**5. IMPLEMENTATION**

**5.1 Sentiment Analysis**

Sentiment analysis (sometimes known as opinion mining or emotion AI) 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.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. The attitude may be a judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication (that is to say, the emotional effect intended by the author or interlocutor).

**5.1.1 Types**

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".

Early work in that area includes Turney[1] and Pang[2] 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[3] and Snyder[4] among others: Pang and Lee[3] 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[4] 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). Even though in most statistical classification methods, the neutral class is ignored under the assumption that neutral texts lie near the boundary of the binary classifier, several researchers suggest that, as in every polarity problem, three categories must be identified. Moreover, it can be proven that specific classifiers such as the Max Entropy[5] and the SVMs[6] can benefit from the introduction of a neutral class and improve the overall accuracy of the classification. There are in principle two ways for operating with a neutral class. Either, the algorithm proceeds by first identifying the neutral language, filtering it out and then assessing the rest in terms of positive and negative sentiments, or it builds a three way classification in one step.[7] This second approach often involves estimating a probability distribution over all categories (e.g. naive Bayes classifiers as implemented by Python's NLTK kit). Whether and how to use a neutral class depends on the nature of the data: if the data is clearly clustered into neutral, negative and positive language, it makes sense to filter the neutral language out and focus on the polarity between positive and negative sentiments. If, in contrast, the data is mostly neutral with small deviations towards positive and negative affect, this strategy would make it harder to clearly distinguish between the two poles.

A different method for determining sentiment is the use of a scaling system whereby words commonly associated with having a negative, neutral or positive sentiment with them are given an associated number on a −10 to +10 scale (most negative up to most positive). This makes it possible to adjust the sentiment of a given term relative to its environment (usually on the level of the sentence). When a piece of unstructured text is analyzed using natural language processing, each concept in the specified environment is given a score based on the way sentiment words relate to the concept and its associated score.[8] This allows movement to a more sophisticated understanding of sentiment, because it is now possible to adjust the sentiment value of a concept relative to modifications that may surround it. Words, for example, that intensify, relax or negate the sentiment expressed by the concept can affect its score. Alternatively, texts can be given a positive and negative sentiment strength score if the goal is to determine the sentiment in a text rather than the overall polarity and strength of the text.[9]

**5.1.1.1 Subjectivity/objectivity identification**

This task is commonly defined as classifying a given text (usually a sentence) into one of two classes: objective or subjective.[10] This problem can sometimes be more difficult than polarity classification.[11] The subjectivity of words and phrases may depend on their context and an objective document may contain subjective sentences (e.g., a news article quoting people's opinions). Moreover, as mentioned by Su,[12] results are largely dependent on the definition of subjectivity used when annotating texts. However, Pang[13] showed that removing objective sentences from a document before classifying its polarity helped improve performance.

**5.1.1.2 Feature/aspect-based**

It refers to determining the opinions or sentiments expressed on different features or aspects of entities, e.g., of a cell phone, a digital camera, or a bank.[14] A feature or aspect is an attribute or component of an entity, e.g., the screen of a cell phone, the service for a restaurant, or the picture quality of a camera. The advantage of feature-based sentiment analysis is the possibility to capture nuances about objects of interest. Different features can generate different sentiment responses, for example a hotel can have a convenient location, but mediocre food.[15] This problem involves several sub-problems, e.g., identifying relevant entities, extracting their features/aspects, and determining whether an opinion expressed on each feature/aspect is positive, negative or neutral.[16] The automatic identification of features can be performed with syntactic methods or with topic modeling.[17][18] More detailed discussions about this level of sentiment analysis can be found in Liu's work.[19]

**5.1.2 Methods and features**

Existing approaches to sentiment analysis can be grouped into three main categories: knowledge-based techniques, statistical methods, and hybrid approaches.[20] Knowledge-based techniques classify text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored.[21] Some knowledge bases not only list obvious affect words, but also assign arbitrary words a probable "affinity" to particular emotions.[22] Statistical methods leverage on elements from machine learning such as latent semantic analysis, support vector machines, "bag of words" and Semantic Orientation — Pointwise Mutual Information (See Peter Turney's[1] work in this area). More sophisticated methods try to detect the holder of a sentiment (i.e., the person who maintains that affective state) and the target (i.e., the entity about which the affect is felt).[23] To mine the opinion in context and get the feature which has been opinionated, the grammatical relationships of words are used. Grammatical dependency relations are obtained by deep parsing of the text.[24] Hybrid approaches leverage on both machine learning and elements from knowledge representation such as ontologies and semantic networks in order to detect semantics that are expressed in a subtle manner, e.g., through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.[25]

Open source software tools deploy machine learning, statistics, and natural language processing techniques to automate sentiment analysis on large collections of texts, including web pages, online news, internet discussion groups, online reviews, web blogs, and social media.[26] Knowledge-based systems, on the other hand, make use of publicly available resources, to extract the semantic and affective information associated with natural language concepts. Sentiment analysis can also be performed on visual content, i.e., images and videos. One of the first approach in this direction is SentiBank[27] utilizing an adjective noun pair representation of visual content.

A human analysis component is required in sentiment analysis, as automated systems are not able to analyze historical tendencies of the individual commenter, or the platform and are often classified incorrectly in their expressed sentiment. Automation impacts approximately 23% of comments that are correctly classified by humans.[28] However, also humans often disagree, and it is argued that the inter-human agreement provides an upper bound that automated sentiment classifiers can eventually reach.[29]

Sometimes, the structure of sentiments and topics is fairly complex. Also, the problem of sentiment analysis is non-monotonic in respect to sentence extension and stop-word substitution (compare THEY would not let my dog stay in this hotel vs I would not let my dog stay in this hotel). To address this issue a number of rule-based and reasoning-based approaches have been applied to sentiment analysis, including defeasible logic programming.[30] Also, there is a number of tree traversal rules applied to syntactic parse tree to extract the topicality of sentiment in open domain setting.[31][32]

**5.1.3 Evaluation**

The accuracy of a sentiment analysis system is, in principle, how well it agrees with human judgments. This is usually measured by precision and recall. However, according to research human raters typically agree 79%[33] of the time (see Inter-rater reliability).

Thus, a 70% accurate program is doing nearly as well as humans, even though such accuracy may not sound impressive. If a program were "right" 100% of the time, humans would still disagree with it about 20% of the time, since they disagree that much about any answer .[34] More sophisticated measures can be applied, but evaluation of sentiment analysis systems remains a complex matter. For sentiment analysis tasks returning a scale rather than a binary judgement, correlation is a better measure than precision because it takes into account how close the predicted value is to the target value.

**5.1.4 Web 2.0**

The rise of social media such as blogs and social networks has fueled interest in sentiment analysis. With the proliferation of reviews, ratings, recommendations and other forms of online expression, online opinion has turned into a kind of virtual currency for businesses looking to market their products, identify new opportunities and manage their reputations. As businesses look to automate the process of filtering out the noise, understanding the conversations, identifying the relevant content and actioning it appropriately, many are now looking to the field of sentiment analysis.[35] Further complicating the matter, is the rise of anonymous social media platforms such as 4chan and Reddit.[36] If web 2.0 was all about democratizing publishing, then the next stage of the web may well be based on democratizing data mining of all the content that is getting published.[37]

One step towards this aim is accomplished in research. Several research teams in universities around the world currently focus on understanding the dynamics of sentiment in e-communities through sentiment analysis.[38] The CyberEmotions project, for instance, recently identified the role of negative emotions in driving social networks discussions.[39]

The problem is that most sentiment analysis algorithms use simple terms to express sentiment about a product or service. However, cultural factors, linguistic nuances and differing contexts make it extremely difficult to turn a string of written text into a simple pro or con sentiment.[35] The fact that humans often disagree on the sentiment of text illustrates how big a task it is for computers to get this right. The shorter the string of text, the harder it becomes.

Even though short text strings might be a problem, sentiment analysis within microblogging has shown that Twitter can be seen as a valid online indicator of political sentiment. Tweets' political sentiment demonstrates close correspondence to parties' and politicians' political positions, indicating that the content of Twitter messages plausibly reflects the offline political landscape.

**5.2 Source Code**

**5.2.1 Retrieving tweets**

from twython import TwythonStreamer

from unidecode import unidecode

import sys

tweet\_text = []

num=20

CONSUMER\_KEY = '1XlO7bDLYARB1tTjZ4cspV1Q2'

CONSUMER\_SECRET = 'othIx7WctmMQ38NEwXja2hKGsamKpVeohvAGgQfuxXoZlyFoZ3'

ACCESS\_TOKEN = '846796640811933697-ttfVB5ekh6zLLh73DdUjFWy5ZpEjYWH'

ACCESS\_TOKEN\_SECRET = 'UJqdU8ZrjR7FmcsJlReJMmI2KcMKFyVxz4d5eLk0F33qA'

tweet\_locations = {

'dfw': "-97.48,32.31,-96.49,33.26,",

'jfk': "-74,40,-73,41",

'sfo': "-122.75,36.8,-121.75,37.8",

'lax': "-118.43,33.73,-117.93,34.21",

'ord': "-88.47,41.46,-87.16,42.35",

'dca': "-77.26,38.73,-76.87,39.10",

'atl': "-84.54,33.62,-84.18,33.96"

}

class MyStreamer(TwythonStreamer):

# overriding

def on\_success(self, data):

if data['lang'] == 'en':

# filter tweet

twitter\_words = [u'http',u'https',u'RT','https','http',u's://t.co','rt','amp',u'amp',u'u2026',u'u2019']

text=unidecode(data['text'])

for word in twitter\_words:

text=text.replace(word,'')

text=text.strip()

text=text.replace('\n','') #to remove newline

tweet\_text.append(text)

if len(tweet\_text) >= num:

self.disconnect()

# overriding

def on\_error(self, status\_code, data):

print(status\_code, data)

self.disconnect()

# download

keyword = ‘Trump’

location = 'sfo'

bounds = tweet\_locations[location]

stream = MyStreamer(CONSUMER\_KEY, CONSUMER\_SECRET, ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET)

stream.statuses.filter(track=keyword,locations=bounds)

f=open(Trump/sfo.txt','w')

for i in range(num):

f.write(tweet\_text[i]+'\n')

f.close()

print(len(tweet\_text))

**5.2.2 Sentiment analysis on tweets**

from textblob import TextBlob

class Sem\_Analysis:

count\_positive = 0

count\_negative = 0

count\_neutral = 0

Average\_polarity = 0

Average\_subjectivity = 0

percent\_positive=''

percent\_negative=''

percent\_neutral=''

def sent\_tweet(self,fname):

count\_positive = 0

count\_negative = 0

count\_neutral = 0

Average\_polarity = 0

Average\_subjectivity = 0

total\_tweets=0

lst1=[]

lst2=[]

f = open(fname,'r')

f1= f.readlines()

for line in f1:

sent1 = TextBlob(line)

tb1 = sent1.sentiment.polarity

lst1.append(tb1)

tb2 = sent1.sentiment.subjectivity

lst2.append(tb2)

for item in lst1:

if item>0:

count\_positive += 1

elif item<0:

count\_negative +=1

else:

count\_neutral +=1

self.count\_positive = count\_positive

self.count\_negative = count\_negative

self.count\_neutral = count\_neutral

total\_tweets = count\_positive + count\_negative + count\_neutral

self.percent\_positive = str(round(count\_positive/total\_tweets\*100 ,4)) + "%"

self.percent\_negative = str(round(count\_negative/total\_tweets\*100 ,4)) +"%"

self.percent\_neutral = str(round(count\_neutral/total\_tweets\*100 ,4)) +"%"

Average\_polarity = sum(lst1)/len(lst1)

self.Average\_polarity = round(Average\_polarity,4)

Average\_subjectivity = sum(lst2)/len(lst2)

self.Average\_subjectivity = round(Average\_subjectivity,4)

**5.2.3 Initialize plotly library in offline mode**

import plotly

plotly.offline.init\_notebook\_mode()

**5.2.4 Creating table**

import plotly as py

import plotly.figure\_factory as ff

import pandas as pd

trump = Sem\_Analysis()

trump.sent\_tweet("Trump/sfo.txt")

hillary = Sem\_Analysis()

hillary.sent\_tweet("Hillary/sfo.txt")

bernie = Sem\_Analysis()

bernie.sent\_tweet("tweetsbernie.txt")

data\_matrix = [['Name', 'Number of<br>Positive<br>comments', 'Number of<br>Neutral<br>comments','Number of<br>Negative<br>comments',

'% Positive ','% Neutral','% Negative ',

'Average<br>Polarity','Average<br>Subjectivity'],

['Trump', trump.count\_positive, trump.count\_neutral,trump.count\_negative,trump.percent\_positive,

trump.percent\_neutral,trump.percent\_negative,

trump.Average\_polarity, trump.Average\_subjectivity],

['Hillary', hillary.count\_positive, hillary.count\_neutral, hillary.count\_negative, hillary.percent\_positive,

hillary.percent\_neutral,hillary.percent\_negative,

hillary.Average\_polarity, hillary.Average\_subjectivity],

['bernie', bernie.count\_positive, bernie.count\_neutral, bernie.count\_negative,bernie.percent\_positive,

bernie.percent\_neutral,bernie.percent\_negative, bernie.Average\_polarity, bernie.Average\_subjectivity]

]

table = ff.create\_table(data\_matrix)

table.layout.width=1000 #width in pixels

py.offline.iplot(table, filename='jupyter/table')

**5.2.5 Creating Bar Graph**

import plotly as py

import plotly.graph\_objs as go

trump = Sem\_Analysis()

trump.sent\_tweet("Trump/sfo.txt")

hillary = Sem\_Analysis()

hillary.sent\_tweet("tweetshillary.txt")

bernie = Sem\_Analysis()

bernie.sent\_tweet("tweetsbernie.txt")

trace1 = go.Bar(

x=['Positive', 'Neutral', 'Negative'],

y=[trump.count\_positive, trump.count\_neutral,trump.count\_negative],

name='Trump'

)

trace2 = go.Bar(

x=['Positive', 'Neutral', 'Negative'],

y=[hillary.count\_positive, hillary.count\_neutral, hillary.count\_negative],

name='Hillary'

)

trace3 = go.Bar(

x=['Positive', 'Neutral', 'Negative'],

y=[bernie.count\_positive, bernie.count\_neutral, bernie.count\_negative],

name='Bernie'

)

data = [trace1, trace2,trace3]

layout = go.Layout(

barmode='group',

bargroupgap=0.1,

title='Comparing Sentiments in USA',

xaxis=dict(

title='Sentiment',

titlefont=dict(

size=16,

color='rgb(107, 107, 107)'

)),

yaxis=dict(

title='Number of tweets<br>',

titlefont=dict(

size=16,

color='rgb(107, 107, 107)'

)),

)

fig = go.Figure(data=data, layout=layout)

py.offline.iplot(fig, filename='grouped-bar-graph')

**5.2.6 Creating Wordcloud**

#wordcloud

from os import path

from wordcloud import WordCloud

import matplotlib.pyplot as plt

import nltk

stopwords = nltk.corpus.stopwords.words('english')

from nltk.stem.snowball import SnowballStemmer

ss = SnowballStemmer("english")

text = open('tweetshillary.txt').read()

text2 = ''

for word in text.split():

if len(word) == 1 or word in stopwords:

continue

text2 += ' {}'.format(word)

ss.stem(text2)

wordcloud = WordCloud(max\_font\_size=150,width=800,height=400).generate(text)

plt.figure(figsize=(20,10))

plt.imshow(wordcloud)

plt.axis("off")

plt.show()