African University of Science & Technology
Computer Science Department

Masters Thesis

Sentiment Analysis Based on Social Media Data

Abstract

Sentiment analysis is a challenging natural language processing (NLP) problem. Many AI systems have been developed which can detect the polarity of a sentence (degree of positivity, neutrality or negativity). But more information such as the emotion of the author can be detected. Our task here is to build an artificial agent – or an AI system – that is capable of detecting polarity in a document as well as the emotion of the author. In a general sense, sentiment analysis determines the opinion regarding the object/subject in discussion. But emotion detection can identify the particular "mood" of the author. Social platforms – such as Facebook, Twitter, IMDB, or comment sections from online newspapers – to name a few – can provide a huge corpus of – usually unlabelled or extremely sparsely labelled – content.

Using modern machine learning tools and the available computational power, an agent can analyse the content – given usually as a list of messages – and detect the general emotion "within the message". It must be capable of identifying subjects with unstable or "chaotic" mood that might require attention.

Our aim is to attempt detecting the underlying emotion as well as the polarity of a document. In this process we will make use of open source libraries and publicly available data sets. The program will be able to run locally, both in geographic and in cultural sense, and to analyse the results that were obtained.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

July 2019 Kudzai Zishumba

Thesis supervisor:
Prof. dr. Lehel Csató

Faculty of Mathematics and Informatics,
Babeş Bolyai University of Cluj-Napoca,
Romania

AFRICAN UNIVERSITY OF SCIENCE & TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Masters Thesis

Sentiment Analysis Based on Social Media Data

THESIS SUPERVISOR:

PROF. DR. LEHEL CSATÓ

FACULTY OF MATHEMATICS AND INFORMATICS,
BABEŞ BOLYAI UNIVERSITY OF CLUJ-NAPOCA,
ROMANIA

STUDENT:

KUDZAI ZISHUMBA

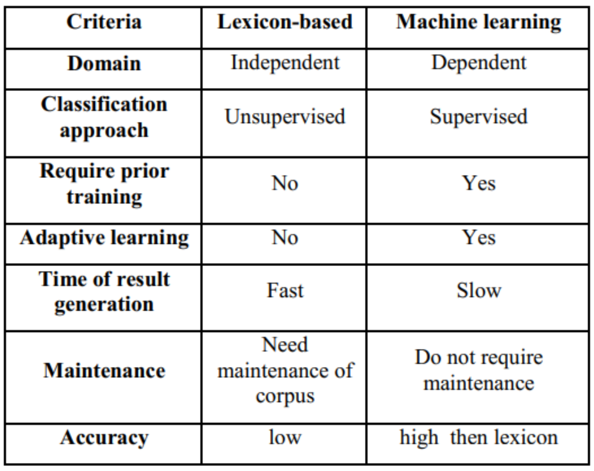
JULY 2019

Contents

1. Chapter

Basics of Sentiment analysis

There are two approaches to sentiment analysis: lexicon-based and machine learning-based. Machine learning as the name suggests requires the model to be trained, lexicon based consists of a set of rules and doesn't require prior training, a third method although not used very often is a hybrid method that combines both lexicon and machine learning approaches, generally it yields better results than either method individually.[Mukherjee, 2012] Table 1.1 will summarize the differences between the two approaches.



Criteria	Lexicon-based	Machine learning
Domain	Independent	Dependent
Classification approach	Unsupervised	Supervised
Require prior training	No	Yes
Adaptive learning	No	Yes
Time of result generation	Fast	Slow
Maintenance	Need maintenance of corpus	Do not require maintenance
Accuracy	low	high then lexicon

Table 1.1: Comparison between lexicon and machine learning

Currently Sentiment analysis techniques seek to classify text as either positive, negative or neutral, using methods in Natural language processing (NLP). The common approach implemented involves statistical analysis and machine learning methods.[Mukherjee, 2012]

However people often express a particular emotion towards a certain topic. Specific emotions such as anger can lead to social unrest. This is undesirable but it can be helpful if we are able to quickly detect this before it occurs. The proposed system will be a hybrid system it will use both machine learning approach as well as lexicon-based approach to detect both sentiment polarity and emotion.

1.1 Lexicon (VaderSentiment)

A lexicon is a collection of information about the words of a language about the lexical categories to which they belong. Lexicon based sentiment analysis makes use of labeled text this is very useful in detecting sentiment by emoticons. VaderSentiment is a python library that can be used to achieve this.

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER makes use of sentiment lexicon, which is a list of lexical features such as words, these words are labeled according to their sentiment polarity, positive, neutral or negative.

An emoticon, short for "emotion icon", also known simply as an emote, is a pictorial representation of a facial expression using characters usually punctuation marks, numbers and letters to express a person's feelings or mood, or as a time-saving method rather than to type out a whole word, an emoticon can be used to convey the writer's feelings or intended tone.

Some key features of Vader sentiment are: -

1. Punctuation: The use of an exclamation mark(!), increases the magnitude of the intensity without modifying the semantic orientation. For example, "The food here is good!" is more intense than "The food here is good." and an increase in the number of (!), increases the magnitude accordingly.
2. Capitalization: Using upper case letters to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity. For example, "The food here is GREAT!" conveys more intensity than "The food here is great!"
3. Degree modifiers: Also called intensifiers, they impact the sentiment intensity by either increasing or decreasing the intensity. For example, "The service here is extremely good" is more intense than "The service here is good", whereas "The service here is marginally good" reduces the intensity.
4. Conjunctions: Use of conjunctions like "but" signals a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant. "The food here is great, but the service is horrible" has mixed sentiment, with the latter half dictating the overall rating.
5. Preceding Tri-gram: By examining the tri-gram preceding a sentiment-laden lexical feature, we catch nearly 90% of cases where negation flips the polarity of the text. A negated sentence would be "The food here isn't really all that great".

1.1.1 Subjectivity and Objectivity (TextBlob)

Subjectivity and objectivity differentiates opinionated and factual based text.[Mukherjee, 2012] Being able to detect subjectivity and objectivity is an important part of sentiment analysis, because it can tell us more about the authors emotions and intentions, a subjective person is more likely to invoke action than an objective person. TextBlob python library can be used to detect the degree of subjectivity and the degree of objectivity, these results together with the polarity scores will be used to determine the impact of the sentiment.

1.1.2 Multi-label convolutional neural network text classifier (Spacy)

SpaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python Spacy makes use of a neural network text classifier. SpaCy can be used to categorize documents into the different emotion classes. The Neural network is trained using stochastic gradient descent where the estimate of the error used to update the weights is calculated based on a subset of the training dataset.

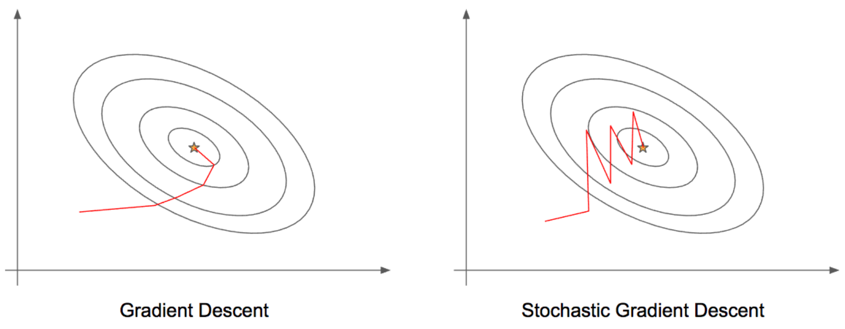


Figure 1.1: Stochastic Gradient decent

A SpaCy model uses statistical decisions to make predictions, the decision is based on examples used to train the model. Training the model requires training data, training data is: examples of text with the labels that we want to predict. This will be in the following categories

- "anger"
- "disgust"
- "fear"
- "sadness"
- "shame"
- "joy"
- "guilt"

The following diagram shows how SpaCy model is trained

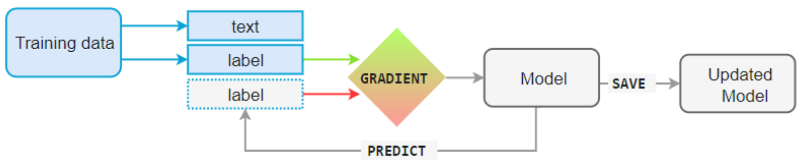


Figure 1.2: SpaCy Training:https://spacy.io/usage/training

1.1.3 Training Dataset (ISEAR)

Robert Plutchik invented a wheel of emotions. He suggested eight primary emotions: joy opposite to sadness. Similarly, anger opposite to fear; trust opposite to disgust and surprise opposite to anticipation. These four opposite emotion pairs, show the 8 basic emotions. Additionally, Plutchik's model shows connections between the ideas of circle of emotions using a color wheel. Like the case of colors, primary emotions can also be expressed at different degrees of their intensities, for each emotion there are three degrees. For example, serenity is a less intense degree of joy and ecstasy is a more intense degree of joy. Plutchik's emotions can be mixed with one another forming a new different emotion. For example, combination of joy and trust resulted to form a new emotion 'love'. Likewise, joy, anger and trust are combined and form jealousy[Pimpalkar, 2014].

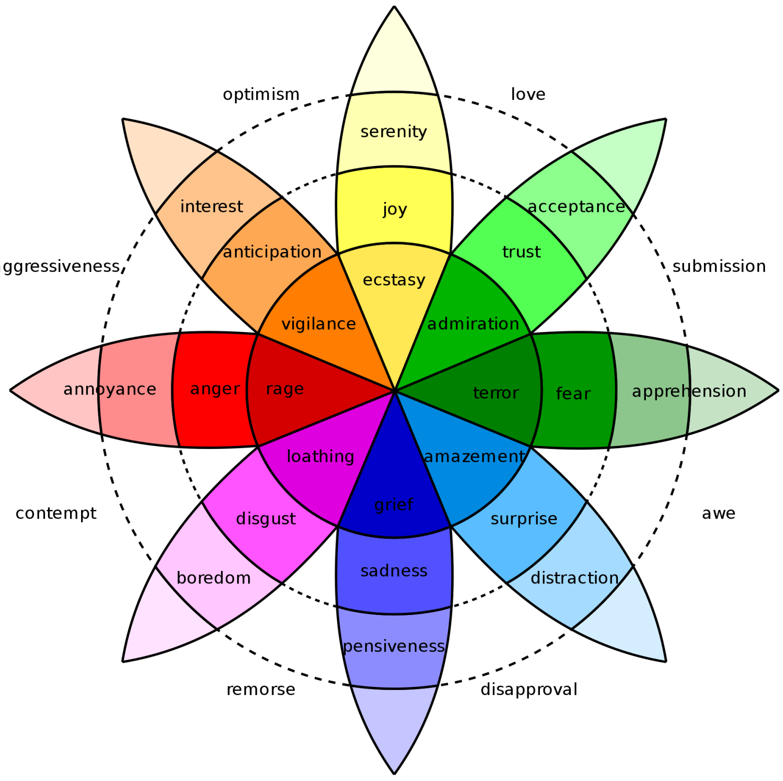


Figure 1.3: Robert Plutchik's wheel of emotions

ISEAR (International Survey on Emotion Antecedents and Reactions) is an open dataset collected by Klaus R. Scherer and Harald Wallbott. ISEAR dataset contains seven major emotions based on Robert Plutchik's wheel of emotions : joy, fear, anger, sadness, disgust, shame, and guilt.
It has over 7,600 entries for the emotion classes. This will be a good training dataset for emotion detection.

1.1.4 Streaming twitter (tweepy)

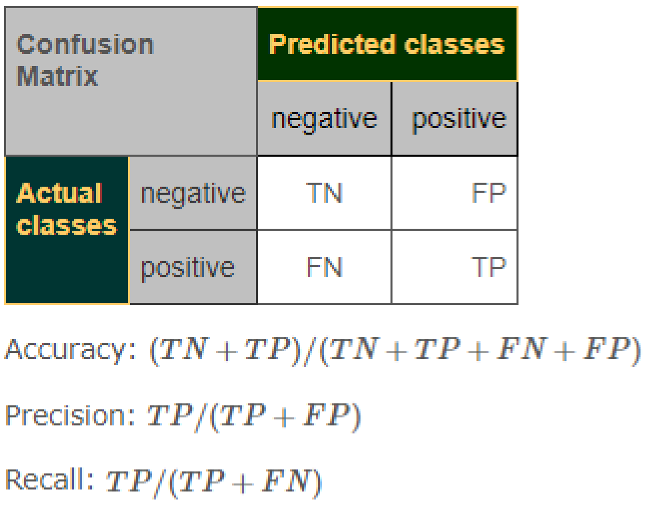
Tweepy is an open source python library that can be used to stream tweets from twitter in real time through the Twitter API.

1.1.5 Confusion matrix

A confusion matrix is used in machine learning to determine the performance of a classification algorithm. It is a tabular form than compares test data for which its true values are known. I will use the confusion matrix to evaluate the performance of my classification model.

A confusion matrix consists of the following classes.

- true positives (TP): This is when we have given a correct positive prediction
- true negatives (TN): This is when we have given a correct negative prediction.
- false positives (FP): This is when we have predicted yes, but the true value is no
- false negatives (FN): This is when we have predicted no, but the true value is yes



Confusion Matrix		Predicted classes	
		negative	positive
Actual classes	negative	TN	FP
	positive	FN	TP

Accuracy: $(TN + TP)/(TN + TP + FN + FP)$

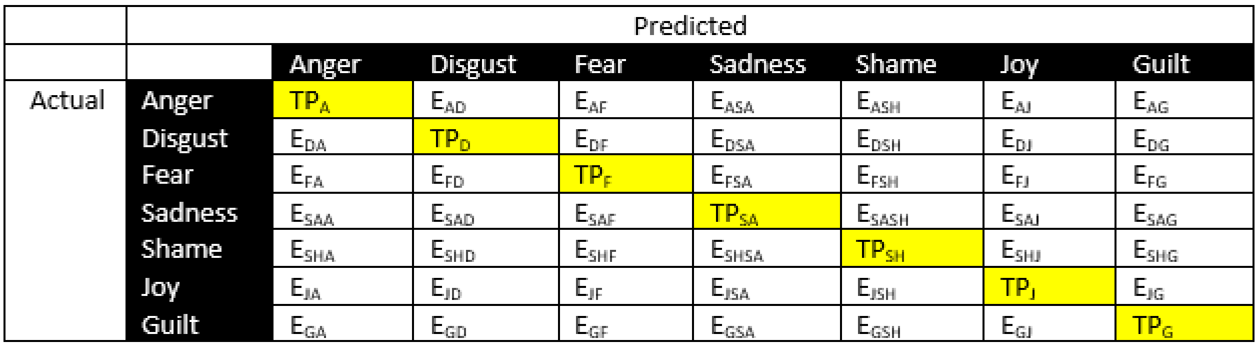
Precision: $TP/(TP + FP)$

Recall: $TP/(TP + FN)$

Figure 1.4: Confusion matrix

The above table shows how to determine the accuracy of the prediction using a confusion matrix. However, there are problems with accuracy. It assumes equal costs for both kinds of errors that is to say it assumes that we have an equal number of positives and negatives. A 99% accuracy can either be excellent, good, mediocre, poor or terrible depending upon the problem

In the case of having multiple classes the confusion matrix becomes



		Predicted						
		Anger	Disgust	Fear	Sadness	Shame	Joy	Guilt
Actual	Anger	TP_{A}	E_{AD}	E_{AF}	E_{ASA}	E_{ASH}	E_{AJ}	E_{AG}
	Disgust	E_{DA}	TP_{D}	E_{DF}	E_{DSA}	E_{DSH}	E_{DJ}	E_{DG}
	Fear	E_{FA}	E_{FD}	TP_{F}	E_{FSA}	E_{FSH}	E_{FJ}	E_{FG}
	Sadness	E_{SAA}	E_{SAD}	E_{SAF}	TP_{SA}	E_{SASH}	E_{SAJ}	E_{SAG}
	Shame	E_{SHA}	E_{SHD}	E_{SHF}	E_{SHSA}	TP_{SH}	E_{SHJ}	E_{SHG}
	Joy	E_{JA}	E_{JD}	E_{JF}	E_{JSA}	E_{JSH}	TP_{J}	E_{JG}
	Guilt	E_{GA}	E_{GD}	E_{GF}	E_{GSA}	E_{GSH}	E_{GJ}	TP_{G}

Figure 1.5: Confusion matrix with multiple classes

Things to notice about a confusion matrix with multiple classes

- **total number of test examples** of any class if given by the sum of the corresponding row **(i.e TP + FN)**
- **Total number of False Negatives** for a class is the sum of values in the corresponding row excluding the **True positives**
- **Total number of false positives** for a class is the sum of values in the corresponding column excluding the **true positives**
- **Total number of true negatives** for a certain class will be the sum of all columns and rows excluding that classes column and row

Accuracy calculated as the sum of correct classifications divided by the total number of classifications
Precision is calculated as **TP/(TP+ FP)**
Recall also called sensitivity, corresponds to the tue positive rate of the considered class calculated as Recall=Sensitivity=TP/(TP+FN)

2. Chapter

Sentiment analysis

2.0.1 Implementing (VaderSentiment and TextBlob)

Using **"VaderSentiment"** library. Using an analyzer object, and looping through the list of tweets calling the analyser.polarity_scores() method and appending the results to an array.

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()#create the vader sentiment analyser object
sentiment_vs_tweets=[]#array will hold the processd tweets with the sentiment and objective
    labels
index=0
for t in tweets:
    sentiment_vs_tweets.append( analyser.polarity_scores(t) )#add the processed tweet to the
        list
    print(sentiment_vs_tweets[index])#print the sentiment polarity and subjecivity
    index=index+1
```

The above code will produces the following results

```
{'neg': 0.252, 'neu': 0.581, 'pos': 0.168, 'compound': -0.3182}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.194, 'neu': 0.806, 'pos': 0.0, 'compound': -0.5574}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.194, 'neu': 0.806, 'pos': 0.0, 'compound': -0.5574}
```

Using **TextBlob** to detect the degree of subjectivity and objectivity of the documents.

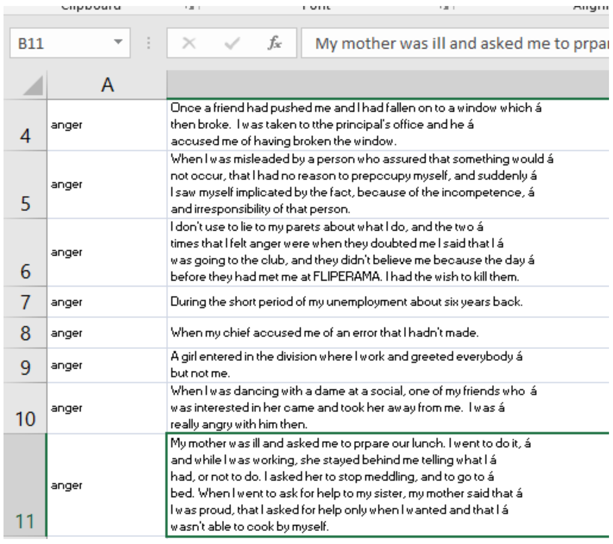
```
#here we use TextBlob python library to detect sentence/tweet/document polarity as well as
    subjectivity and store the results
#textblob uses a machine learning approach
from textblob import TextBlob
sentiment_tb_tweets=[]#array will hold the processd tweets with the sentiment and objective
    labels
index=0
for t in tweets:
    sentiment_tb_tweets.append( TextBlob(t) )#add the processed tweet to the list
    print(sentiment_tb_tweets[index].sentiment)#print the sentiment polarity and subjecivity
    index=index+1

sentiment_tb_tweets[0].subjectivity
```

The above code produces the following results

```
Sentiment(polarity=0.0, subjectivity=0.25)
Sentiment(polarity=0.14285714285714285, subjectivity=0.31785714285714284)
Sentiment(polarity=0.0, subjectivity=0.06666666666666667)
Sentiment(polarity=0.2, subjectivity=0.2)
Sentiment(polarity=0.0, subjectivity=0.06666666666666667)
```

2.0.2 Training SpaCy Text classifier

Using the ISEAR training data We can train a SpaCy model to classify text into some categories



B11 | My mother was ill and asked me to prpai

	A	
4	anger	Once a friend had pushed me and I had fallen on to a window which á then broke. I was taken to tthe principal's office and he á accused me of having broken the window.
5	anger	When I was misleaded by a person who assured that something would á not occur, that I had no reason to prepccupy myself, and suddenly á I saw myself implicated by the fact, because of the incompetence, á and irresponsibility of that person.
6	anger	I don't use to lie to my parets about what I do, and the two á times that I felt anger were when they doubted me I said that I á was going to the club, and they didn't believe me because the day á before they had met me at FLIPERAMA. I had the wish to kill them.
7	anger	During the short period of my unemployment about six years back.
8	anger	When my chief accused me of an error that I hadn't made.
9	anger	A girl entered in the division where I work and greeted everybody á but not me.
10	anger	When I was dancing with a dame at a social, one of my friends who á was interested in her came and took her away from me. I was á really angry with him then.
11	anger	My mother was ill and asked me to prpare our lunch. I went to do it, á and while I was working, she stayed behind me telling what I á had, or not to do. I asked her to stop meddling, and to go to á bed. When I went to ask for help to my sister, my mother said that á I was proud, that I asked for help only when I wanted and that I á wasn't able to cook by myself.

Figure 2.1: Sample ISEAR training dataset with labels

We begin training the SpaCy model on the above training set, start by reading in the data

```
nlp     = spacy.load('en_core_web_sm')#load english model
file     = open("data/train_data.txt","r")#read in training set from text file uses pipe
    delimiter
#read in the training set with the labels
for row in f_data:
    train_data.append( (row.split('|')[0],\
                        {"cats":\
                         {\
                          "anger"      :int(row.split('|')[1]),\
                          "disgust"    :int(row.split('|')[2]),\
                          "fear"       :int(row.split('|')[3]),\
                          "sadness"    :int(row.split('|')[4]),\
                          "shame"      :int(row.split('|')[5]),\
                          "joy"        :int(row.split('|')[6]),\
                          "guilt"      :int(row.split('|')[7])\
                        }}) )
```

We then update the nlp model with the new training set

```
#create the spacy nlp pipeline
textcat = nlp.create_pipe('textcat')
#add the pipeline to the nlp model| last=true since we want to add these conponents at the end
    of the model
nlp.add_pipe(textcat, last=True)
#declare the lables/categories
textcat.add_label('anger')
textcat.add_label('disgust')
textcat.add_label('fear')
textcat.add_label('sadness')
textcat.add_label('shame')
textcat.add_label('joy')
textcat.add_label('guilt')

optimizer = nlp.begin_training()
#next we will update the model with our own train_data from the csv file
for itn in range(1):
    for doc, gold in train_data:
        random.shuffle(train_data)
        losses = {}
        index = 0
        for text, annotations in train_data:
            nlp.update([doc], [gold], sgd=optimizer, drop=0.45, losses=losses)#drop rate at
                .45
            print("Iteration: {0}, %{1:.2f} Complete".format((itn+1), index/len(train_data) *
                100))#show the progress
            index += 1
        print(losses)
```

After training the model we can now run some tweets via the model

```
#determine the class of the tweets using the spacy model trained earlier
#create a document and run it thru the nlp pipeline
classified_tweets = []#array will hold the spacy classified tweets
index=0
for t in tweets:
    classified_tweets.append(  nlp(t) )
    print( classified_tweets[index].cats )#print the categories/classes of the document
    index=index+1
```

The above code produces the following results

```
{'anger': 0.8638094067573547, 'disgust': 0.41107258200645447, 'fear': 0.007243141997605562, '
    sadness': 0.04211084172129631, 'shame': 0.7023115158081055, 'joy': 0.6675902605056763, '
    guilt': 0.9010916352272034}
{'anger': 0.7449342012405396, 'disgust': 0.4830150306224823, 'fear': 0.016353892162442207, '
    sadness': 0.2870638370513916, 'shame': 0.40636202692985535, 'joy': 0.1324012577533722, '
    guilt': 0.41464924812316895}
{'anger': 0.7480689883232117, 'disgust': 0.4755619764328003, 'fear': 0.035106029361486435, '
    sadness': 0.1592441201210022, 'shame': 0.4828783869743347, 'joy': 0.16910341382026672, '
    guilt': 0.4793942868709564}
{'anger': 0.7859129905700684, 'disgust': 0.47182315587997437, 'fear': 0.026509756222367287, '
    sadness': 0.32458046078681946, 'shame': 0.12689484655857086, 'joy': 0.23759214580059052, '
    guilt': 0.9908844828605652}
{'anger': 0.7480689883232117, 'disgust': 0.4755619764328003, 'fear': 0.035106029361486435, '
    sadness': 0.1592441201210022, 'shame': 0.4828783869743347, 'joy': 0.16910341382026672, '
    guilt': 0.4793942868709564}
```

SpaCy model is able to classify the class of the text based on our training set

2.0.3 Streaming twitter API(Tweepy)

The python library tweepy can support accessing Twitter via Basic Authentication and the newer method, OAuth. The API parameters allow filtering tweets based on language, geolocation and tweets containing a particular word or phrase.

```
auth    = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

stream = Stream(auth, listner)
#run this code async
streamer=stream.filter(track=search_terms,is_async=True,languages=langs,locations=loca_tions)
```

The listener object is a class that inherits from StreamListener class, it listens for any incoming tweets, cleans the tweets and then stores them in an array.

```
class StdOutListener(StreamListener):
    def on_data(self, data):
        obj = json.loads(str(data))#convert the tweet into a json format
        #clean the tweets before inserting into the array
        clean_tweet = remove_pattern(obj["text"],"@[\w]*")#remove twitter handles (@user)
        clean_tweet = clean_tweet.replace("[^a-zA-Z#]", "␣")#replace puntuation marks
        clean_tweet = clean_tweet.replace("RT␣:","")#replace the re tweet symbol
        clean_tweet = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F
            ][0-9a-fA-F]))+', '', clean_tweet, flags=re.MULTILINE)#remove hyperlinks

        score       = vs_analyser.polarity_scores(clean_tweet)
        sentiment_polarity_scores.append( float(score["compound"]) )#append the sentiment
            score to the array
        sentiment_tb_tweets.append( TextBlob(clean_tweet) )#append the texblob object to the
            array
        sentiment_vs_tweets.append( nlp(clean_tweet) )#append the vader sentiment object to
            the array
        tweets.append(clean_tweet)#append the clean tweet to the list of tweets
```

2.0.4 Evaluating SpaCy classification model

Because of the huge size of the training dataset, using the whole data set to train and evaluate the model would take too long. The solution therefore is to use a small sample of the dataset to train and evaluate. The training set has 23 samples and the evaluation set has 8 samples.
The following is the results of the evaluation based on the confusion matrix.

```
Accuracy:  1.4285714285714286



Precision_anger:  0.6666666666666666
Precision_disgust:  1.0
Precision_fear:  0.4
Precision_sadness:  1.0
Precision_shame:  1.0
Precision_joy:  1.0
Precision_guilt:  1.0
Total_Precision:  0.8666666666666666



Recall_anger:  1.0
Recall_disgust:  0.6666666666666666
Recall_fear:  1.0
Recall_sadness:  0.5
Recall_shame:  0.5
Recall_joy:  0.5
Recall_guilt:  0.5
Total_Recall:  0.6666666666666666
```

The results for each individual classification is quite satisfactory, considering that the training set and evaluation set is considerably small. Both the precision and the Recall is 50% and above.

3. Chapter

Results and analysis

3.1 Data visualization

Because of the way the human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than poring over spreadsheets or reports. Data visualization makes it easier to identify areas that need attention.

3.1.1 matplotlib and seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Matplotlib allows plotting multiple graphs on a single figure. Analysis can be made on different metrics at the same time. It also allows updating graph data in realtime.

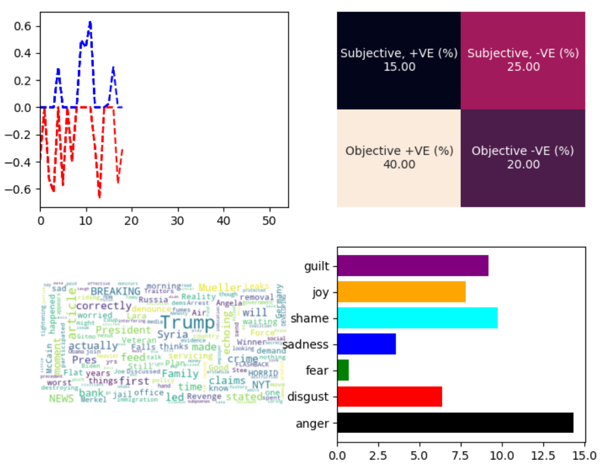


Figure 3.1: Real time plot of the analysis

The **wordcloud** gives a summary of the tweets in realtime.
The **timeseries** gives a **stochastic trend** of the polarity of the tweets, with **1 = positive, -1 = negative and 0 = neutral**. The **barchart** displays the values of the number of tweets in the given category
The **heat map** shows the **polarity** as well as the **subjectivity** in terms of percentage, a negative and subjective scenario is the least desirable while a positive and objective scenario is the most desirable.

With this realtime analysis one can quickly see the prevailing mood as well as a summary of the tweets around a given topic.

Bibliography

S. Mukherjee. Sentiment analysis, a literature survey. Technical report, Indian Institute of Technology, Bombay, 2012. URL https://www.researchgate.net/publication/236203597_Sentiment_Analysis_A_Literature_Survey.

A. Pimpalkar. *Review on Developing Corpora for Sentiment Analysis Using Plutchik?s Wheel of Emotions with Fuzzy Logic*. 2014. URL https://www.researchgate.net/publication/309180696_Review_on_Developing_Corpora_for_Sentiment_Analysis_Using_Plutchik%27s_Wheel_of_Emotions_with_Fuzzy_Logic.