return
  qmAmp

@staticmethod
def _shiftSentimentsScore(sentiments):

  positiveSum  = 0.0
  negativeSum  = 0.0
  neutralCount  = 0
  for sentimentScore  in sentiments:
    if sentimentScore  > 0:
      positiveSum  += (float(sentimentScore)
    + 1)  if sentimentScore  < 0:
      negativeSum  += (float(sentimentScore)
    - 1)  if sentimentScore == 0:
      neutralCount  += 1
  return  positiveSum,  negativeSum,  neutralCount

def valenceScore(self,  sentiments,
  text): if sentiments:
    sumSenti = float(sum(sentiments))

    puctEmphAmp =
    self._punctutaionEmphasis(text)  if sumSenti
    > 0:
      sumSenti +=
puctEmphAmp elif
sumSenti < 0:
  sumSenti -= puctEmphAmp
```

```
      compound = normalize(sumSenti)

      positiveSum, negativeSum, neutralCount =

      self._shiftSentimentsScore(sentiments) if positiveSum >

      math.fabs(negativeSum):
        positiveSum += puctEmphAmp
      elif positiveSum < math.fabs(negativeSum):
        negativeSum -= puctEmphAmp

      total = positiveSum + math.fabs(negativeSum) +
      neutralCount pos = math.fabs(positiveSum / total)
      neg = math.fabs(negativeSum /
      total) neu =
      math.fabs(neutralCount / total)

    else:
      compound = 0.0
      pos = 0.0
      neg = 0.0
      neu = 0.0

    sentiment_dict = \
      {"neg": round(neg, 3),
       "neu": round(neu, 3),
       "pos": round(pos, 3),
       "compound": round(compound,

    4)} return sentiment_dict

def visualize(compoundScore):
  compoundPositive,compoundNegative, compoundNeutral
  =0,0,0 for i in compoundScore:
    if i>0:
      compoundPositive+=
    1 elif i<0:
      compoundNegative+=
    1 else:
      compoundNeutral+=1
  print("Number of Negative Emotions detected:-
  "+str(compoundNegative)) print("Number of Neutral Emotions
  detected:- "+str(compoundNeutral)) print("Number of Postive
  Emotions detected:- "+str(compoundPositive))

  data = [compoundPositive,compoundNegative,compoundNeutral]
  #label = ["Postive: "+str(compoundPositive),"Negative-
"+str(compoundNegative),"Neutral- "+str(compoundNeutral)]
  label =
  ["Postive","Negative","Neutral"]
  colors = ("cyan","orange","grey")
  wp = { 'linewidth' : 1, 'edgecolor' :
  "green" } explode=(0.0,0.2,0.4)
  def func(pct, allvalues):
    #print(pct)
    absolute = math.ceil(pct /
    100*np.sum(allvalues)) return
    "{:.1f}%\n({:d})".format(pct, absolute)

  fig = plt.figure(figsize =(10,
7)) plt.pie(data, labels =
label,
     autopct = lambda pct: func(pct,data),
      wedgeprops =
      wp,
      colors=colors,
      explode=explod
      e, shadow=True,
      textprops = dict(color = "magenta",size=11,
  weight="bold")) plt.title("Different Emotions Pie-Chart")
  plt.show()

def scatterplot(x,y,z):
        fig = plt.figure()
        ax = fig.add_subplot(111,
        projection='3d') ax.scatter(x, y, z, c='r',
        marker='o') ax.set_xlabel('Negative
        score') ax.set_ylabel('Neutral score')
        ax.set_zlabel('Positive score')
        plt.show()



if_name_== '_main_':
  fp = open("processed.txt",'r',encoding='utf-8')
  sentences,compoundScore,compoundScoreNeg, compoundScoreNeu, compoundScorePos
  =[],[],[],[],[] for line in fp:
          line = " ".join(filter(lambda x:x,
          line.split('\n'))) sentences.append(line)
  #print(sentence
  s) fp.close()
  fp = open("completeOriginal.txt",'w',encoding='utf-8')
  fileData = open("data.txt",'w') # stored in order
  negative,neutral,positive analyzer = SentimentIntensityAnalyzer()
  for sentence in sentences:
    vs = analyzer.polarityScore(sentence)
    compoundScore.append(vs["compound"])
    compoundScoreNeg.append(vs["neg"])
    compoundScoreNeu.append(vs["neu"])
    compoundScorePos.append(vs["pos"])
    fileData.write(str(vs["neg"])+","+str(vs["neu"])+","+str(vs["pos"])+"\n")
    print("{:-<65} {}".format(sentence, str(vs))+"\n")
    fp.write("{:-<65} {}".format(sentence,
  str(vs))+"\n") print("----------------------------------")
  fp.close()
  fileData.close()
  #print(compoundScore)
  visualize(compoundScore)
  scatterplot(compoundScoreNeg,compoundScoreNeu,compoundScore
  Pos) print("\n\n Done!")
```

kMeans.py

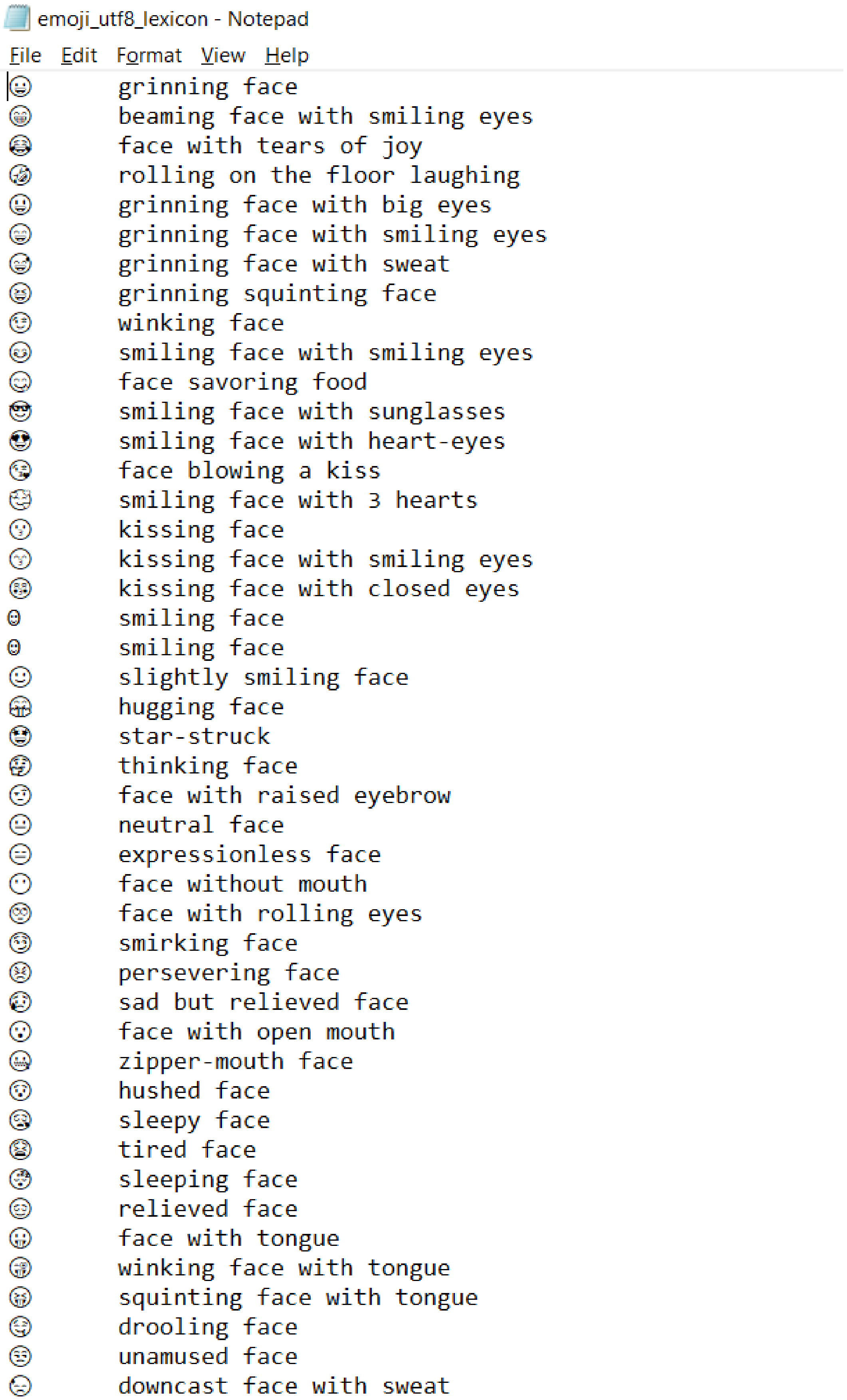
```
import math
import
random
import numpy as
np import pandas
as pd
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def kMeans(sentimentList):
        # Parameters k = number of clusters, & max
        iterations k = 3
        maxIter = 2500
        # Randomly selecting three unique
        centers centroid =
        random.sample(sentimentList, k)

        n =
        len(sentimentList)
        clusterLabels = [0] *
        n

        # Creating a list length to store number of data belonging to each
        cluster length = []

        while maxIter:
                # Assigning label of nearest centroid to each
                point for i in range(n):
                        minDistance = math.dist(sentimentList[i], centroid[0]) # Euclidean
                        distance clusterLabels[i] = 0
                for j in range(1, k):
                        if math.dist(sentimentList[i], centroid[j]) < minDistance:#
                                Euclidean minDistance = math.dist(sentimentList[i],
                                centroid[j]) clusterLabels[i] = j

                #Creating a list for new centroid
                points newCentroid = []

                length= [0]*k
                for i in
                range(k):
                        newCentroid.append([0.0, 0.0,0.0])


                # To find mean, first finding sum of data belonging to each
                cluster for i in range(n):
                        temp = clusterLabels[i]
                        newCentroid[temp][0] +=
                        sentimentList[i][0] newCentroid[temp][1]
                        += sentimentList[i][1]
                        newCentroid[temp][2] +=
                        sentimentList[i][2] length[temp] += 1

                # Now dividing sum by length of respective
                cluster for i in range(k):
                        if length[i]==0:
                                continue
                        newCentroid[i][0]    =    newCentroid[i][0]    /
                        length[i]            newCentroid[i][1]            =
                        newCentroid[i][1]  /  length[i]  newCentroid[i][2]
                        = newCentroid[i][2]  / length[i]
```

```
                # Assigning new centroids to original
                centroids centroid = newCentroid
                maxIter -= 1

        # Returning labels belonging to each entry
        return
        clusterLabels,length[0],length[1],length[2]

def visual(len_c1,len_c2,len_c3):
  print("Number of elements in Cluster-1 :-
  "+str(len_c1)) print("Number of elements in
  Cluster-2 :- "+str(len_c2)) print("Number of
  elements in Cluster-3 :- "+str(len_c3))

  #Pi - chart data
  data = [len_c3,len_c1,len_c2]
  #label = ["Cluster-3: "+str(len_c3),"Cluster-1: "+str(len_c1),"Cluster-2:
  "+str(len_c2)] label = ["Cluster-3","Cluster-1","Cluster-2"]
  wp = { 'linewidth' : 0.1, 'edgecolor' : "green" }
  explode=(0.0,0.2,0.4)
  colors =
  ("cyan","orange","grey") def
  func(pct, allvalues):
    #print(pct)
    absolute = math.ceil(pct /
    100*np.sum(allvalues)) return
    "{:.1f}%\n({:d})".format(pct, absolute)

  fig = plt.figure(figsize =(10,
  7)) plt.pie(data, labels =
  label,
                autopct = lambda pct:
                func(pct,data), shadow = True,
                wedgeprops =
                wp, explode =
                explode, colors =
                colors,
                textprops= dict(color ="magenta",size= 8,
  weight="bold")) plt.title("Different Emotions Pie-Chart")
  plt.show()


def scatterplot(dataset):
        colors= ['blue','green','red']
        dataset['color']=
        dataset.Cluster.map({0:colors[0],1:colors[1],2:colors[2]}) fig =
        plt.figure(figsize=(26,7))
        ax = fig.add_subplot(131, projection='3d')
        ax.scatter(dataset.Negative, dataset.Neutral, dataset.Positive, c=dataset.color)

        ax.set_xlabel('Negative')
        ax.set_ylabel('Neutral')
        ax.set_zlabel('Positive')
        ax.set_title('Cluster-
        Graph') plt.show()

        plt.scatter(dataset.Negative, dataset.Positive,
        c=dataset.color) plt.xlabel('Negative')
        plt.ylabel('Positive')
        plt.title('Cluster-plot Neg-Pos
        axis') plt.show()
```

```
plt.scatter( dataset.Neutral,  dataset.Positive,
c=dataset.color) plt.xlabel('Neutral')
```

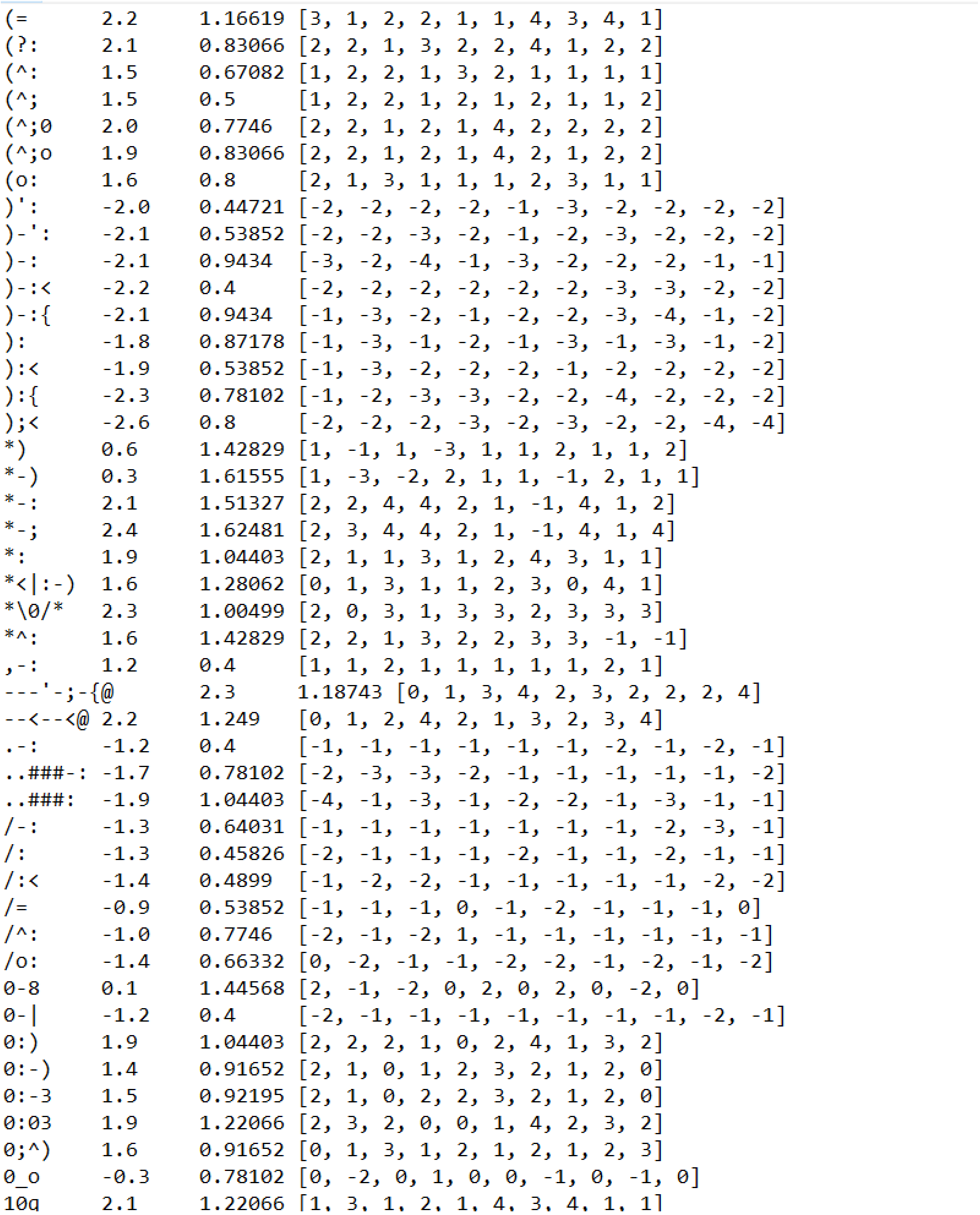
```
    plt.ylabel('Positive')
    plt.title('Cluster-plot  Neu-Pos
    axis') plt.show()

    plt.scatter(dataset.Negative,
    dataset.Neutral,c=dataset.color)  plt.xlabel('Negative')
    plt.ylabel('Neutral')
    plt.title('Cluster-plot  Neg-Neu
    axis') plt.show()

if_name_== "_main_":
    fp = open("data.txt",  'r')
    lines = fp.readlines()  #in order  of
    negative,neutral,positive   fp.close()

    sentimentList,neg,neu,pos  =
    [],[],[],[] for line in lines:
        temp = line.split(',')
        temp[2] =
        temp[2].rstrip('\n')  temp[0]
        = float(temp[0])
        neg.append(float(temp[0]))
        temp[1] = float(temp[1])
        neu.append(float(temp[1]))
        temp[2] = float(temp[2])
        pos.append(float(temp[2]))
        sentimentList.append(tem
        p)

    result, len_c1, len_c2, len_c3 =
    kMeans(sentimentList)  dataset =
    pd.DataFrame(list(zip(neg,neu,pos,result)),
      columns= ['Negative','Neutral','Positive','Cluster'])
    visual(len_c1,len_c2,len_c3)
    scatterplot(dataset)
```

SNIPS OF TEXT FILE USED FOR ANALYSIS



```
😀	grinning face
😁	beaming face with smiling eyes
😂	face with tears of joy
🤣	rolling on the floor laughing
😃	grinning face with big eyes
😄	grinning face with smiling eyes
😅	grinning face with sweat
😆	grinning squinting face
😉	winking face
😊	smiling face with smiling eyes
😋	face savoring food
😎	smiling face with sunglasses
😍	smiling face with heart-eyes
😘	face blowing a kiss
🥰	smiling face with 3 hearts
😗	kissing face
😙	kissing face with smiling eyes
😚	kissing face with closed eyes
☺️	smiling face
☺	smiling face
🙂	slightly smiling face
🤗	hugging face
🤩	star-struck
🤔	thinking face
🤨	face with raised eyebrow
😐	neutral face
😑	expressionless face
😶	face without mouth
🙄	face with rolling eyes
😏	smirking face
😣	persevering face
😥	sad but relieved face
😮	face with open mouth
🤐	zipper-mouth face
😯	hushed face
😪	sleepy face
😫	tired face
😴	sleeping face
😌	relieved face
😛	face with tongue
😜	winking face with tongue
😝	squinting face with tongue
🤤	drooling face
😒	unamused face
😓	downcast face with sweat
```

Figure 20 The file contains the description of all the Emojis



```
(=      2.2     1.16619 [3, 1, 2, 2, 1, 1, 4, 3, 4, 1]
(?:     2.1     0.83066 [2, 2, 1, 3, 2, 2, 4, 1, 2, 2]
(^:     1.5     0.67082 [1, 2, 2, 1, 3, 2, 1, 1, 1, 1]
(^;     1.5     0.5     [1, 2, 2, 1, 2, 1, 2, 1, 1, 2]
(^;0    2.0     0.7746  [2, 2, 1, 2, 1, 4, 2, 2, 2, 2]
(^;o    1.9     0.83066 [2, 2, 1, 2, 1, 4, 2, 1, 2, 2]
(o:     1.6     0.8     [2, 1, 3, 1, 1, 1, 2, 3, 1, 1]
)':     -2.0    0.44721 [-2, -2, -2, -2, -1, -3, -2, -2, -2, -2]
)-':    -2.1    0.53852 [-2, -2, -3, -2, -1, -2, -3, -2, -2, -2]
)-:     -2.1    0.9434  [-3, -2, -4, -1, -3, -2, -2, -2, -1, -1]
)-:<    -2.2    0.4     [-2, -2, -2, -2, -2, -2, -3, -3, -2, -2]
)-:{    -2.1    0.9434  [-1, -3, -2, -1, -2, -2, -3, -4, -1, -2]
):      -1.8    0.87178 [-1, -3, -1, -2, -1, -3, -1, -3, -1, -2]
):<     -1.9    0.53852 [-1, -3, -2, -2, -2, -1, -2, -2, -2, -2]
):{     -2.3    0.78102 [-1, -2, -3, -3, -2, -2, -4, -2, -2, -2]
);<     -2.6    0.8     [-2, -2, -2, -3, -2, -3, -2, -2, -4, -4]
*)      0.6     1.42829 [1, -1, 1, -3, 1, 1, 2, 1, 1, 2]
*-)     0.3     1.61555 [1, -3, -2, 2, 1, 1, -1, 2, 1, 1]
*-:     2.1     1.51327 [2, 2, 4, 4, 2, 1, -1, 4, 1, 2]
*-;     2.4     1.62481 [2, 3, 4, 4, 2, 1, -1, 4, 1, 4]
*:      1.9     1.04403 [2, 1, 1, 3, 1, 2, 4, 3, 1, 1]
*<|:-)  1.6     1.28062 [0, 1, 3, 1, 1, 2, 3, 0, 4, 1]
*\0/*   2.3     1.00499 [2, 0, 3, 1, 3, 3, 2, 3, 3, 3]
*^:     1.6     1.42829 [2, 2, 1, 3, 2, 2, 3, 3, -1, -1]
,-:     1.2     0.4     [1, 1, 2, 1, 1, 1, 1, 1, 2, 1]
---'-;-{@       2.3     1.18743 [0, 1, 3, 4, 2, 3, 2, 2, 2, 4]
--<--<@ 2.2     1.249   [0, 1, 2, 4, 2, 1, 3, 2, 3, 4]
.-:     -1.2    0.4     [-1, -1, -1, -1, -1, -1, -2, -1, -2, -1]
..###-: -1.7    0.78102 [-2, -3, -3, -2, -1, -1, -1, -1, -1, -2]
..###:  -1.9    1.04403 [-4, -1, -3, -1, -2, -2, -1, -3, -1, -1]
/-:     -1.3    0.64031 [-1, -1, -1, -1, -1, -1, -1, -2, -3, -1]
/:      -1.3    0.45826 [-2, -1, -1, -1, -2, -1, -1, -2, -1, -1]
/:<     -1.4    0.4899  [-1, -2, -2, -1, -1, -1, -1, -1, -2, -2]
/=      -0.9    0.53852 [-1, -1, -1, 0, -1, -2, -1, -1, -1, 0]
/^:     -1.0    0.7746  [-2, -1, -2, 1, -1, -1, -1, -1, -1, -1]
/o:     -1.4    0.66332 [0, -2, -1, -1, -2, -2, -1, -2, -1, -2]
0-8     0.1     1.44568 [2, -1, -2, 0, 2, 0, 2, 0, -2, 0]
0-|     -1.2    0.4     [-2, -1, -1, -1, -1, -1, -1, -1, -2, -1]
0:)     1.9     1.04403 [2, 2, 2, 1, 0, 2, 4, 1, 3, 2]
0:-)    1.4     0.91652 [2, 1, 0, 1, 2, 3, 2, 1, 2, 0]
0:-3    1.5     0.92195 [2, 1, 0, 2, 2, 3, 2, 1, 2, 0]
0:03    1.9     1.22066 [2, 3, 2, 0, 0, 1, 4, 2, 3, 2]
0;^)    1.6     0.91652 [0, 1, 3, 1, 2, 1, 2, 1, 2, 3]
0_o     -0.3    0.78102 [0, -2, 0, 1, 0, 0, -1, 0, -1, 0]
10q     2.1     1.22066 [1, 3, 1, 2, 1, 4, 3, 4, 1, 1]
```

Figure 21 Contains the score for different words and emoticons which is used for analysis